Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) iii-iv

Procedia Computer Science

www.elsevier.com/locate/procedia

Table of Contents

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 1–2

Procedia Computer Science

www.elsevier.com/locate/procedia

Preface

It gives me great pleasure to present this 3rd edition of our book of proceedings which gathers some of the very best work completed using EuroHPC JU resources in 2025. I thank the Editor-in-Chief, Krishnakshi Bhuyan, and the reviewers for their commitment and effort, which were instrumental in the successful completion of this book of proceedings.

Central to the activities of EuroHPC JU is the support of the uptake of demand-oriented and user-driven innovation and talent development.

Since its creation in 2018, EuroHPC JU allows the European Union, and its participating countries, to coordinate their efforts and pool resources to deploy technological advancements in the field of supercomputing. The success of these endeavours is measured in the first and foremost in the science and innovations produced using the EuroHPC systems and in the growing number of scientific and industrial users from across Europe who have accessed and leveraged these resources to advance scientific, social and business innovation and knowledge.

User support services funded by EuroHPC JU such as National Competence Centres (NCCs) and Centres of Excellence (CoEs), as well as dedicated skills initiatives such as EPICURE, Minerva, Fortissimo Plus (FFplus) and Evita, guide new users to EuroHPC supercomputers.

The papers submitted to this publication span a wide range of disciplinary areas such as Computational physics, astrophysics, engineering, earth sciences and climate, artificial intelligence, and life sciences. As such, they are testament to the diverse research and innovation environment linked to EuroHPC JU activities.

The papers were also presented at the third EuroHPC User Days which took place on 30 September and 1 October 2025 in Copenhagen, Denmark. EuroHPC User Days is EuroHPC's annual event bringing together the European user communities and celebrating innovative research projects undertaken on EuroHPC's supercomputing infrastructure.

Building on this foundation of user-driven innovation and diverse research support, EuroHPC JU continues to expand its ecosystem through new strategic initiatives.

In the future, users will have more EuroHPC compute resources to choose from:

- EuroHPC's Quantum computers will soon come online, opening the way for new applications to span both HPC and Quantum systems.
- The EuroHPC JU is also overseeing the implementation of 13 AI factories (with more to be announced) across Europe that offer free, customised support to SMEs and startups. These comprehensive open AI ecosystems will initially be located around existing EuroHPC supercomputing facilities. They will support the growth of a highly competitive and innovative AI ecosystem in Europe, further expanding the available resources for new and existing user communities. They will also contribute to positioning Europe at the forefront of the emerging field of quantum and hybrid quantum, HPC, and AI innovation.
- The delivery of the Federation Platform will facilitate a single access point to data lakes and data spaces across Europe by seamlessly integrating both private and public solutions including well-established platforms like SIMPL, EOSC, and FENIX.

Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day
10.1016/j.procs.2025.08.227

Annual events like EuroHPC User Days bring to life the work we do at EuroHPC JU. The publication of the Proceedings creates a record of some of the best papers and shares this work with audiences well beyond the scope of the event. EuroHPC JU is grateful to all of the researchers who submitted their work to be part of this publication. Papers were selected on the basis of their work demonstrating the transformative potential of EuroHPC supercomputers. Peer review of the received manuscripts was performed by an editorial board of scientific reviewers.

EuroHPC JU extends its gratitude to the presenters and the organising partners who contributed to the success of the EuroHPC User Day 2025. Additionally, we thank Elsevier for their support in enabling this publication.

Anders Dam Jensen, Executive Director, EuroHPC JU

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 3

Procedia Computer Science

www.elsevier.com/locate/procedia

Editorial Board

Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day
10.1016/j.procs.2025.08.228

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 4–13

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

The QCD crossover line in a finite volume

Szabolcs Borsányi[a,*], Zoltán Fodor[b,a,c,d], Jana N. Guenther[a], Paolo Parotto[e], Attila Pásztor[c], Ludovica Pirelli[a], Kálmán K. Szabó[a,d], Chik Him Wong[a]

[a] *Department of Physics, Wuppertal University, Gaussstr. 20, D-42119, Wuppertal, Germany*
[b] *Pennsylvania State University, Department of Physics, State College, PA 16801, USA*
[c] *Institute for Theoretical Physics, ELTE Eötvös Loránd University, Pázmány P. sétány 1/A, H-1117 Budapest, Hungary*
[d] *Jülich Supercomputing Centre, Forschungszentrum Jülich, D-52425 Jülich, Germany*
[e] *Dipartimento di Fisica, Università di Torino and INFN Torino, Via P. Giuria 1, I-10125 Torino, Italy*

Abstract

We compute the crossover line of strongly interacting matter in the temperature (T) – baryo-chemical potential (μ_B) plane of the QCD phase diagram. We use large scale lattice QCD simulations at vanishing μ_B to solve the path integral in the transition regime between $T = 120$ and 200 MeV. We introduce a high order Taylor scheme for the Polyakov loop and extract the static quark entropy. The peak position of the latter defines the transition line that we can extract up to 400 MeV in μ_B. For the simulations the 4HEX staggered action was used with 2+1 flavors at physical quark masses.

Keywords: Lattice QCD ; QCD phase diagram

1. Introduction

Vacuum heated to the extreme temperatures of $\sim 10^{12}$ K turns into a new phase where matter and antimatter and their force carriers form a fluid, the Quark Gluon Plasma (QGP). As its name suggests quark fields constitute the matter component and gluons mediate their interaction. If we increase the temperature, interactions between quarks weaken, if we lower the temperature quarks and antiquarks will be confined into hadrons and anti-hadrons, respectively, forming a weakly interacting gas of baryons and mesons without visible gluonic terms. The transition from QGP to the hadronic phase has actually happened in our Universe when it was ~ 10 μs old. The same transition has been reproduced several billion times in the Large Hadron Collider (LHC) at CERN as well as in the Relativistic Heavy Ion Collider (RHIC) at the Brookhaven National Lab (BNL).

There is ample experimental evidence for this transition being an analytical crossover [1, 2, 3]. Theory has predicted the crossover nature of the transition [4]. In fact, one can apply statical field theory methods to solve the path

* Corresponding author. Tel.: +49-202-439-2635
E-mail address: borsanyi@uni-wuppertal.de

1877-0509
10.1016/j.procs.2025.08.229

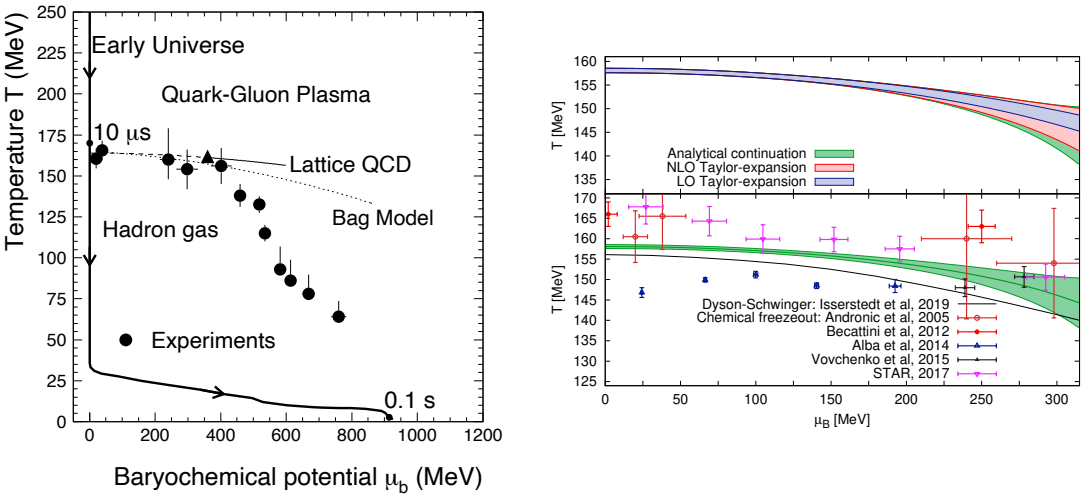


Fig. 1. The QCD phase diagram in the $T - \mu_B$ plane. Left: a version from 2008 [8], right: our recent continuum extrapolated lattice result [9]. The data points refer to the chemical freeze-out line in both figures. The green band on the right plot is our lattice result, the other lines refer to various theoretical approaches.

integral of Quantum Chromodynamics (QCD) the underlying theory to strong interactions. Lattice QCD, a computational discipline within theoretical high energy physics, can implement importance sampling to generate field configurations with the weight given by the Euclidean quantum field theory. With such lattice simulations the central temperature of the crossover has been determined to be $T \approx 155$ MeV [5, 6].

The beam energy in collision experiments controls the baryon stopping and through that the excess of quarks over antiquarks in the plasma. This gave the opportunity for a Beam Energy Scan (BES) program at RHIC, where the transition could be studied as a function of net baryon density. However, lattice simulations struggle with an asymmetric setup between matter and antimatter. Most theoretical frameworks introduce a baryo-chemical potential μ_B as the free energy associated with the shifting of the balance by three quarks. While for vanishing values of μ_B the statistical weight of the field configurations are real and positive, allowing for a stochastic process to sample it, lattice QCD faces with complex probabilities whenever a grand canonical simulation with non-zero μ_B is asked for. Though it is possible to restrict the simulation to the real part based on exact cancellation of the imaginary part, the physics result is encoded in the approximate cancellation of large positive and negative real contributions [7]. This is called the sign problem in finite density lattice QCD. It does not immediately make simulations at finite μ_B impossible, but drives the costs exponentially high both in μ_B^2 and in the system size.

Given the rich experimental program at RHIC and the future CBM experiment at FAIR the theory community is challenged to make reliable predictions for finite density QCD near its crossover temperature. One key feature in the $T - \mu_B$ phase diagram is the transition line (see Fig. 1). We already know from lattice simulations that it starts at the temperature axis ($\mu_B = 0$) as a crossover at $T \approx 155$ MeV.

The data points in Fig. 1 refer to experimental observations of the collision outcome. Particle abundances are matched against a thermal model based on the grand canonical parameters T and μ_B. These are the chemical freeze-out parameters and reflect the temperature of the matter at the moment of the last inelastic scattering. While the underlying model assuming an instantaneous disintegration of the thermal plasma may seem oversimplified, it is very successful in the description of the particle yields and gives results close to lattice transition line [10].

Our earlier lattice result in Fig. 1 is based on the analytical continuation method [11, 12]. It exploits that for imaginary valued chemical potentials the weight of individual lattice configurations are still real and positive, allowing importance sampling to work for $\mu_B^2 < 0$. The transition temperatures were determined for several fixed negative $(\mu_B/T)^2$ values and were extrapolated to the physical side with $\mu_B^2 > 0$. The crossover line was first computed in the continuum limit in Ref. [13].

Fig. 1 also shows an abrupt change of the chemical freeze-out line near $\mu_B \approx 400$ MeV. It is not clear, whether the lattice extrapolation breaks down, the thermal model's assumptions are no longer valid, or there is a highly compressed hadronic phase above the chemical freeze-out line. This region will be explored in detail by the CBM experiment in the near future.

It is reasonable to assume that the crossover line may turn in to a first order line separated by a critical end-point. If it happens, fluctuation measurements of net baryon (or proton) density will show a possibly large non-monotonic signal as a function of beam energy [14]. One of the science goals of the Beam Energy Scan program of the STAR experiment was to find these. As it happens, Nature has placed the critical endpoint outside of STAR's range [15].

The intense experimental research has stimulated significant developments in the theoretical frameworks. While lattice QCD was experimenting with novel ways to extrapolate or directly simulate at finite μ_B, diagrammatic approaches, such as the Dyson-Schwinger method [16] and similar ways based on the functional renormalization group [17] have delivered promising results.

Our work presented in this paper is motivated by the urgent need to follow up on the crossover line and search for the signatures of criticality. In section 2 we start with the role of the simulation volume and outline the simulation strategy in 3. After presenting various cross-checks in section 4 we present a new result on the transition line in section 5. We finally discuss further planned use of the generated data in section 6.

2. Simulation volume and the sign problem

A popular and successful method to extract finite density information from simulations that were run at $\mu_B = 0$ is to compute Taylor expansion coefficients. To illustrate this we consider the pressure, normalized to temperature

$$\left.\frac{p}{T^4}\right|_{T,\mu_B} = \left.\frac{p}{T^4}\right|_{T,0} + \frac{\hat{\mu}_B^2}{2!}\chi_2(T) + \frac{\hat{\mu}_B^4}{4!}\chi_4(T) + \frac{\hat{\mu}_B^6}{6!}\chi_6(T) + \dots \tag{1}$$

where the coefficients themselves are highly complex lattice observables. We used the shorthand $\hat{\mu} = \mu/T$. For instance, $\chi_2(T)$ is obtained as the variance of the quark number on the configurations, as dictated by the fluctuation-response theorem.

What we call a lattice configuration contains the bosonic fields (gluons) only. The fermionic part (quarks) is quadratic and can be integrated out leaving us with a non-local bosonic theory that we actually simulate. To extract variance, kurtosis or even higher moments of the quark number (for χ_6) we have to revisit the fermionic contribution and find out the Taylor expansion for each configuration U. Since the quark contribution is a determinant, $\det M(U,\mu)$, its expansion can be written as

$$\begin{aligned}\log\det M_j^{1/4}(U,m_j,\mu_j) = \log\det M_j^{1/4}(U,m_j,0) + A_j\hat{\mu}_j \\ + \frac{1}{2!}B_j\hat{\mu}_j^2 + \frac{1}{3!}C_j\hat{\mu}_j^3 + \dots\end{aligned} \tag{2}$$

Notice, that wile charge conjugation symmetry forbids the odd term on the level of a physical observable, such as the pressure, Eq. (2) is not protected. In Eq. (2) we introduced a separate chemical potential for each quark flavor, since their numbers are conserved separately. Dropping the flavor index for simplicity we find for the first few orders

$$\begin{aligned}
\chi_2 &= +\langle B\rangle + \langle AA\rangle \\
\chi_4 &= +\langle D\rangle + 3\langle BB\rangle - 3\langle B\rangle\langle B\rangle + 4\langle AC\rangle + \langle AAAA\rangle - 3\langle AA\rangle\langle AA\rangle + 6\langle AAB\rangle - 6\langle B\rangle\langle AA\rangle \\
\chi_6 &= +\langle F\rangle + 10\langle CC\rangle + 15\langle BD\rangle + 15\langle BBB\rangle + 6\langle AE\rangle + 60\langle ABC\rangle + 15\langle AAD\rangle \\
&\quad +45\langle AABB\rangle + 20\langle AAAC\rangle + 15\langle AAAAB\rangle + \langle AAAAAA\rangle - 15\langle D\rangle\langle B\rangle \\
&\quad -15\langle D\rangle\langle AA\rangle - 45\langle B\rangle\langle BB\rangle - 60\langle B\rangle\langle AC\rangle - 90\langle B\rangle\langle AAB\rangle \\
&\quad -15\langle B\rangle\langle AAAA\rangle - 45\langle BB\rangle\langle AA\rangle - 60\langle AC\rangle\langle AA\rangle - 90\langle AA\rangle\langle AAB\rangle \\
&\quad -15\langle AA\rangle\langle AAAA\rangle + 30\langle B\rangle\langle B\rangle\langle B\rangle + 90\langle B\rangle\langle B\rangle\langle AA\rangle \\
&\quad +90\langle B\rangle\langle AA\rangle\langle AA\rangle + 30\langle AA\rangle\langle AA\rangle\langle AA\rangle \\
\chi_8 &= 79 \text{ terms} \dots
\end{aligned}$$

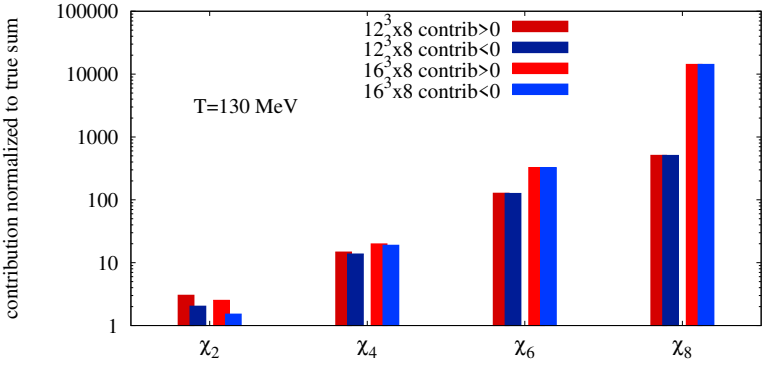


Fig. 2. Remnant sign problem in the Taylor coefficients of the finite-μ_B extrapolation for two volumes. We show the sum of positive and negative contributions as two bars, their difference is the actual value of χ_n. With every new order the signal-to-noise ratio deteriorates with a volume-dependent factor.

High orders in μ_B (where all quarks flavors are considered) are very tedious to obtain. We developed a computer algebra system to compute these and to act as a code generator. The biggest problem, however, is not computational complexity, but the cancellation between positive and negative contributions within the expressions for χ_n. This is a remnant of the finite density sign problem (see Fig. 2).

It is clear from Fig. 2 that large simulation volumes turn the high order extrapolation schemes impractical. Past lattice studies have often used large volumes e.g. $32^3 \times 8$, then even extreme statistics could not reduce the large statistical error on χ_6 and χ_8 [18]. In our recent work we started with the much smaller $16^3 \times 8$ simulation size and could compute the high orders in the continuum limit [19].

But before we proceed with a small system size to obtain the crossover line we have to find answer two very practical questions: which observable can be used to identify the transition temperature in a small volume, and how can this be extrapolated to finite density?

The first question we addressed in Ref. [20]. In Fig. 3 we show the main result. We investigated five different observables and plotted the transition temperature as a function of the volume (at vanishing μ_B). Three of these observables are based on chiral observables ($\chi_{\text{full}}, \chi_{\text{disc}}$ and pbp for $\langle\bar{\psi}\psi\rangle$), two are related to the Polyakov loop as

$$F_Q = -T \log\left(\frac{1}{V}\sum_{\vec{x}} |\langle P(\vec{x})\rangle_T|\right) \tag{3}$$

$$S_Q = -\frac{\partial F_Q}{\partial T}, \tag{4}$$

where $P(\vec{x})$ is the Polyakov loop

$$P(\vec{x}) = \frac{1}{3}\mathrm{Tr}\prod_{x_4=0}^{N_t-1} U_4(\vec{x}, x_4), \tag{5}$$

the $U_\mu(\vec{x}, x_4)$ are the usual link variables in lattice gauge theory.

In this work we focus on the Polyakov loop and the $S_Q(T)$ observable.

3. Transition temperature at finite density

Our next task is to compute $S_Q(T)$ at finite μ_B. For this we will work out the Taylor series of the real and imaginary parts of the Polyakov loop and through that we will find the expansion coefficients of $F_Q(T)$. We will use these to obtain $F_Q(T, \mu_B)$ to eight order and find the crossover temperature by finding its inflection point.

We start by defining the chain rule for an arbitrary variable X. Following Ref. [21] we have:

$$\partial_j \langle X\rangle = \left\langle XA_j\right\rangle - \langle X\rangle\left\langle A_j\right\rangle + \left\langle \partial_j X\right\rangle, \tag{6}$$

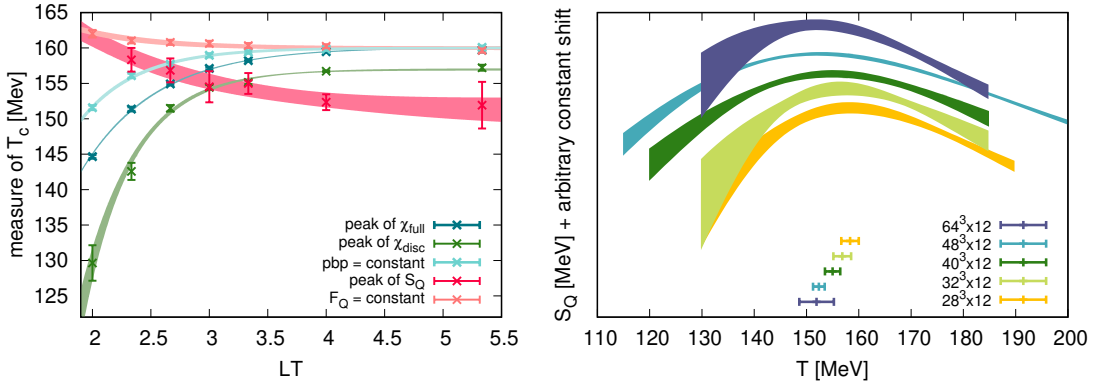


Fig. 3. Left: Five observables that can be used as a proxy for the transition temperature. Right: The static quark entropy exhibits a peak at the transition. The volume dependence is mild. (Plots from Ref. [20]).

$$\begin{aligned}\partial_k\partial_j\langle X\rangle = &\frac{1}{2}\left\langle XA_jA_k\right\rangle - \frac{1}{2}\langle X\rangle\left\langle A_jA_k\right\rangle - \left\langle XA_j\right\rangle\langle A_k\rangle\\ &+\langle X\rangle\langle A_k\rangle\left\langle A_j\right\rangle + \left\langle(\partial_k X)A_j\right\rangle - \langle\partial_k X\rangle\langle A_k\rangle\\ &+\frac{1}{2}\left\langle\partial_k\partial_j X\right\rangle - \frac{1}{2}\langle X\rangle\left\langle\partial_k A_j\right\rangle\\ &+(j\leftrightarrow k)\,.\end{aligned} \tag{7}$$

where ∂_j stands for $\partial/\partial\hat{\mu}_j$ and we used a flavor-specific coefficients $A_j, B_j, \ldots$ for the quark determinant $\det M_j(T,\mu_j)$. These directly yield the derivatives

$$\partial_j\langle P_R\rangle\big|_{\mu\equiv 0} = 0\,, \tag{8}$$

$$\partial_j\langle P_I\rangle\big|_{\mu\equiv 0} = \left\langle A_jP_I\right\rangle\,, \tag{9}$$

$$\begin{aligned}\partial_j\partial_k\langle P_R\rangle\big|_{\mu\equiv 0} = &\,\delta_{jk}\left[\left\langle B_jP_R\right\rangle - \left\langle B_j\right\rangle\langle P_R\rangle\right]\\ &\left\langle A_jA_kP_R\right\rangle - \left\langle A_jA_k\right\rangle\langle P_R\rangle\,,\end{aligned} \tag{10}$$

$$\partial_j\partial_k\langle P_I\rangle\big|_{\mu\equiv 0} = 0\,. \tag{11}$$

or for F_Q one can work out the expansion scheme

$$F = -\frac{T}{2}\log|\langle P\rangle|^2 = T\sum_{n=0,2,\ldots}\frac{F_n}{n!}\left(\frac{\mu}{T}\right)^n \tag{12}$$

$$F_0 = -\frac{1}{2}\log(Q) \tag{13}$$

$$F_2 = -\frac{1}{2}\left[+\frac{Q^{(2)}}{Q}\right] \tag{14}$$

For higher orders we used a code generator. The complete list of formulas can be found in Ref. [22]. We show the resulting coefficients as a function of temperature in Fig. 4.

Fig. 4 shows the results for two different expansion schemes. The red data points represent our results in the straightforward μ_B expansion scheme. The blue points also refer to coefficients in μ_B, however, in that case the strangeness chemical potential μ_S is tuned order-by-order such that the resulting strangeness density vanishes in the complete parameter range. The latter setup is the closest to the experimentally studied case, and it was already used in Fig. 1.

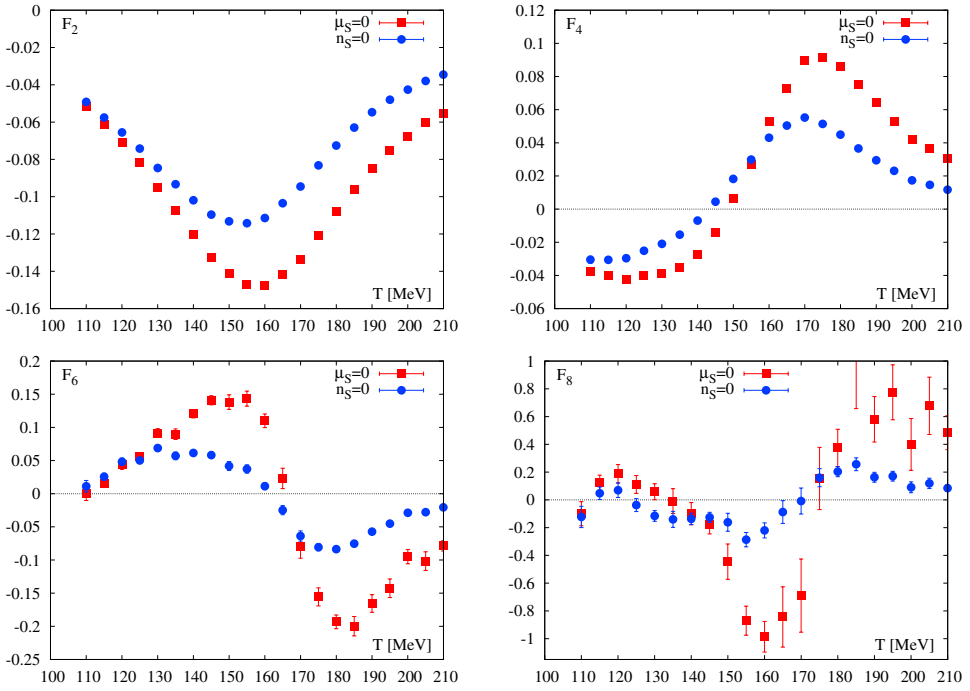


Fig. 4. Taylor coefficients of the static quark free energy in two setups: vanishing strangeness chemical potential in red, along strangeness neutral trajectories in blue (Updated version of the original plots in Ref. [22])

4. Validity of the extrapolation

To check the validity of the extrapolation we compare our extrapolated results to direct simulation data. As we have already mentioned, simulations with negative μ_B^2 are technically possible and we supplemented our data set with such runs so that such a comparison can be made.

In the following we will use the already determined expansion coefficients and insert negative μ_B^2 in the Taylor formula. We plot the resulting real and imaginary parts of the Polyakov loop against direct data in Fig. 5.

As an additional check we compute the bare $F_Q(\mu_B)$ for imaginary chemical potentials. This quantity requires renormalization, but this only applies a constant shift to $F_Q(\mu_B)$, it plays no role for this comparison. In order to obtain direct data along the strangeness neutral line at imaginary μ_B, we performed a reweighting of each Im $\mu_B > 0$ ensemble to the specific imaginary μ_S where $n_S = 0$. The target $\mu_S(\mu_B)$ was also computed using reweighting, with reweighting factors spread between 0.5 and 2. Even in the $n_S = 0$ scheme, our error bars are smaller than the symbol size. We stress that reweighting was only applied to the imaginary μ_B data and only for the sake of this crosscheck. We will refer to the resulting F_Q for both schemes as direct data, as opposed to the Taylor expansion, that is based on the $\mu_B = 0$ ensemble. We show the comparison in the right panel of Fig. 5.

5. A new estimate for the crossover line

We start with the presentation of the renormalized Polyakov loop, since most readers are more familiar with this observable than F_Q or S_Q. Since we have already established F_Q at finite μ_B we define the renomalized Polyakov loop as $P^r = \exp -(F_Q - \Delta)/T$, where Δ sets the scheme. This means that for observables in this paper we always expand F_Q, not P_R or P_I. We choose Δ such that $P^r(T = 160\ \text{MeV}, \mu_B = 0) = 1$ as we did in Ref. [20]. The Polyakov loop curves are shown in Fig. 6. In the bottom panel of this figure, we also show the fourth, sixth and eighth order curves for $T = 130$ MeV. The expansion is under control below $\mu_B \approx 400$ MeV.

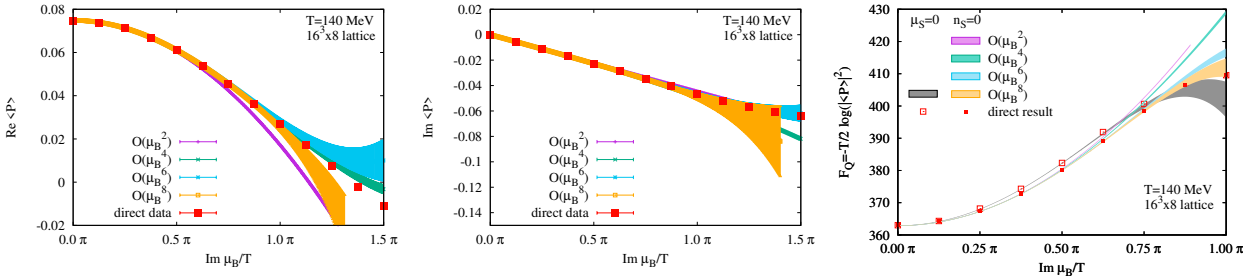


Fig. 5. The Taylor coefficients are used to extrapolate the bare Polyakov loop expectation value from $\mu_B = 0$ to imaginary values of the baryochemical potential. (Left: real part, Middle: imaginary part, Right: the resulting static quark potential F_Q) We demonstrate the convergence of the series at $T = 140$ MeV by comparing orders of the extrapolations (bands) to directly measured expectation values at Im $\mu_B > 0$ from a separate set of simulations. In the right plot we show two comparisons, one in the $\mu_S = 0$ scheme, and one in the strangeness netural $n_S = 0$ case.

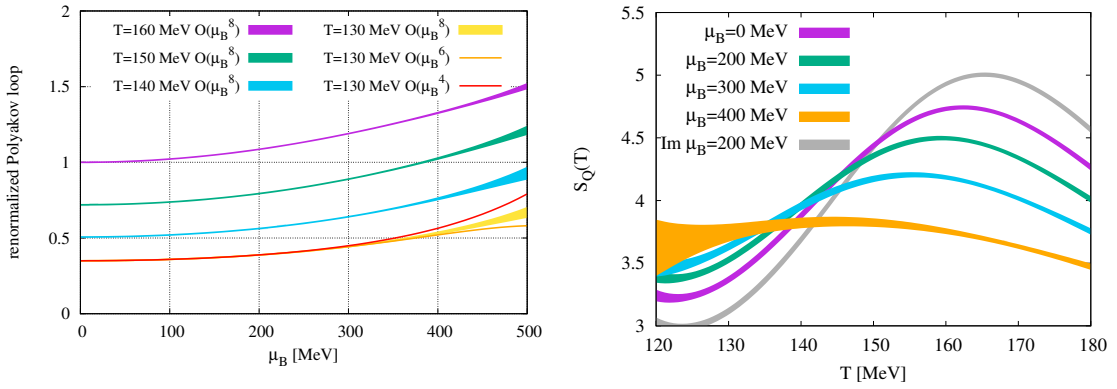


Fig. 6. Left: The Polyakov loop shown as a function of μ_B with fixed T. For one of the temperatures we show the fourth and sixth order results. The sixth order does not visibly differ from the full computation below $\mu_B = 350$ MeV. Right: The static quark entropy ($S_Q(T)$) extrapolated to various chemical potentials. The errors in this plot are statistical only. We added one imaginary chemical potential (gray band) to better visualize the trend in μ_B.

In the right panel of Fig. 6 we show $S_Q(T)$ as defined in Eq. (4). We also show $S_Q(T)$ for one imaginary value of μ_B for comparison. The $S_Q(T)$ curves show two trends very clearly. First, the change in the peak position indicates the chemical potential dependence of the transition temperature. Second, the width of the peak increases as μ_B grows, indicating a weakening of the deconfinement transition. At the largest value of μ_B, the curve is the flattest. This feature contributes to the difficulty in determining the deconfinement temperature at large μ_B, in addition to the growing errors from the extrapolation. On the other hand, in the opposite (imaginary) direction, the peak is higher and narrower, eventually becoming singular in the Roberge-Weiss point [23].

Our final results are the deconfinement transition temperature and the width of the deconfinement transition as a function of the chemical potential. For the strangeness neutral case they are shown in Fig. 7. On the left, the deconfinement crossover temperature calculated in this work is compared to the chiral crossover temperature from Ref. [9] and an estimate of the freeze-out parameters in heavy ion collisions by the STAR collaboration [25] and the parametrization in Ref. [10]. On the right we show the chiral transition width from Ref. [9] and the deconfinement width from this work. Already at $\mu_B = 0$ the deconfinement transition is much broader. Even more interestingly, as a function of the baryochemical potential, the deconfinement transition broadens, while the width of the chiral transition stays roughly the same. This contrasting behaviour was already observed in our earlier work [20], but only to leading order in μ_B. Here, we observe the same behavior in a large range of chemical potential. Our results disfavor the existence of a deconfinement critical endpoint below $\mu_B \approx 400$MeV. The same statement is also true for zero strangeness chemical potential. However, in that setup, we see the reverse trend of the width. Above $\mu_B \approx 300$MeV the width of the

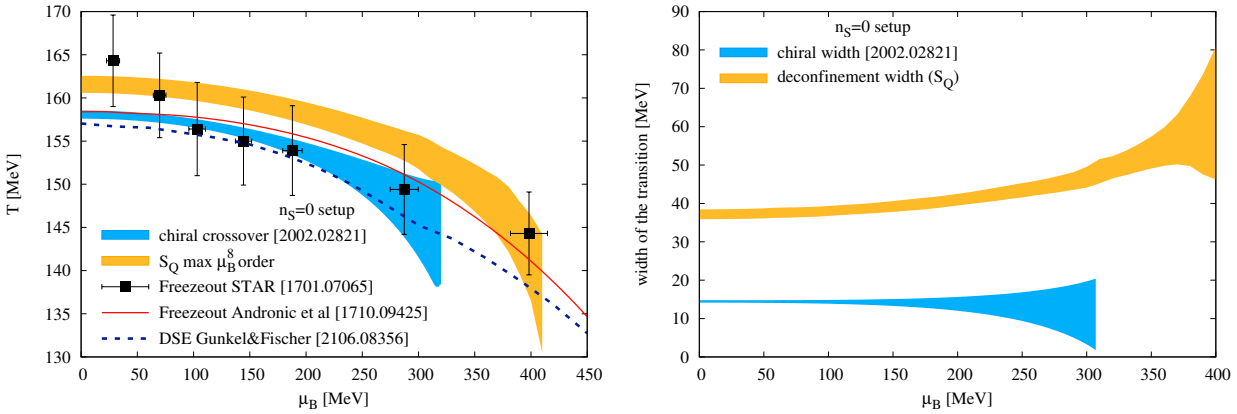


Fig. 7. Left: the deconfinement line (orange) defined as the peak position of the static quark entropy (S_Q) at fixed μ_B as a function of μ_B. We add the chiral transition line from lattice QCD [9] in the phase diagram, defined from the peak of the full chiral susceptibility, as well as the corresponding result based on Dyson-Schwinger equations [24] (the latter used the $\mu_S = \mu_B/3$ scheme as an approximation to $n_S = 0$). We also include the chemical freeze-out parameters [25, 10]. Right: the width associated to the deconfinement (orange) and chiral (blue) transitions: the two differ already at $\mu_B = 0$. With increasing μ_B, while the chiral width is mostly constant, the deconfinement width grows.

deconfinement transition gets smaller again. This is what one expects if a critical endpoint exists somewhere above $\mu_B = 400$MeV.

Finally, we address the robustness of the present computation. We use the eighth order of the Taylor series, which may seem a very arbitrary decision. In fact, the order of the expansion was only limited by the statistics. Up to 300 MeV the effect of the eighth order is invisible, and only gets statistically significant above $\mu_B \simeq 350$ MeV. For the $\mu_S = 0$ scheme the onset of the eighth order appears 50 MeV earlier. It is safe to claim that up to $\mu_B \leq 300$ MeV the systematic errors from the truncated extrapolation can be completely neglected, and we give a fair description of finite volume QCD at finite μ_B. At the same time, the eighth order is the lowest possible truncation in the range 300-400 MeV. Lower orders cannot give reliable results. The difference in the width between the $n_S = 0$ and $\mu_S = 0$ schemes is an effect that is visible only if the eighth order is included.

6. Outlook

The computational challenge in this work was twofold. First, it required a statistics of 1-2 million gauge configurations per temperature. The second half of the computer time was used for the measurements of the $A_j, B_j, C_j, \ldots$ coefficients. For this we used the reduced matrix formalism [26] that translates the problem of the μ_B dependence of the fermion determinant into an eigenvalue problem. We used the MAGMA library [27] to cope with the dense linear algebra problems on LUMI's accellerators.

While this was computation was very costly, having these eigenvalues we can, compute many other observables using the same configurations on the $16^3 \times 8$ lattice without much effort. Examples for future use of the data include:

i) We can compute the canonical determinants and rewrite the grand canonical ensemble as a canonical. This can give us access to the physics of small systems, relevant e.g. for proton-lead events.
ii) We can compute high order baryon fluctuations and use these to extrapolate the baryon kurtosis-to-variance ratio. This observable was measured by the STAR experiment for protons and is key in the search for the critical end point in the QCD phase diagram.
iii) Modelling the QCD grand canonical partition sum one can estimate the position of Lee-Yang zeros. These approach real values in the vicinity of the critical end-point with universal critical scaling.
iv) We can extend the range of fluctuation-based observables to strangeness. The statistical errors on strangeness fluctuations are significantly smaller, and the extrapolation up to 500 MeV is feasible with our statistics.

The data analysis for all these projects are currently underway and will deliver results in the second project year.

Acknowledgements

The project was supported by the BMBF Grant No. 05P21PXFCA. This work is also supported by the MKW NRW under the funding code NW21-024-A. Further funding was received from the DFG under the Project No. 496127839. This work was also supported by the Hungarian National Research, Development and Innovation Office, NKFIH Grant No. KKP126769. This work was also supported by the NKFIH excellence grant TKP2021_NKTA_64. This work is also supported by the Hungarian National Research, Development and Innovation Office under Project No. FK 147164. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer Juwels-Booster at Juelich Supercomputer Centre. We acknowledge the EuroHPC Joint Undertaking for awarding this project access to the EuroHPC supercomputer LUMI, hosted by CSC (Finland) and the LUMI consortium through a EuroHPC Extreme Access call, EHPC-EXT-2023E02-067.

References

[1] S. Pratt, E. Sangaline, P. Sorensen, H. Wang, Constraining the Eq. of State of Super-Hadronic Matter from Heavy-Ion Collisions, Phys. Rev. Lett. 114 (2015) 202301. arXiv:1501.04042, doi:10.1103/PhysRevLett.114.202301.

[2] L.-G. Pang, K. Zhou, N. Su, H. Petersen, H. Stöcker, X.-N. Wang, An equation-of-state-meter of quantum chromodynamics transition from deep learning, Nature Commun. 9 (1) (2018) 210. arXiv:1612.04262, doi:10.1038/s41467-017-02726-3.

[3] M. Abdallah, et al., Measurement of the Sixth-Order Cumulant of Net-Proton Multiplicity Distributions in Au+Au Collisions at $\sqrt{s_{NN}} = 27$, 54.4, and 200 GeV at RHIC, Phys. Rev. Lett. 127 (26) (2021) 262301. arXiv:2105.14698, doi:10.1103/PhysRevLett.127.262301.

[4] Y. Aoki, G. Endrodi, Z. Fodor, S. Katz, K. Szabo, The Order of the quantum chromodynamics transition predicted by the standard model of particle physics, Nature 443 (2006) 675–678. arXiv:hep-lat/0611014, doi:10.1038/nature05120.

[5] S. Borsanyi, et al., Is there still any T_c mystery in lattice QCD? Results with physical masses in the continuum limit III, JHEP 1009 (2010) 073. arXiv:1005.3508, doi:10.1007/JHEP09(2010)073.

[6] A. Bazavov, T. Bhattacharya, M. Cheng, C. DeTar, H. Ding, et al., The chiral and deconfinement aspects of the QCD transition, Phys.Rev. D85 (2012) 054503. arXiv:1111.1710, doi:10.1103/PhysRevD.85.054503.

[7] S. Borsanyi, Z. Fodor, M. Giordano, S. D. Katz, D. Nogradi, A. Pasztor, C. H. Wong, Lattice simulations of the QCD chiral transition at real baryon density, Phys. Rev. D 105 (5) (2022) L051506. arXiv:2108.09213, doi:10.1103/PhysRevD.105.L051506.

[8] P. Braun-Munzinger, J. Wambach, The Phase Diagram of Strongly-Interacting Matter, Rev. Mod. Phys. 81 (2009) 1031–1050. arXiv:0801.4256, doi:10.1103/RevModPhys.81.1031.

[9] S. Borsanyi, Z. Fodor, J. N. Guenther, R. Kara, S. D. Katz, P. Parotto, A. Pasztor, C. Ratti, K. K. Szabo, The QCD crossover at finite chemical potential from lattice simulations, Phys. Rev. Lett. 125 (2020) 052001. arXiv:2002.02821.

[10] A. Andronic, P. Braun-Munzinger, K. Redlich, J. Stachel, Decoding the phase structure of QCD via particle production at high energy, Nature 561 (7723) (2018) 321–330. arXiv:1710.09425, doi:10.1038/s41586-018-0491-6.

[11] M. D'Elia, M.-P. Lombardo, Finite density QCD via imaginary chemical potential, Phys. Rev. D67 (2003) 014505. arXiv:hep-lat/0209146, doi:10.1103/PhysRevD.67.014505.

[12] P. de Forcrand, O. Philipsen, The QCD phase diagram for small densities from imaginary chemical potential, Nucl.Phys. B642 (2002) 290–306. arXiv:hep-lat/0205016, doi:10.1016/S0550-3213(02)00626-0.

[13] C. Bonati, M. D'Elia, M. Mariti, M. Mesiti, F. Negro, F. Sanfilippo, Curvature of the chiral pseudocritical line in QCD: Continuum extrapolated results, Phys. Rev. D92 (5) (2015) 054503. arXiv:1507.03571, doi:10.1103/PhysRevD.92.054503.

[14] M. Stephanov, Non-Gaussian fluctuations near the QCD critical point, Phys.Rev.Lett. 102 (2009) 032301. arXiv:0809.3450, doi:10.1103/PhysRevLett.102.032301.

[15] S. collaboration, Precision Measurement of (Net-)proton Number Fluctuations in Au+Au Collisions at RHIC (4 2025). arXiv:2504.00817.

[16] P. Isserstedt, M. Buballa, C. S. Fischer, P. J. Gunkel, Baryon number fluctuations in the QCD phase diagram from Dyson-Schwinger equations, Phys. Rev. D100 (7) (2019) 074011. arXiv:1906.11644, doi:10.1103/PhysRevD.100.074011.

[17] Y. Lu, F. Gao, Y.-x. Liu, J. M. Pawlowski, Finite density signatures of confining and chiral dynamics in QCD thermodynamics and fluctuations of conserved charges (4 2025). arXiv:2504.05099.

[18] D. Bollweg, D. A. Clarke, J. Goswami, O. Kaczmarek, F. Karsch, S. Mukherjee, P. Petreczky, C. Schmidt, S. Sharma, Equation of state and speed of sound of (2+1)-flavor QCD in strangeness-neutral matter at nonvanishing net baryon-number density, Phys. Rev. D 108 (1) (2023) 014510. arXiv:2212.09043, doi:10.1103/PhysRevD.108.014510.

[19] S. Borsanyi, Z. Fodor, J. N. Guenther, S. D. Katz, P. Parotto, A. Pasztor, D. Pesznyak, K. K. Szabo, C. H. Wong, Continuum-extrapolated high-order baryon fluctuations, Phys. Rev. D 110 (1) (2024) L011501. arXiv:2312.07528, doi:10.1103/PhysRevD.110.L011501.

[20] S. Borsányi, Z. Fodor, J. N. Guenther, R. Kara, P. Parotto, A. Pásztor, L. Pirelli, C. H. Wong, Chiral versus deconfinement properties of the QCD crossover: Differences in the volume and chemical potential dependence from the lattice, Phys. Rev. D 111 (1) (2025) 014506. doi:10.1103/PhysRevD.111.014506.

[21] C. Allton, M. Doring, S. Ejiri, S. Hands, O. Kaczmarek, et al., Thermodynamics of two flavor QCD to sixth order in quark chemical potential, Phys.Rev. D71 (2005) 054508. arXiv:hep-lat/0501030, doi:10.1103/PhysRevD.71.054508.

[22] S. Borsanyi, Z. Fodor, J. N. Guenther, P. Parotto, A. Pasztor, L. Pirelli, K. K. Szabo, C. H. Wong, QCD deconfinement transition line up to μB=400 MeV from finite volume lattice simulations, Phys. Rev. D 110 (11) (2024) 114507. arXiv:2410.06216, doi:10.1103/PhysRevD.110.114507.

[23] C. Bonati, M. D'Elia, M. Mariti, M. Mesiti, F. Negro, F. Sanfilippo, Roberge-Weiss endpoint at the physical point of $N_f = 2 + 1$ QCD, Phys. Rev. D93 (7) (2016) 074504. arXiv:1602.01426, doi:10.1103/PhysRevD.93.074504.

[24] P. J. Gunkel, C. S. Fischer, Locating the critical endpoint of QCD: Mesonic backcoupling effects, Phys. Rev. D 104 (5) (2021) 054022. arXiv:2106.08356, doi:10.1103/PhysRevD.104.054022.

[25] L. Adamczyk, et al., Bulk Properties of the Medium Produced in Relativistic Heavy-Ion Collisions from the Beam Energy Scan Program, Phys. Rev. C 96 (4) (2017) 044904. arXiv:1701.07065, doi:10.1103/PhysRevC.96.044904.

[26] A. Hasenfratz, D. Toussaint, Canonical ensembles and nonzero density quantum chromodynamics, Nucl. Phys. B371 (1992) 539–549. doi:10.1016/0550-3213(92)90247-9.

[27] C. Brown, A. Abdelfattah, S. Tomov, J. J. Dongarra, Design, optimization, and benchmarking of dense linear algebra algorithms on AMD gpus, in: 2020 IEEE High Performance Extreme Computing Conference, HPEC 2020, Waltham, MA, USA, September 22-24, 2020, IEEE, 2020, pp. 1–7. doi:10.1109/HPEC43674.2020.9286214.
URL https://doi.org/10.1109/HPEC43674.2020.9286214

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 14–39

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Benchmarking lattice quantum field theory codes across EuroHPC platforms for physics beyond the standard model on physical systems

Frédéric D.R. Bonnet[a,*], Ryan Hill[b], Ed Bennett[a]

[a]School of Mathematical and Computer Science, Swansea University Bay Campus, Fabian Way, Swansea, SA1 8EN, Wales, UK.
[b]School of Physics and Astronomy, University of Edinburgh, James Clerk Maxwell Building, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, UK.

Abstract

The aim of this study is to benchmark codes used in lattice quantum field theory for physics beyond the Standard Model (SM) on various clusters within the `EuroHPC` infrastructures. We have used our existing benchmarking suite called `SOMBRERO`, which is derived from `HiRep`, and `BKeeper` which is derived from Grid, and created an automated workflow framework from scratch called `Bench_Grid_HiRep`. We find that the way the global problem is partitioned between MPI ranks used can have a significant impact on the code performance and run time.

Keywords: Benchmarking; lattice quantum field theory; particle physics; physics beyond the standard model;

1. Introduction

The Standard Model (SM) of particle interactions is a synthesis of three of the four forces of nature and is probably one of the greatest achievements in theoretical physics so far. These forces are described by gauge theories, each of which is characterized by a coupling constant, g. For the strong interaction, $g_s \sim 1$. For the electromagnetic interaction, the fine structure constant is $g_{\rm em} \sim 1/137$. This constant is known to a very high precision, the predictions of Quantum Electrodynamics (QED) agree with nature to extremely high accuracy making it the most successful theory in physics. The strength of the coupling between particles is described by g, and its size is also the reason why the perturbative approach has been very successful for QED. For weak interactions, the Fermi constant [1] (which has been determined experimentally with great accuracy), $G_{\rm F} \sim 10^{-5}$ GeV^{-2}, is much less than 1, leading to another weakly-coupled theory.

* Corresponding author.
E-mail address: frederic.bonnet@swansea.ac.uk

1877-0509
10.1016/j.procs.2025.08.230

It is now almost universally accepted that Quantum Chromodynamics (QCD) [2] is the underlying quantum field theory [3] of the strong interaction, which binds atomic nuclei and fuels the sun and the stars. The bound states of QCD are referred to as hadrons, and the study of these will provide answers on the mechanisms that bind matter.

Experimental work has confirmed that the basic constituents of matter are quarks, which are confined inside the hadrons by the gluons. Over the past few years, there have been major upgrades in the energy scale that particles accelerators can access. As a result, we have growing opportunities to extend our understanding beyond the Standard Model and open the door for unknown physics at both the subatomic and astrophysical scales. Moreover, the onset of the Exascale computing era puts particle physicists in a position to be able to compute observables that were simply unattainable in a reasonable amount of time just a few years ago.

In the evolution, as far as we know, of the (or a) universe as we go back in time, $t \to 0$, near the Planck scale and beyond for that matter a lot the physics is not fully understood or known. The geometry at that scale is extremely different to what we are commonly used to in our solar system. It is strongly believed that dark matter plays an essential role. Furthermore, quantum gravity is yet to be fully understood and formulated at the theoretical level.

The Lattice Gauge Theory approach is one of the very few first-principles methods for studying QCD in the nonperturbative low-energy regime. The most widely used Lattice Gauge Theory formulation is the so-called Lagrangian-based lattice field theory, which discretises the field theory on a Euclidean space-time lattice [4,5]. An alternative lattice approach is based on the Hamiltonian formulation of quantum field theory and makes use of cluster decompositions and again Monte Carlo methods to carry out the simulations [6]. In addition, there are numerous studies based on a light-front formulation of QCD [7] and much use has been made of Schwinger-Dyson equations [8] to assist with the construction of QCD-based quark models. The Lagrangian-based lattice technique simulates the functional integral using a four-dimensional hypercubic Euclidean spacetime lattice together with Monte Carlo methods for generating an ensemble of gauge field configurations with the appropriate Boltzmann distribution $exp(-S_G)$. The argument of the exponential, S_G is a discretized form of the gauge action on the hypercubic lattice. The simplest discretizations of the QCD and other actions involve only nearest neighbours on the lattice and have $O(a^2)$ errors, where a is the lattice spacing.

Lattice QCD, and more generally Lattice Gauge Theory, is based on a Monte Carlo treatment of the path integral formulation which makes it a computationally demanding method for calculating physical observables. In QCD, for example, the gluon field is represented by 3×3 complex $SU(3)$ matrices, where there is one such $SU(3)$ matrix associated with every link on the lattice. The links lie only along one of the four Cartesian directions and join neighbouring lattice sites. Since all lattice links require identical numerical calculations, lattice gauge theory is ideally suited for parallel computers. Quark (pseudofermion) fields, meanwhile, are three–element complex vectors, transforming under the gauge field by standard matrix multiplication. Beyond the standard model applications involve gauge groups other than $SU(3)$ and fermions in representations other than the fundamental. In the latter case, rather than N–element vectors, pseudofermions are $M > N$–element vectors, transforming by matrix multiplication with a projection of the gauge field into a higher representation of the gauge group [22-24]. Popular extensions of the standard model are built with gauge theories based on $SU(N)$ or $Sp(2N)$ groups, and fermions in one or more of the fundamental, antisymmetric, symmetric, and adjoint representations. An example is the Higgs compositeness framework, based on the pseudo Nambu-Goldstone mechanism. A specific realization of this mechanism with rich phenomenology is provided by an $Sp(2N)$ gauge theory with two fermion flavours transforming in the fundamental representation. When this theory is augmented with three antisymmetric fermion flavours, the model can account for the high mass of the top quark with respect to the QCD scale through a mechanism known as *top partial compositeness*. These concepts are reviewed, for instance, in [21], which discusses a realization of Higgs compositeness with top partial compositeness built on the $Sp(4)$ gauge theory with two fermions in the fundamental and three fermions in the antisymmetric representation.

This project is part of the field of high-energy physics. It strongly relies on supercomputing resources to enable the extraction of observable quantities from the numerical simulation of theories of strong interactions beyond the standard model performed in the framework of Lattice Gauge Theories. It contributes to improving and extending our current understanding of the physics of elementary interaction and of the early universe addressing questions such as the nature of dark matter, the dynamical origin of electroweak symmetry breaking and the existence of a relic background of gravitational waves. Using state of the art numerical techniques and codes that are developed in-house we aim at extending and improving calculation of the $N_f = 2$ meson spectrum in $Sp(4)$ using Domain Wall (Möbius) fermions at small masses is an example of the transition to precision results, as is the physical characterization of

the confinement transition in the $Sp(4)$ Yang-Mills theory at finite temperature. The study of scattering phase shift of Pseudo Nambu-Goldstone Bosons, the singlet antisymmetric spectrum in $Sp(4)$ gauge theory or the study of the beta function in the $SU(2)$ gauge theory with adjoint matter, are other examples of the scientific investigations performed within our program. Our studies are relevant in the context of the experimental work carried out at the LHC or at gravitational wave observatories like the upcoming Einstein Telescope.

We make use of two open source code bases, `HiRep` [1] and `Grid` [2], to which our group has contributed support for $Sp(2N)$ groups. The benchmarking is essential in order to optimize codes on the various systems that we intend to work on within the `EuroHPC` infrastructure; moreover, leverage the output of current investment in HPC and benefit the community of supercomputer users, with a particular attention to the current and future user base of multi-purpose software libraries that we are using. Due to the implied very high computational demands and the necessity to work in the communication-dominated regime as well as in the computation-dominated regime and in more balanced scenarios, our code lends itself to adaptations as a general-purpose and health-monitoring benchmark suite. To facilitate usage, we have extracted from our codes benchmarks that are widely used in academia (e.g., on the DiRAC HPC service in the UK) and industry (industrial users include Intel and NVIDIA). The benchmarking may also be done using other Lattice QCD codes as a comparison test.

In this study we present the results that we obtained during our time allocation on the `EuroHPC` (call number `EHPC-BEN-2024B10-015`) machines which include results from the following computing centers `Vega` located in Slovenia, `Lumi` in Finland and `Leonardo` at Cineca–HPC in Italy.

2. The various codes

In this section, we will briefly describe the key features of the codes we intend to use in this study. The codes gives the possibility to study many different physical systems in Lattice Gauge Theory. We will omit many details of the inner workings of the codes used in this study; however, we lead the interested reader to the references provided in this document and therein.

2.1. Lattice software frameworks

`HiRep` [9,10,11] and `Grid`[3] [12,13] are software tool kits to perform computations in lattice quantum field theory.

`HiRep` is a collection of lattice software tools designed to study physics beyond the Standard Model (BSM). It is written in `C99`, with a C++ and Perl-based code generator to produce a set of bespoke linear algebra macros specifically targeting the matrix size of interest. It makes use of MPI for parallelism. Recent work, not all of which is merged into the main trunk of `HiRep`, has included implementation of symplectic gauge groups [14], $Sp(2N,F)$, implementation of the Logarithmic Linear Relaxation (LLR) algorithm [15], which provides an alternative to typical importance sampling-based computations, and a highly-performant CUDA and HIP implementation [11].

`Grid` is a performance portability framework for large multidimensional containers, targeting both CPUs with in principle arbitrary length vector units, and GPUs. It has been developed with applications to lattice quantum field theory (and in particular Lattice Quantum Chromodynamics (LQCD)) in mind, and is particularly feature-complete in this direction. It is written in `C++17`, with GPU implementations in CUDA, HIP, and SYCL, and supports both OpenMP for multithreading and MPI for distributed memory parallelism. Similarly to `HiRep`, `Grid` has also recently been extended with an implementation of symplectic gauge groups [16].

The two codes provide complementary implementations of the desired physics. `Grid` offers a wider range of features that are designed to be composed together allowing, for example, a variety of fermion formulations. `HiRep` is more targeted in its feature set, but its smaller scope can make it easier to build and test new functionality on top of.

[1] The main branch of `HiRep` may be found via the following link: https://github.com/claudiopica/HiRep. The code `SOMBRERO`, used in this study, is based on the following branch https://github.com/sa2c/HiRep.

[2] https://github.com/UCL-ARC/Grid.git is a branch that we have used for specific benchmarking test. The main and current branch is available via https://github.com/paboyle/Grid.

[3] https://arxiv.org/abs/1512.03487

2.2. Benchmarks

2.2.1. BKeeper

BKeeper [4] [20] was developed to provide a benchmark comparable to SOMBRERO for the Grid code. BKeeper itself is compiled to a binary executable that is statically linked against the Grid library. BKeeper leverages the flexibility afforded by the deep templating of the Grid code-base to be easily extended to the full range of gauge groups and fermion representations supported by Grid, and can be compiled with support for arbitrary subsets of the supported theories to mitigate the lengthy compile times required for each fermion implementation. It is driven via an XML input file, which governs the scenarios that we wish to benchmark.

2.2.2. Sombrero

SOMBRERO [5] [17] is a mini-app benchmark derived from HiRep. Based on a CPU-only branch of HiRep supporting symplectic groups, the benchmark tests the performance of the conjugate gradient inversion of the Dirac operator, which typically accounts for 90% or more of the execution time of physics problems. To minimize the size of the code package and simplify porting, the SOMBRERO code is extracted from the HiRep source in a semi-automated way, making use of the pycallgraph library [18].

Benchmarking the Dirac operator inversion, rather than the Dirac operator itself, allows a better picture of how the Dirac operator kernel performs as part of the full computational workflow. Benchmarks are performed for six theories:

- case-1: $SU(2)$ with adjoint matter
- case-2: $SU(2)$ with fundamental matter
- case-3: $SU(3)$ with fundamental matter (QCD)
- case-4: $Sp(4)$ with fundamental matter
- case-5: $SU(3)$ with two-index symmetric matter
- case-6: $Sp(4)$ with adjoint matter

We expect [19] that theories with a larger gauge group (i.e. larger number in brackets) increase the demand on computing resources more strongly than those on communications, while a larger fermion representation (fundamental < two-index symmetric < adjoint) increases the communication demands more than the computation. As such, we would expect to see the best strong scaling for $Sp(4)$ with fundamental matter, and the worst for $SU(2)$ with adjoint matter.

2.3. Bench_Grid_HiRep framework

Bench_Grid_HiRep is a framework used to deploy over a set of systems or a HPC cluster. The code provides an automated way of deploying Grid and HiRep (Sec 2.3), SOMBRERO (Sec 2.2.2) and BKeeper (Sec 2.2.1) and their library dependencies which are GNU-MPFR[6], the US Lattice QCD[7] software suite, with its C-Lime library and the GNU-GMP[8] libraries. Once the codes are deployed onto a given system(s), the code(s) are compiled and linked together. A set of benchmarks runs are created as a first step during the process. Once all of the dependencies are compiled, configured properly and successfully, these just created benchmark runs are then launched on the fly and in the continuity of the process. Subsequent runs can then be launched without having to recompile everything, as it is a modular workflow. Provided that one has information on how to distribute the computation over the multi–GPUs on the node in BKeeper before starting the process and knows the correct modules to load for an error-free compilation, then the entire process can be deployed from a local host and remotely using the SSH and SCP protocols, or directly on

[4] https://github.com/RChrHill/BKeeper
[5] https://github.com/sa2c/sombrero
[6] The MPFR library is a C library for multiple-precision floating-point computations, https://www.mpfr.org
[7] http://usqcd-software.github.io
[8] GMP is a library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. https://gmplib.org

the cluster if there is Internet access on that cluster. The framework is modular; in short, it means that it is possible to adjust, extend, modify certain sections without affecting the others, provided that one uses the same variables declared in the `common_main` block. The idea of porting, case generation, compilation/building, case gathering, and then case launching on the cluster are then separated and disjoint but yet connected in a workflow system of sequential calls. This gives the possibility to start/restart incrementally without having to go through the entire workflow from the start. Once the runs have completed, an analysis code can be used to generate a data visualization of the results. The code is written as a combination of Linux Bash and Python-3.12 and is available on GitHub [9]. The code has been tested on Windows via `WSL` and on Linux systems, so far. The overall architecture of the code is shown in Figure A.1,

3. Benchmarking results

In this section, we present the results obtained on the different systems using the codes mentioned in Section 2. The runs were performed on the various machines to which we have gained access through the `EuroHPC` call (project call number `EHPC-BEN-2024B10-015`). These calls extend for a period of three months and provide full access to the machine partitions. Our limited allocation consisted of 400 node hours on `Vega-GPU`, 2000 and 2500 nodes hours for the `Lumi-C` and `Lumi-G` respectively, and similarly on `Leonardo` with 2000 and 3500 node hours on the two partitions `Leonardo-DCGP` and `Leonardo-Booster` respectively. Figures are available in Appendix A. A Zenodo [10] repository has been setup for the figures, output data and codes. These can be found using the links `10.5281/zenodo.15782266` or `https://doi.org/10.5281/zenodo.15782266`.

3.1. *BKeeper results*

As a first step in the benchmark procedure, using `BKeeper`, we screened a set of MPI combinations over the nodes with a given number of GPU available on that node over space–time, which can be thought of as a 4–vector $x = (\tilde{x}, t)$. For example, if a node has 4 GPUs, ($n_{\rm gpu} = 4$), then the resulting set for the possible MPI combination (partitioning between MPI ranks used at run time) are based on the following expression:

$$\begin{aligned} M \quad &= \quad \left\{ \Gamma = \prod_{i,j,k,j=0}^{n_{\rm gpu}} x_i y_j z_k t_l \;\mid\; \Gamma = n_{\rm gpu}, \quad \text{and} \quad i,j,k,l = 1,..,n_{\rm gpu} \in \mathbb{Z}^+ \right\} \\ \longrightarrow \quad & M = \left\{ x_i, y_j, z_k, t_l \right\}_{i,j,k,l=1,..,n_{\rm gpu}} = \left[1, n_{\rm gpu} \right] \in \mathbb{Z}^+. \end{aligned} \tag{1}$$

The number of combinations with repetition, will then be 10 distinct possibilities. Using this idea, we ran the cases with an increasing number of nodes ($n_{\rm nodes}$), from 1 to 2 for the `Small` set, and for $n_{\rm nodes} = \{4, 8, 12, 16, 24, 32\}$ for the large set. The `Small` set was constructed using a lattice of $24^3 \times 32$. For the `Large` set, we used two larger lattices $32^3 \times 64$ and $64^3 \times 96$. The lattice spacing for the conjugate gradient was set at its default size of $a = 0.1(fm)$ in the case `BKeeper`. These lattice dimensions are realistic and common in large simulation in lattice gauge theories. Using `BKeeper` we were able to extract results for three representations, the $SU(2)$–Adjoint, $SU(2)$–Fundamental and $SU(3)$–Fundamental. Additional runs are underway to add higher representations.

For the `Small` set, we compared the results between runs on 1 and 2 nodes. `Vega-GPU` and `Leonardo-Booster` have the same number of GPUs per nodes, i.e. 4, these are therefore comparable. On `Lumi-G` there are 8 GPUs per node, which means that we should expect a factor of two when compared with the other two machines i.e. `Vega` and `Leonardo-Booster`. However, this may not necessarily be the case as `Lumi-G` and the other two systems have different GPUs in it, with AMD and NVidia-A100 in the other respectively. In Fig A.6, we show the results for the runs on the `Small` set for lattice $24^3 \times 32$ on 1 node and 2 nodes for a set of MPI distribution versus conjugate gradient run time in seconds (`CG Run Time (s)`) obtained on `Vega-GPU` and `Leonardo-Booster`. In Fig A.9, for the runs obtained `Lumi-G`. Since `Lumi-G` has 8 GPUs per node, the number of MPI possibilities is larger, as the different combinations can now go to 8 and not 4, as was the case on the other two systems just mentioned. From

[9] `https://github.com/fbonnet08/Bench_Grid_HiRep`
[10] Zenodo repository are available via `https://zenodo.org`

these Figures A.6 and A.9, we can clearly notice that in all cases and for all the different representations we observe significant variations on the run time per second when we vary the MPI distribution. Furthermore, we are able to identify the MPI combination that gives the best performance on that nodes, i.e. when we use `node001` and `node002`. This result is consolidated if we plot the same `CG Run Time (s)` versus the number of nodes for each run and each representation. The results are shown in Fig. A.14 for `Vega-GPU` and `Leonardo-Booster` and in Fig. A.17 on `Lumi-G`.

The graphs shown in Figure A.14 and in Fig. A.17 are particularly informative, because they show multiple aspects of the scaling. The first aspect is the scaling on each node (the graph is being read from bottom to top) and the second one being the scaling as we increase the number of nodes (the graph is then read from left to right).

Looking at the first one, we can see how each representation are clustered together and we are also able to identify the MPI distribution that gives the best run time in seconds, this is achieved by taking the lowest value on the vertical line for each node number. There we have the three representations shown there and we can observe that the $SU(3)$–Fundamental is the most time-consuming as expected since it is the biggest gauge group among the set because the matrices have 9 elements instead of 4 as it is the case in the $SU(2)$ and this is for every lattice sites. In an $Sp(4)$ theory, there will then be 16 elements in the gauge field matrices, hence we expect the run time to be considerably slower than the $SU(3)$ theory. The second relevant aspect is as we go from 1 node to 2 (reading the graph from left to right in this case) and look at the slope of the lines joining the two, which gives a clear indication of the cost of communication between the nodes. The line is the average of the data and the band is an 80% band, the values that fall outside these bands are values that are more or less than 80% of the value of the average. In almost all cases we see that the slopes are negative for the run time (`CG Run Time (s)`) showing a speed up going from one node to two on `Vega-GPU` and a coherent positive slop when we look at the number of flops per seconds (`Flops/s (GFlop/s)`). This appear to be true on `Vega-GPU` and `Leonardo-Booster` but not on `Lumi-G` when we consider the `case-1:` $SU(2)$ `adj`. This is kind of expected as there very little amount of computing to be done hence we spend more time fetching out data from memory than actually computing and also because on two nodes on `Lumi-G` we would be utilizing not 8 GPUs but 16, which is a large amount considering the size of the problem at hand. On much larger lattices this may not necessarily be the case. We also repeated the experiment for large lattice size over a larger number of nodes on a 4-GPU per node system (`Vega-GPU`), Fig A.22, and on a 8–GPU per node system such as `Lumi-G`, Fig A.27.

The results reinforces the ones previously stated and enables us to really see the scaling between the number of nodes on all systems. On `Vega-GPU` we were not able to extract the equivalent number of nodes as we did on `Lumi-G`, and we ran out of access time on `Leonardo-Booster`. However, based on the runs that we were able to perform on `Vega-GPU` and `Lumi-G` on lattices $32^3 \times 64$ and $64^3 \times 96$ we can clearly see an increase in the slope as we increase the number of nodes, and this is on all systems. On `Lumi-G` we are seeing a speed up that exceeds linear until we get to some kind of plateau near `node032`. Again, this is due to the size of the problem. It is therefore possible to anticipate that the speed-up trend will continue as we increase the problem size to very large lattices. Based on Figure A.22, there we are also seeing speed-ups that are exceeding linear speed-up as we increase the number of nodes on large lattices.

3.2. *SOMBRERO results*

We are now going to examine the results obtained from `SOMBRERO`. This application was used to benchmark `HiRep` indirectly and obtain equivalent results as if we were using that code directly. `SOMBRERO` enables us to perform a comparative study of several scenarios, with different degree of computational intensity. Moreover, `SOMBRERO` offers the possibility of generating six distinct cases, which can each be mapped to a `BKeeper` run case. The correspondence is defined in Section 2.2.2. `SOMBRERO` performs all computations on the CPU, hence we have used the following CPU partitions on the `EuroHPC` systems: `Leonardo-DCGP`, `Lumi-C` and `Vega-CPU`.

Since `SOMBRERO` is a pure CPU based code, it hence offers a much bigger variability in the number cases that can be tested, in particular, the many different lattice sizes (in the `Weak` case), number of tasks per node and the number of total cores for each run, for example. We tested for each run, by varying the number of nodes as $n_{\text{nodes}} = \{1, 2, 4\}$, as well as the number of tasks per node, which depends on the available number of cores on that node. In general, we want to use all the cores available on that node. However, in this case, we have varied it as well to see how the code communicates on the node and between the CPU sockets. The following question then follows; is it better to focus half of the total number of threads on that node or split that half into 2 chunk cores to target each socket on that node and totaling

the number of cores on that node? To address this question, we have incremented the parameter number of tasks per node from one to half the total number of cores on that node. For example, on Leonardo-DCGP the maximum number of cores on the node is 192, hence we have varied the number of tasks per node as $n_{\text{ntpns}} = \{1, 2, 3, 6, 12, 24, 48, 96\}$. Not all of the runs have completed successfully. Our runs are constructed in several groups. For each system, we ran a Small and Large set over the number of nodes. Those just mentioned, in addition to the number of tasks per node (n_{ntpns}). Both sets were run in the Weak and Strong settings. The Weak case offers the possibility to scan over many different lattice sizes as opposed to the Strong case, which focuses on larger lattices. SOMBRERO then produces the output in terms of GFlops per seconds (s) for each case and each lattice in all representations (the six just mentioned above). The total number of cores on all systems does vary and is not the same.

3.2.1. Weak

In figs. A.34, A.41 we show the results obtained on Vega-CPU for mpi_distribution vs GFlops per seconds (s) and nodes vs GFlops per seconds (s) respectively. In Figs. A.48, A.55 for Lumi-C, and in A.62, A.69 for Leonardo-DCGP in the Weak case. The plots combine both the Small, and the Large sets. The Large is generally located on the right-hand side cluster of points of the graph. This allows a direct side-by-side comparison between the two sets.

On each set of graphs the six different representations are depicted and allow a direct comparison between each representation, and the performances differences obtained on each system. It is important to mention and keep in mind that we have also varied the n_{ntpns} variable on each node for all systems. In the Weak case, there are many lattice sizes being tested on different representations, some of which are as small as $4^3 \times 4$ and the largest being $48^3 \times 1024$, where 1024 represents the number of lattice sites in the time direction. It is also important to note that, in the Weak case we also vary the lattice sizes in a permutative variation pattern with repetitions. For example, if we take the small lattices, the set of variations may take the form of $\{4^3 \times 8, ..., 8^3 \times 8\}$, and for the large lattices as $\{48^3 \times 8, ..., 48^3 \times 1024\}$. This offers a larger screening and the possibility to see scaling in all directions. In $Sp(2N)$ theories, when we are computing the density of states using the LLR method mentioned earlier, one of the major computational cost comes from computing all of the replicas, and from the fact that we need to extend in the time direction. As a result, we need to gain insight in this way. Also shown in the graphs, and for each point on the graphs, are the times taken in seconds. The number of nodes used (n_{nodes}) and the number of tasks per node (n_{ntpns}) are also shown. In short, in the Weak experiment, all the points on each graph correspond to a different lattice size, n_{nodes}, n_{ntpns} or mpi_distribution.

We may now be able to compare the performances on each systems in the Weak case in two possible ways, mpi_distribution vs GFlops per seconds (s) and nodes vs GFlops per seconds (s). For the mpi_distribution vs GFlops per seconds (s). This is shown in Fig. A.34 for Vega-CPU, in Fig. A.48 for Lumi-C, and, in Fig. A.62 for Leonardo-DCGP. As already mentioned, the Small results are located on the left cluster of points while the Large set are generally located in the cluster on the right hand side of the same graph. We notice that for adj representation the clusters of points are more blended and particularly pronounced in the case of the $Sp(4)$. The fundamental representations seem to provide the best GFlops per seconds (s), that is, for the cases of case-2: $SU(2)$ fun, case-3: $SU(3)$ fun and case-4: $Sp(4)$ fun. This appears to be true on all systems. Another relevant aspect that we can observe on the graphs, is the slope of the scaling in both the Small and Large. We observe that the Small sets are more likely to show plateau type behavior as opposed to the Large sets in particular on Vega-CPU and on Leonardo-DCGP as opposed to Lumi-C, where we see in both cases good performances. We recall, that the Small set is mostly composed of very small lattices.

If we look at when we plot same data runs but this time nodes vs GFlops per seconds (s) this is shown in Fig. A.41 for Vega-CPU, in Fig. A.55 for Lumi-C and in Fig. A.69 for Leonardo-DCGP. The total number of cores on all systems do vary and are not the same. In these plots, we can observe what happens when we increase the number of nodes. As before, the plots can be read from left to right for the number of increasing nodes and, from bottom to top, which corresponds to the variation of the number of tasks per node as n_{ntpns}. The things to look for are the slopes of the curves when we increase the number of nodes. In general we expect to observe an upward trend or a positive slope, however, it also depends on n_{ntpns}.

3.2.2. Strong

In Figures. A.76, A.83 we show the results obtained on Vega-CPU for mpi_distribution vs GFlops per seconds (s) and nodes vs GFlops per seconds (s) respectively. In Figs A.90, A.97 for Lumi-C, and in A.104,

A.111 for Leonardo-DCGP in the Strong case. To reduce the number of plots we have combined both the Small and Large cases as well. The strong case focuses on two different lattice sizes, for the Small, the lattice size is $\{24^3 \times 32\}$, while for the Large set the lattice size is significantly large $\{64^3 \times 96\}$. In Fig. A.76, Fig. A.90, and Fig. A.104 plots the mpi_distribution vs GFlops per seconds (s) for all the systems. Also shown on the plots, are the number of nodes, n_{nodes}, and n_{ntpns}, as was done for the Weak case. Again, as in the BKeeper plots we also provide the average band and the 80% band on the line plots for all sets (the variability being on the variable n_{ntpns}). In all cases, we see a sharp speed-up as we increase the size of the resources provided. It also appears that in all cases we obtain the maximum number of GFlops per seconds (s) when all of the available cores on the node are mobilized to the task. The mpi_distribution needs to be tuned accordingly. As in the Weak case the fundamental representations provide the highest GFlops per seconds (s). We then plot the same data but this time nodes vs GFlops per seconds (s). This is shown in Figs. A.83, Fig. A.111 and Fig. A.111. When we differentiate the umber of nodes used in the computation it is really possible to pinpoint the effects of varying n_{ntpns} with the associated mpi_distribution. The shaded bands are again is the average value with an 80% band on the data. Essentially the value located on the top of the of the vertical line for a given number of nodes with its associated mpi_distribution is the combination that provides the best performances for the representation in question. It is worth mentioning that on Leonardo-DCGP we obtain very large variability in the number of flops per second (GFlops per seconds (s)) compared with the other systems.

4. Conclusion and outlooks

In this study we have bench marked our benchmarking suite codes on the EuroHPC machines under the benchmark call EHPC-BEN-2024B10-015 for a small set of lattices. We have found that, the mpi_distribution produces significant variations on both the execution time and the number of flops, GFlops per seconds (s) for SOMBRERO and Flops/s (GFlop/s) for BKeeper.

4.1. Outlook

In the very near future, we intend to extend our benchmarking runs on physics simulations. In particular to include Domain Wall Fermions, extend the BKeeper to all represention to match those obtained with SOMBRERO. In this study, we have not benchmarked HiRep directly, but intend to do so as well. These runs will be performed in the EuroHPC, call EHPC-BEN-2025B03-046.

Acknowledgments

We acknowledge the support of EuroHPC in particular for the Benchmarking calls EHPC-BEN-2024B10-015 and EHPC-BEN-2025B03-046. We also would thank the support teams on all of the systems, Vega, Lumi and Leonardo for their support with some of the compilation and deployments issues that we have faced for each systems. This work has been funded by the UKRI Science and Technologies Facilities Council (STFC) Research Software Engineering Fellowship EP/V052489/1, by STFC under Consolidated Grant No. ST/X000648/1, by the ExaTEPP project EP/X017168/1, and by the project to form a Collaborative Computational Project in Theoretical and Experimental Particle Physics (CCP-TEPP).

Data Availability Statement

Data used to generate the plots provided in this document can be made available. All of the numerical results presented in this document have been obtained on the EuroHPC machines using the time allocation that was provided in call EHPC-BEN-2024B10-015. All data presented will be made available in advance of the EuroHPC User Day. There are two files available the main text and the Appendix A for all the figures (https://doi.org/10.5281/zenodo.15782266).

Software Availability Statement

All of the codes mentioned in this document are available via GitHub, this is the case for Grid, BKeeper, SOMBRERO, HiRep and Bench_Grid_HiRep. The codes are available via the links provided within this document. The codes follow licenses determine therein. Both SOMBRERO [18] and BKeeper [20] are openly available, licensed under the GNU General Public License version 2. The framework Bench_Grid_HiRep presented in Sec. 2.3 is openly available (https://github.com/fbonnet08/Bench_Grid_HiRep) under the GNU-GPLv3.

References

[1] Navas, S. and others, Particle Data Group, *Review of particle physics*, Phys. Rev. D, 110, 3, 030001, (2024).
[2] T. Muta, *Foundations of quantum chromodynamics: An introduction to perturbative methods in gauge theories*, World Scientific Lecture Notes In Physics, 5 (World Scientific, Singapore 1987).
[3] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory*, (Addison-Wesley, New York, 1995).
[4] H. J. Rothe, *Lattice Gauge Theories: An Introduction* (World Scientific, Singapore, 1992).
[5] I. Montvay and G. Munster, *Quantum Fields on a Lattice*, (Cambridge Univ. Press, Cambridge, UK, 1994).
[6] D. Schutte, W. Zheng, and C. J. Hamer, *Phys. Rev. D* **55**, 2974 (1997) [hep-lat/9603026].
[7] S. J. Brodsky, H.-C. Pauli, and S. S. Pinsky, *Phys. Rept.* **301**, 299 (1998) [hep-ph/9705477]; S. J. Brodsky, G. McCartor, H.-C. Pauli, and S. S. Pinsky, *Part. World* **3**, 109 (1993); S. Glazek, A. Harindranath, S. Pinsky, J. Shigemitsu, and Kenneth Wilson, *Phys. Rev. D* **47**, 1599 (1993).
[8] C. D. Roberts and A. G. Williams, "Dyson-Schwinger equations and their application to hadronic physics", *Prog. Part. Nucl. Phys.* **33**, 477 (1994) [hep-ph/9403224].
[9] L. Del Debbio, *et.al.*, *Meson spectroscopy from spectral densities in lattice gauge theories*, (2024), arXiv:2405.01388v1 [hep-lat].
[10] L. Del Debbio, A. Patella, and C. Pica, *Higher representations on the lattice: Numerical simulations. SU(2) with adjoint fermions*, Phys. Rev. D, 81, 094503, 2010, arXiv:0805.2058 [hep-lat].
[11] S. Martins, E. Kjellgren, E. Molinaro, C. Pica and A. Rago, *GPU-accelerated Higher Representations of Wilson Fermions with HiRep*, PoS, EuroPLEx2023, 035, (2024), arXiv:2405.19294 [hep-lat].
[12] P. Boyle, A. Yamaguchi, G. Cossu and A. Portelli, *Grid: A next generation data parallel C++ QCD library*, (2015), 1512.03487, arXiv:1512.03487 [hep-lat].
[13] A. Yamaguchi, P. Boyle, G. Cossu, G. Filaci, C. Lehner and A. Portelli *Grid: OneCode and FourAPIs*, PoS, LATTICE2021, 035, (2022), arXiv:2203.06777 [hep-lat].
[14] E. Bennett, D. Ki Hong, J. Lee, C. -J. D. Lin, B. Lucini, M. Piai, D. Vadacchino, *Sp(4) gauge theory on the lattice: towards SU(4)/Sp(4) composite Higgs (and beyond)*, JHEP, 03, 185, (2018),arXiv:1712.04220 [hep-lat].
[15] B. Lucini, D. Mason, M. Piai, E. Rinaldi, and D. Vadacchino, *First-order phase transitions in Yang-Mills theories and the density of state method*", Phys. Rev. D", 108, 7, 074517 (2023), arXiv:2305.07463 [hep-lat].
[16] N. Forzano, and others, *Lattice studies of Sp(2N) gauge theories using GRID*, PoS, LATTICE2023, 097 (2024), arXiv:2310.02111 [hep-lat].
[17] E. Bennett, M. Mesiti, *SOMBRERO*, https://github.com/sa2c/SOMBRERO.
[18] E. Bennett, *SOMBRERO: A high-performance parallel benchmark from computational particle physics*, (2019), https://rseconuk2019.sched.com/event/SMZU/p8-sombrero-a-high-performance-parallel-benchmark-from-computational-particle-physics.
[19] E. Bennett, B. Lucini, L. Del Debbio, K. Jordan, A. Patella, C. Pica, A. Rago, *BSMBench: A flexible and scalable HPC benchmark from beyond the standard model physics*, International Conference on High Performance Computing & Simulation, 834–839 (2016), arXiv:1401.3733 [cs.DC].
[20] R. Hill, *BKeeper*, https://github.com/RChrHill/BKeeper.
[21] E. Bennett, J. Holligan, D. Ki Hong, H. Hsiao, J. W. Lee, C. J. D. Lin, B. Lucini, M. Mesiti, M. Piai, D. Vadacchino, *Sp(2N) Lattice Gauge Theories and Extensions of the Standard Model of Particle Physics*, Universe, **9**, 5, 236 (2023), arXiv:2304.01070 [hep-lat].
[22] G. Howard, *Lie Algebras In Particle Physics : from Isospin To Unified Theories*, (2000), Boca Raton by Taylor & Francis
[23] P. Woit, *Quantum Theory, Groups and Representations*, (2017), Springer.
[24] B. C. Hall, *Lie Groups, Lie Algebras, and Representations*, (2015), Springer.

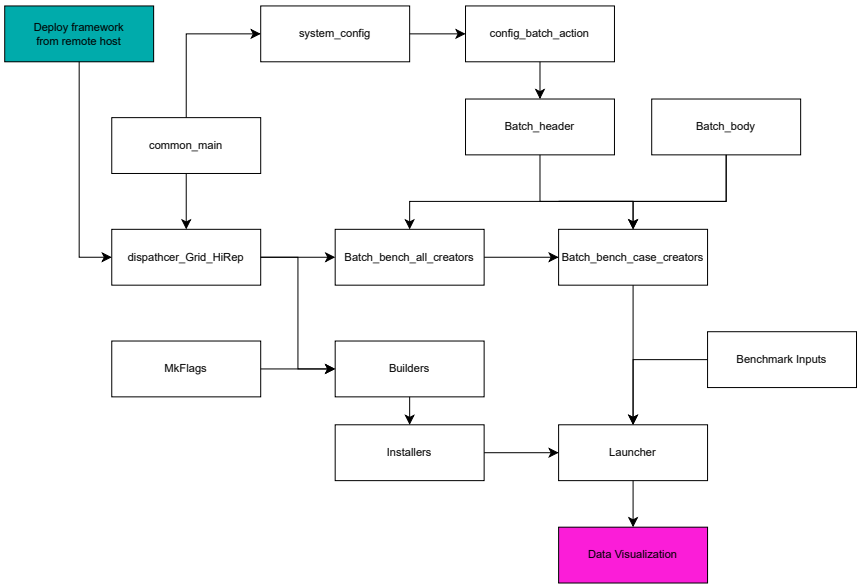


Fig. A.1. Overall code architecture of the framework `Bench_Grid_HiRep`.

Appendix A. Benchmarking graphs from `BKeeper` and `SOMBRERO`

In this Section we show the results for the various runs on all clusters that we gained access via the `EuroHPC` on project number `EHPC-BEN-2024B10-015`. The machines includes `Vega-CPU`, `Vega-GPU`, `Lumi-C`, `Lumi-G`, `Leonardo-DCGP` and `Leonardo-Booster`. The results are divided in the `SOMBRERO` and `BKeeper` distinctly. The figures are referred in the main text, in their appropriate sections.

A.1. Code architecture of the framework `Bench_Grid_HiRep`

In Fig A.1 the overall code architecture is shown.

A.2. `BKeeper` results

In Fig. A.6, we show the plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental using `BKeeper`. a). `Vega-GPU` *node*001. b). `Vega-GPU` *node*002. c). `Leonardo-Booster` *node*001. d). `Leonardo-Booster` *node*002.

In Fig A.9, we show the plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental using `BKeeper`. a). `Lumi-G` *node*001. b). `Lumi-G` *node*002.

In Fig. A.14, we show the plots for `Vega-GPU` and `Leonardo-Booster`. The plots are for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes `node001` and `node002` for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` for the `Small` lattice $24^3 \times 32$.

In Fig. A.17, we show the plots, on on `Lumi-G`, for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes `node001` and `node002` for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` for the `Small` lattice $24^3 \times 32$.

In Fig A.22, we show plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes *node*001 and *node*002 for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` on $n_{\text{nodes}} = \{4, 8, 12\}$.

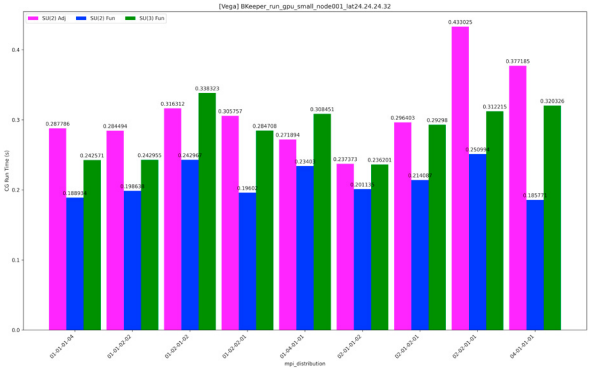


Fig. A.2. a). `Vega-GPU` *node*001.

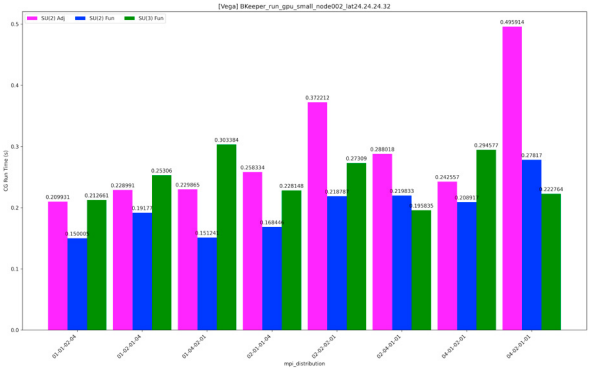


Fig. A.3. b). `Vega-GPU` *node*002.

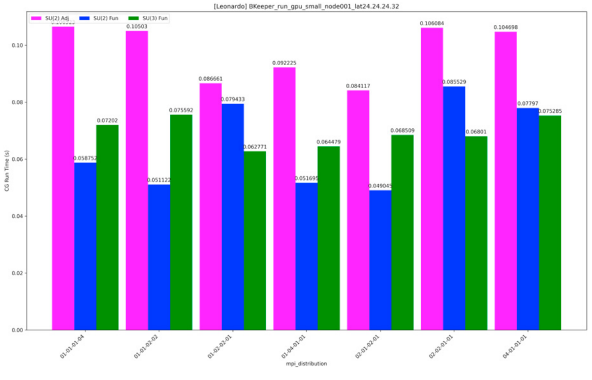


Fig. A.4. c). `Leonardo-Booster` *node*001.

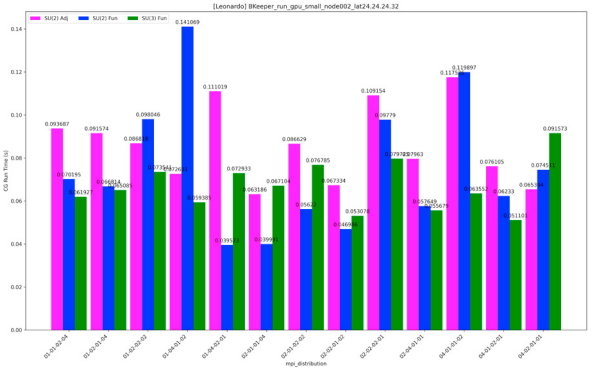


Fig. A.5. d). `Leonardo-Booster` *node*002.

Fig. A.6. Plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental using `BKeeper`. a). `Vega-GPU` *node*001. b). `Vega-GPU` *node*002. c). `Leonardo-Booster` *node*001. d). `Leonardo-Booster` *node*002.

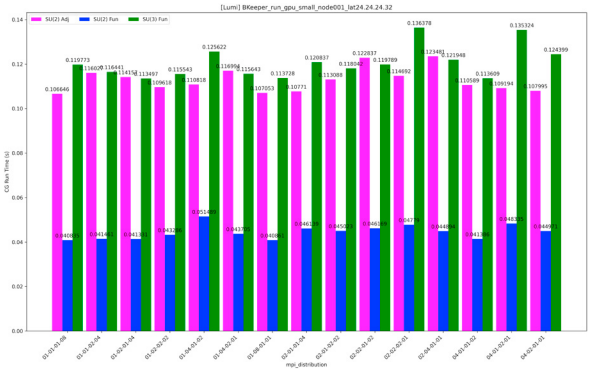


Fig. A.7. a). `Lumi-G` *node*001.

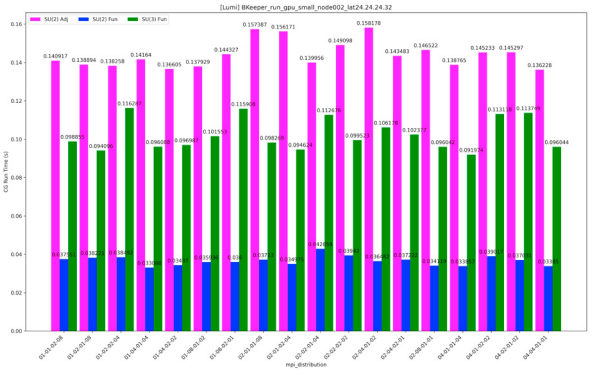


Fig. A.8. b). `Lumi-G` *node*002.

Fig. A.9. Plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental using `BKeeper`. c). `Lumi-G` *node*001. d). `Lumi-G` *node*002.

In Fig A.27, we show plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes *node*001 and *node*002 for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` on $n_{\text{nodes}} = \{4, 8, 12, 16, 24, 32\}$.

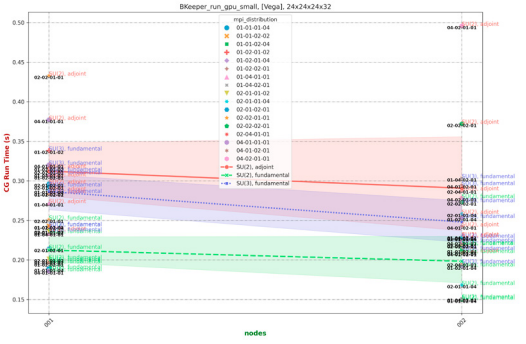


Fig. A.10. a). `Vega-GPU`, (`CG Run Time (s)`).

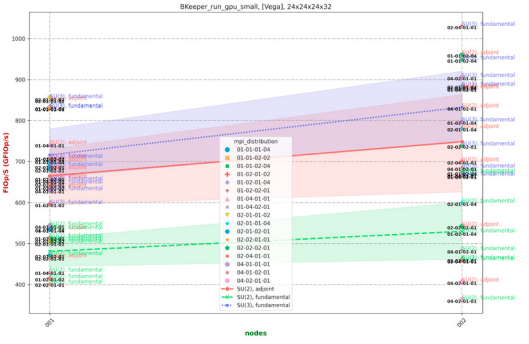


Fig. A.11. b). `Vega-GPU`, (`Flops/s (GFlop/s)`).

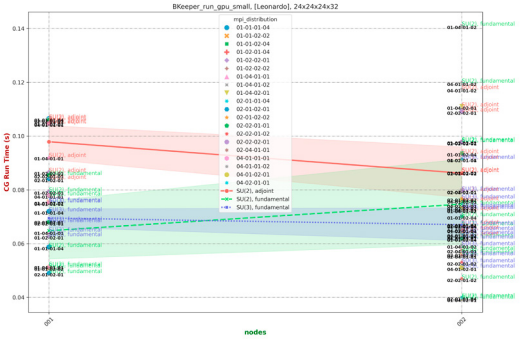


Fig. A.12. c). `Leonardo-Booster`, (`CG Run Time (s)`).

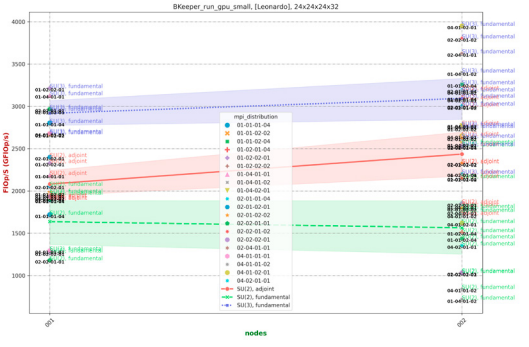


Fig. A.13. d). `Leonardo-Booster`, (`Flops/s (GFlop/s)`).

Fig. A.14. Plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes `node001` and `node002` for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` for the `Small` lattice $24^3 \times 32$.

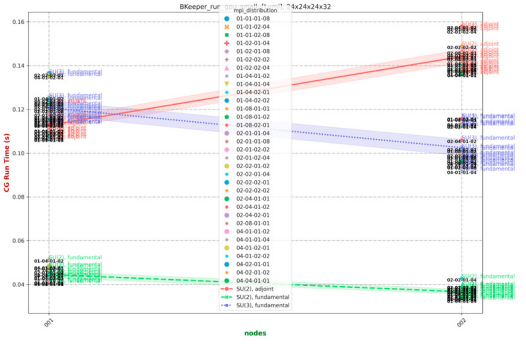


Fig. A.15. a). `Lumi-G`, (`CG Run Time (s)`).

Fig. A.16. b). `Lumi-G`, (`Flops/s (GFlop/s)`).

Fig. A.17. Plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes `node001` and `node002` for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` for the `Small` lattice $24^3 \times 32$.

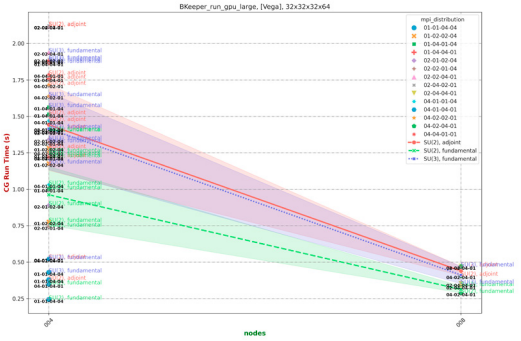


Fig. A.18. a). `Vega-GPU`, (`CG Run Time (s)`) ($32^3 \times 64$).

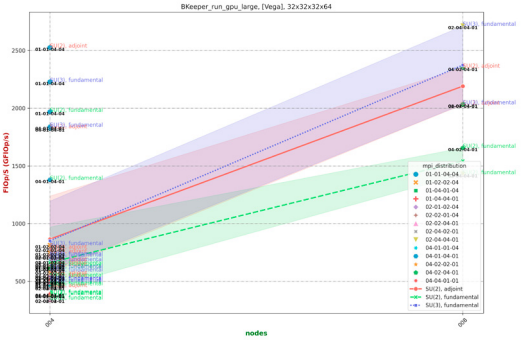


Fig. A.19. b). `Vega-GPU`, (`Flops/s (GFlop/s)`) ($32^3 \times 64$).

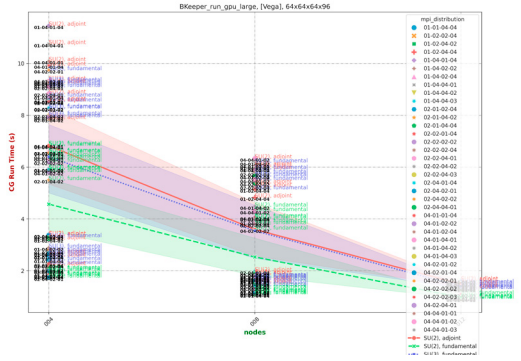


Fig. A.20. c). `Vega-GPU`, (`CG Run Time (s)`) ($64^3 \times 96$).

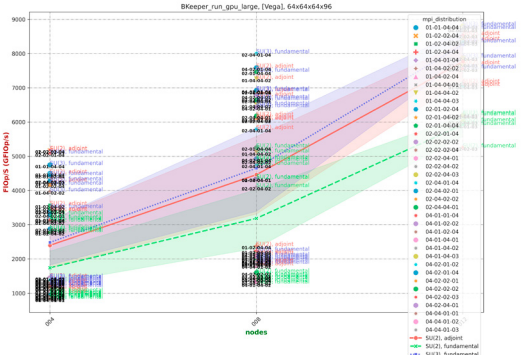


Fig. A.21. d). `Vega-GPU`, (`Flops/s (GFlop/s)`) ($64^3 \times 96$).

Fig. A.22. Plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes *node*001 and *node*002 for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` on $n_{\text{nodes}} = \{4, 8, 12\}$.

A.3. *Vega-CPU plots Weak small–large*

In Fig. A.34 we show the plots for `Vega-CPU` in the case of `SOMBRERO`–weak–small–large for the `mpi_distribution` vs `GFlops per seconds (s)`.

In Fig. A.41 we show the plots for `Vega-CPU` in the case of `SOMBRERO`–weak–small–large for the `nodes` vs `GFlops per seconds (s)`.

A.4. *Lumi-C plots Weak small–large*

In Fig. A.48 we show the plots for `Lumi-C` in the case of `SOMBRERO`–weak–small–large for the `mpi_distribution` vs `GFlops per seconds (s)`.

In Fig. A.55 we show the plots for `Lumi-C` in the case of `SOMBRERO`–weak–small–large for the `nodes` vs `GFlops per seconds (s)`.

A.5. *Leonardo-DCGP plots Weak small–large*

In Fig. A.62 we show the plots for `Leonardo-DCGP` in the case of `SOMBRERO`–weak–small–large for the `mpi_distribution` vs `GFlops per seconds (s)`.

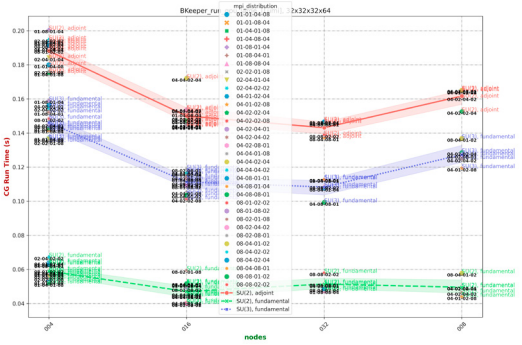


Fig. A.23. a). `Lumi-G`, (`CG Run Time (s)`) ($32^3 \times 64$).

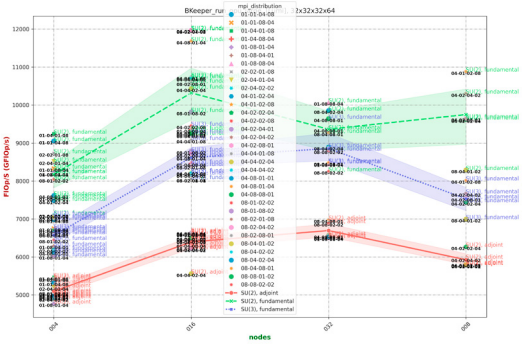


Fig. A.24. b). `Lumi-G`, (`Flops/s (GFlop/s)`) ($32^3 \times 64$).

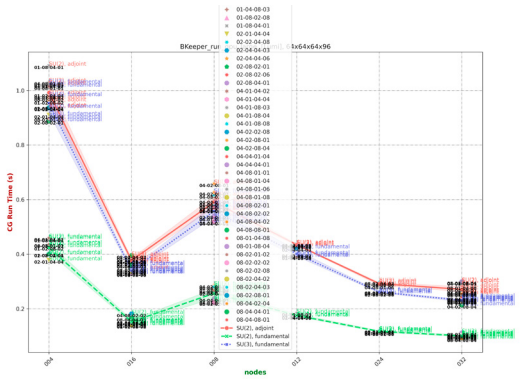


Fig. A.25. c). `Lumi-G`, (`CG Run Time (s)`) ($64^3 \times 96$).

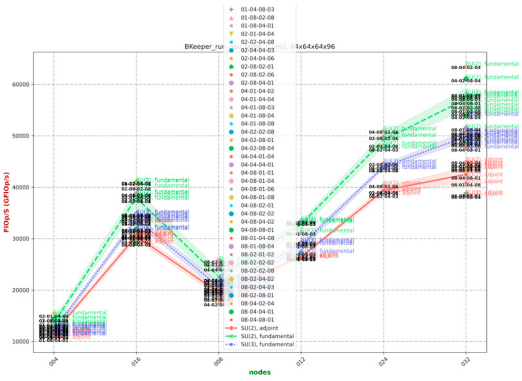


Fig. A.26. d). `Lumi-G`, (`Flops/s (GFlop/s)`) ($64^3 \times 96$).

Fig. A.27. Plots for three different representation $SU(2)$–Adjoint and $SU(2,3)$–Fundamental for number of nodes *node*001 and *node*002 for nodes versus both `CG Run Time (s)` and `Flops/s (GFlop/s)` on $n_{\text{nodes}} = \{4, 8, 12, 16, 24, 32\}$.

In Fig. A.69 we show the plots for `Leonardo-DCGP` in the case of `SOMBRERO`–weak–small–large for the `nodes` vs `GFlops per seconds (s)`.

A.6. Vega-CPU plots Strong small–large

In Fig. A.76 we show the plots for `Vega-CPU` in the case of `SOMBRERO`–strong–small–large for the `mpi_distribution` vs `GFlops per seconds (s)`.

In Fig. A.83 we show the plots for `Vega-CPU` in the case of `SOMBRERO`–strong–small–large for the `nodes` vs `GFlops per seconds (s)`.

A.7. Lumi-C plots Strong small–large

In Fig. A.90 we show the plots for `Lumi-C` in the case of `SOMBRERO`–strong–small–large for the `mpi_distribution` vs `GFlops per seconds (s)`.

In Fig. A.97 we show the plots for `Lumi-C` in the case of `SOMBRERO`–strong–small–large for the `nodes` vs `GFlops per seconds (s)`.

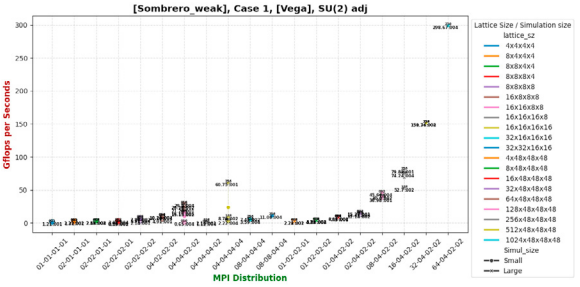


Fig. A.28. a). `Vega-CPU`, (`case-1:` $SU(2)$ `adj`).

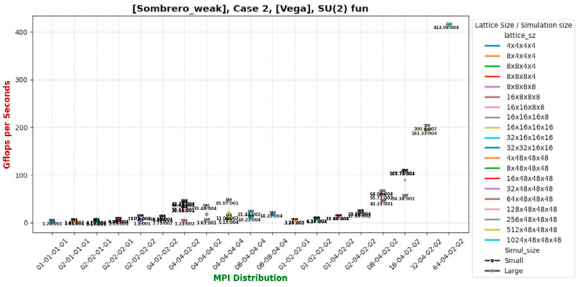


Fig. A.29. b). `Vega-CPU`, (`case-2:` $SU(2)$ `fun`).

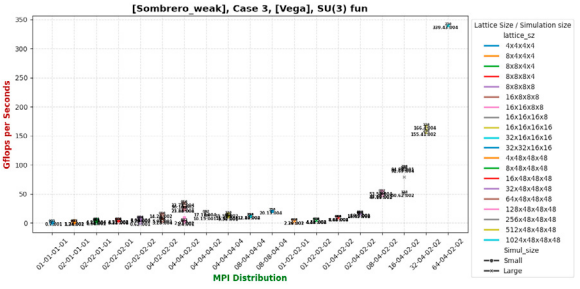


Fig. A.30. c). `Vega-CPU`, (`case-3:` $SU(3)$ `fun`).

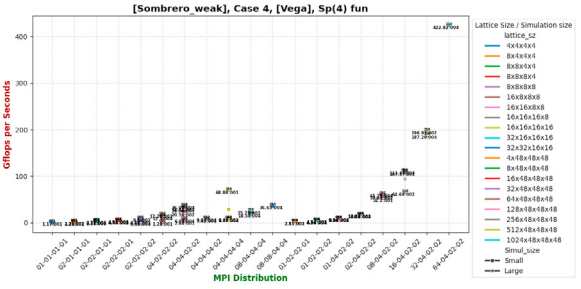


Fig. A.31. d). `Vega-CPU`, (`case-4:` $Sp(4)$ `fun`).

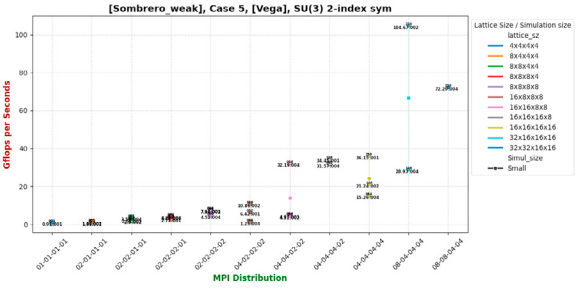


Fig. A.32. c). `Vega-CPU`, (`case-5:` $SU(3)$ `adj`).

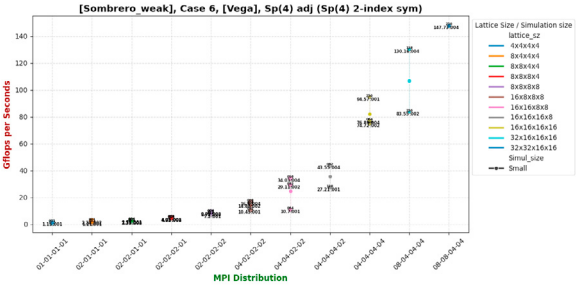


Fig. A.33. d). `Vega-CPU`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.34. Plots for the six different representation within `SOMBRERO` on each node. `mpi_distribution` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Vega-CPU` for the `Weak` cases.

A.8. `Leonardo-DCGP` plots `Strong` small–large

In Fig. A.104 we show the plots for `Leonardo-DCGP` in the case of `SOMBRERO`–strong–small–large for the `mpi_distribution` vs `GFlops per seconds (s)`.

In Fig. A.104 we show the plots for `Leonardo-DCGP` in the case of `SOMBRERO`–strong–small–large for the `nodes` vs `GFlops per seconds (s)`.

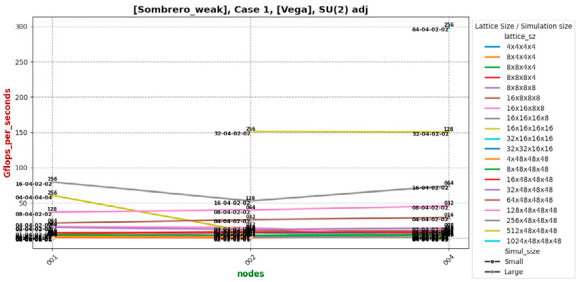


Fig. A.35. a). `Vega-CPU`, (`case-1:` $SU(2)$ `adj`).

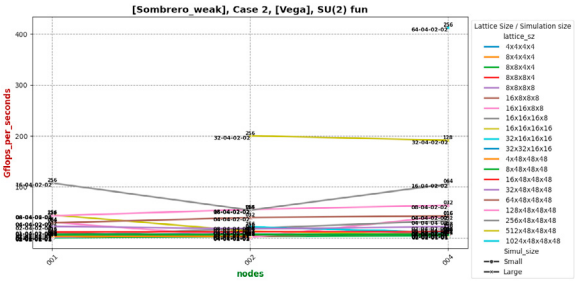


Fig. A.36. b). `Vega-CPU`, (`case-2:` $SU(2)$ `fun`).

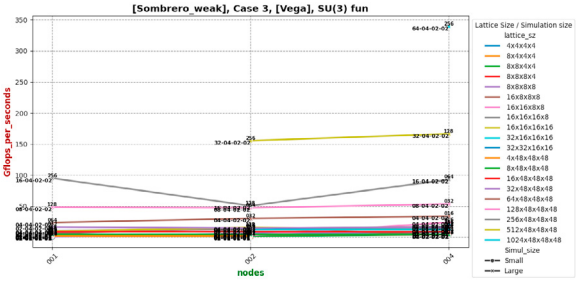


Fig. A.37. c). `Vega-CPU`, (`case-3:` $SU(3)$ `fun`).

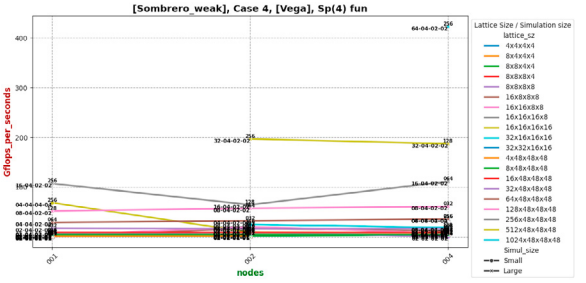


Fig. A.38. d). `Vega-CPU`, (`case-4:` $Sp(4)$ `fun`).

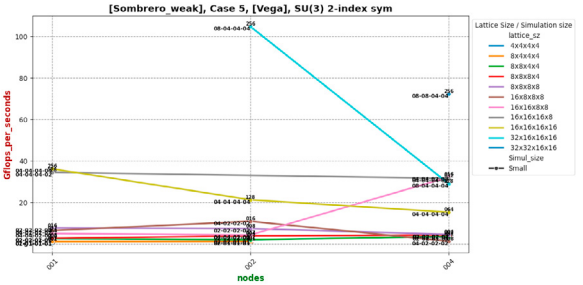


Fig. A.39. c). `Vega-CPU`, (`case-5:` $SU(3)$ `adj`).

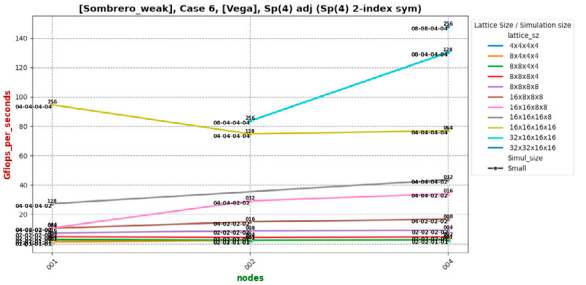


Fig. A.40. d). `Vega-CPU`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.41. Plots for the six different representation within `SOMBRERO` on each node. `nodes` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Vega-CPU` for the `Weak` cases.

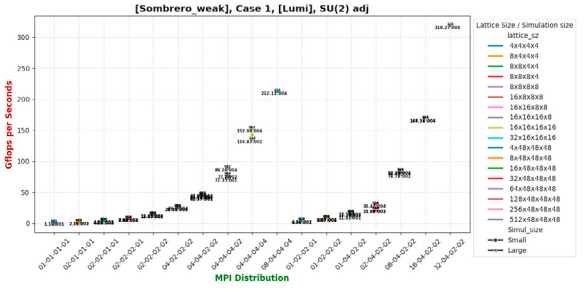


Fig. A.42. a). `Lumi-C`, (`case-1:` $SU(2)$ `adj`).

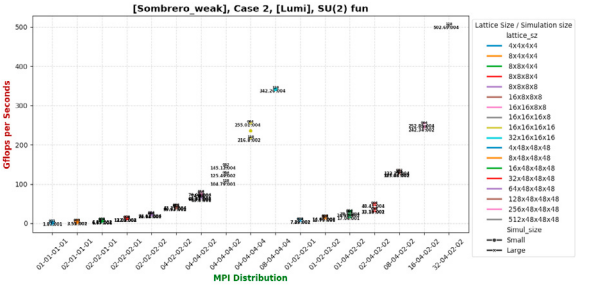


Fig. A.43. b). `Lumi-C`, (`case-2:` $SU(2)$ `fun`).

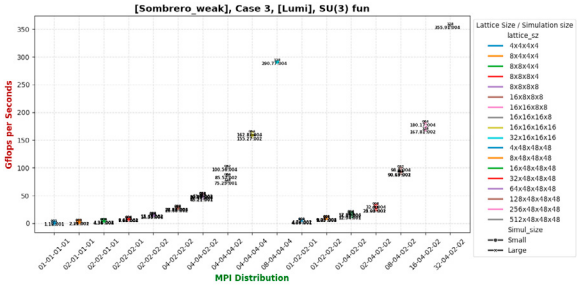


Fig. A.44. c). `Lumi-C`, (`case-3:` $SU(3)$ `fun`).

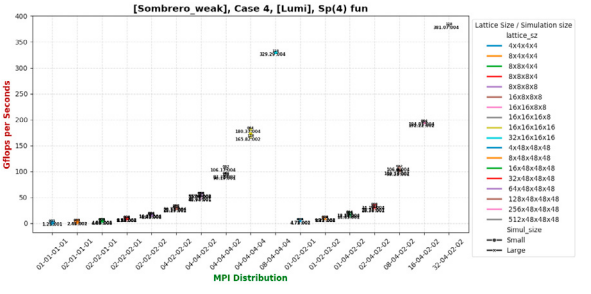


Fig. A.45. d). `Lumi-C`, (`case-4:` $Sp(4)$ `fun`).

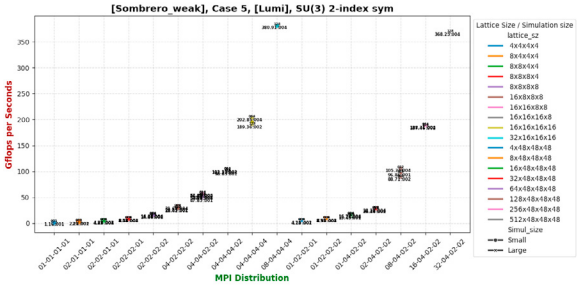


Fig. A.46. c). `Lumi-C`, (`case-5:` $SU(3)$ `adj`).

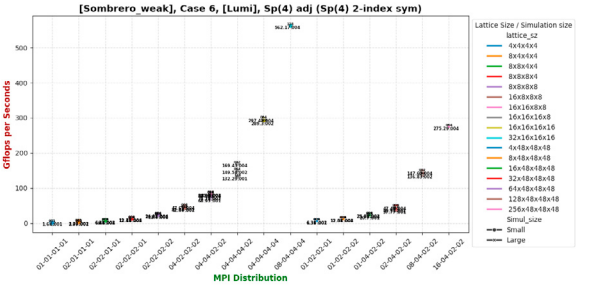


Fig. A.47. d). `Lumi-C`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.48. Plots for the six different representation within `SOMBRERO` on each node. `nodes` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Lumi-C` for the `Weak` cases.

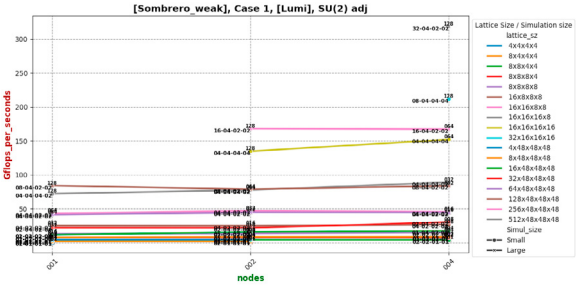


Fig. A.49. a). `Lumi-C`, (`case-1:` $SU(2)$ `adj`).

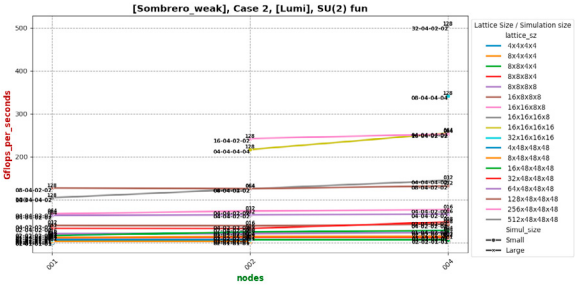


Fig. A.50. b). `Lumi-C`, (`case-2:` $SU(2)$ `fun`).

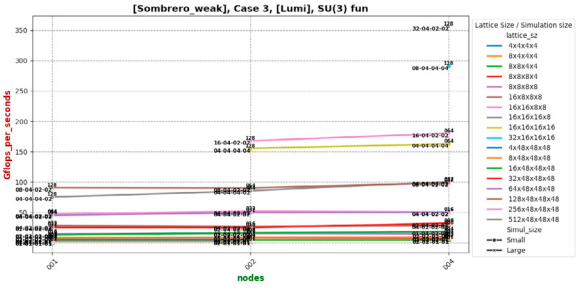


Fig. A.51. c). `Lumi-C`, (`case-3:` $SU(3)$ `fun`).

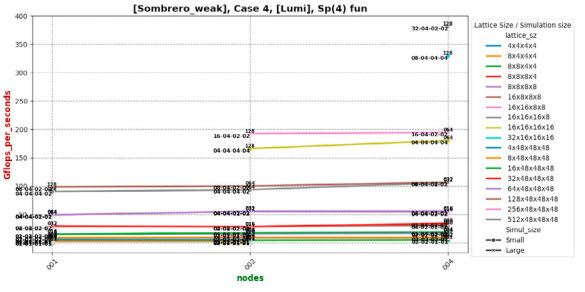


Fig. A.52. d). `Lumi-C`, (`case-4:` $Sp(4)$ `fun`).

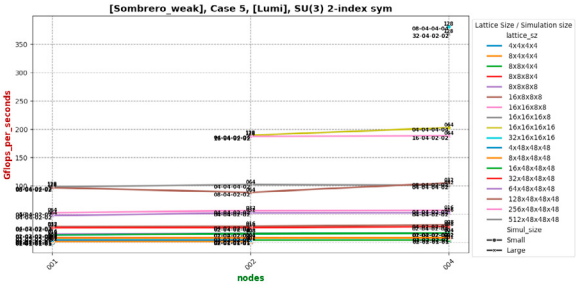


Fig. A.53. c). `Lumi-C`, (`case-5:` $SU(3)$ `adj`).

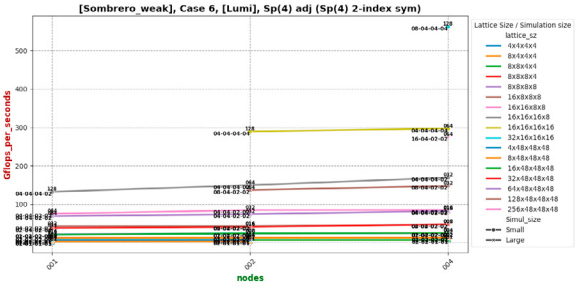


Fig. A.54. d). `Lumi-C`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.55. Plots for the six different representation within `SOMBRERO` on each node. `nodes` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Lumi-C` for the `Weak` cases.

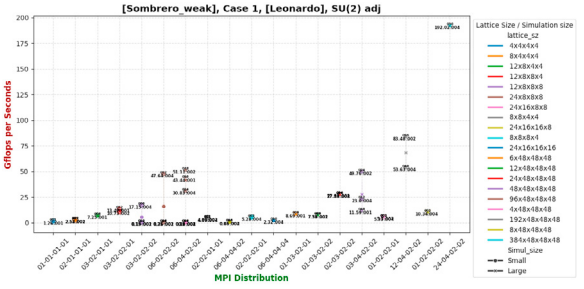


Fig. A.56. a). Leonardo-DCGP, (case-1: $SU(2)$ adj).

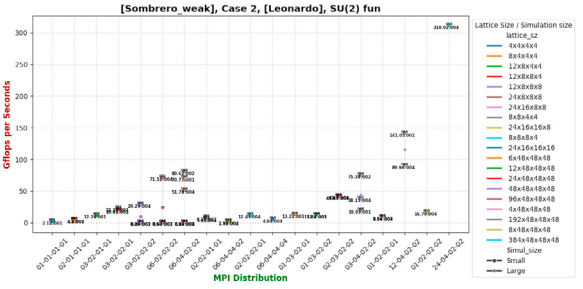


Fig. A.57. b). Leonardo-DCGP, (case-2: $SU(2)$ fun).

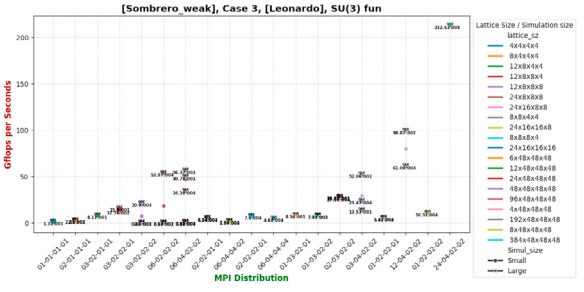


Fig. A.58. c). Leonardo-DCGP, (case-3: $SU(3)$ fun).

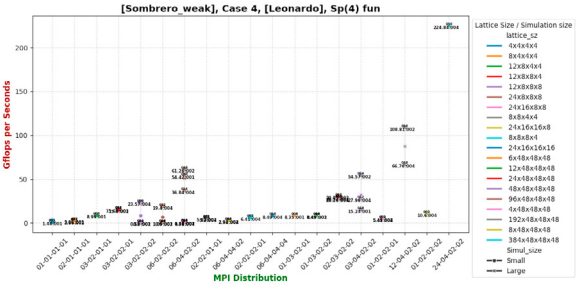


Fig. A.59. d). Leonardo-DCGP, (case-4: $Sp(4)$ fun).

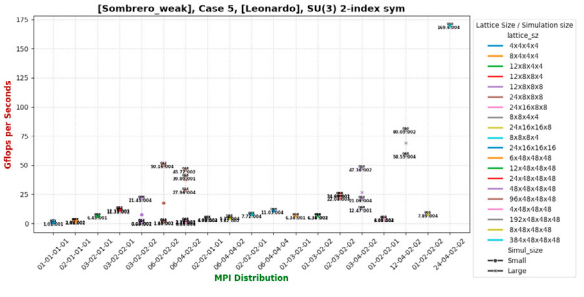


Fig. A.60. c). Leonardo-DCGP, (case-5: $SU(3)$ adj).

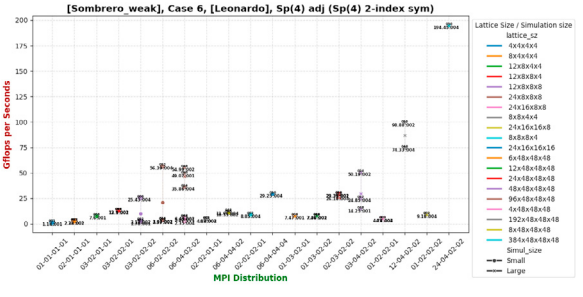


Fig. A.61. d). Leonardo-DCGP, (case-6: $Sp(4)$ adj).

Fig. A.62. Plots for the six different representation within SOMBRERO on each node. mpi_distribution versus GFlops per seconds (s) on $n_{\text{nodes}} = \{1, 2, 4\}$ on Leonardo-DCGP for the Weak cases.

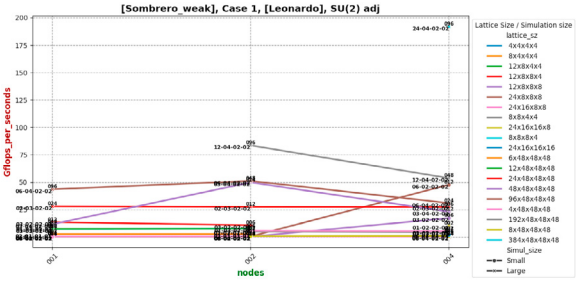


Fig. A.63. a). Leonardo-DCGP, (case-1: $SU(2)$ adj).

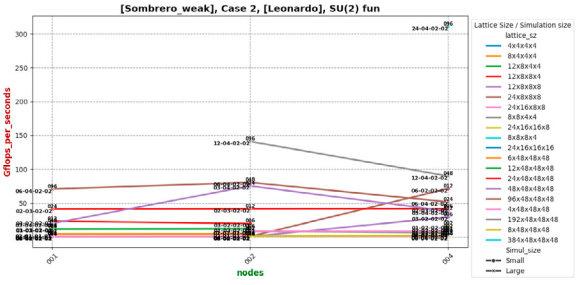


Fig. A.64. b). Leonardo-DCGP, (case-2: $SU(2)$ fun).

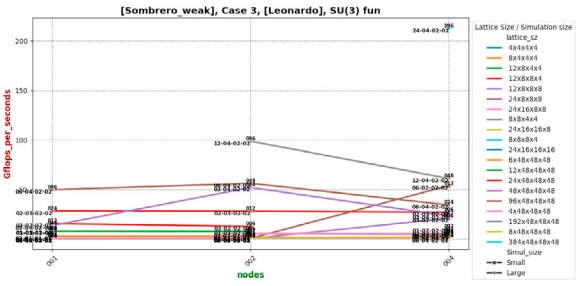


Fig. A.65. c). Leonardo-DCGP, (case-3: $SU(3)$ fun).

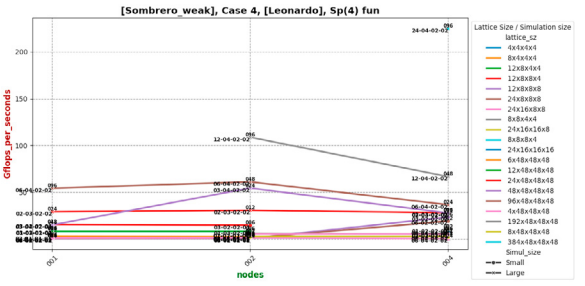


Fig. A.66. d). Leonardo-DCGP, (case-4: $Sp(4)$ fun).

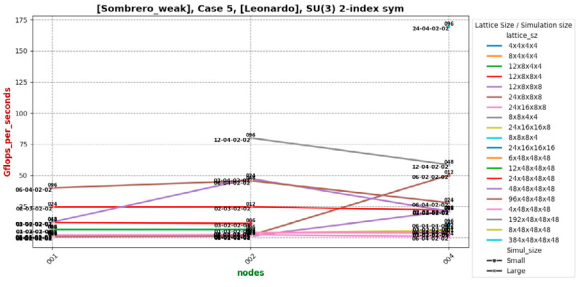


Fig. A.67. c). Leonardo-DCGP, (case-5: $SU(3)$ adj).

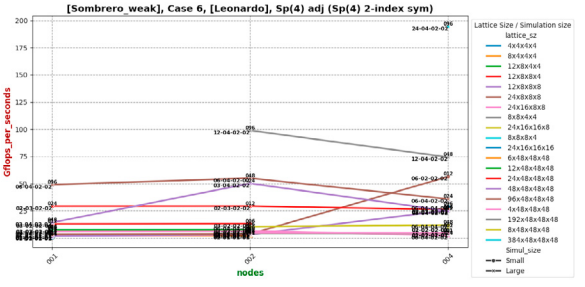


Fig. A.68. d). Leonardo-DCGP, (case-6: $Sp(4)$ adj).

Fig. A.69. Plots for the six different representation within SOMBRERO on each node. nodes versus GFlops per seconds (s) on $n_{\text{nodes}} = \{1, 2, 4\}$ on Leonardo-DCGP for the Weak cases.

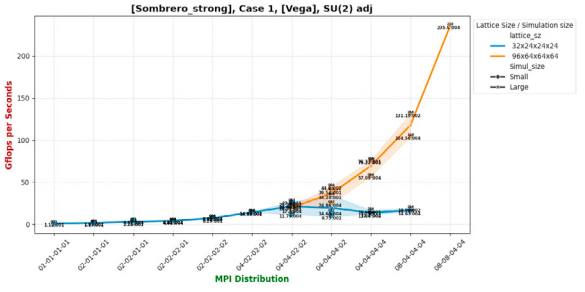


Fig. A.70. a). `Vega-CPU`, (`case-1:` $SU(2)$ `adj`).

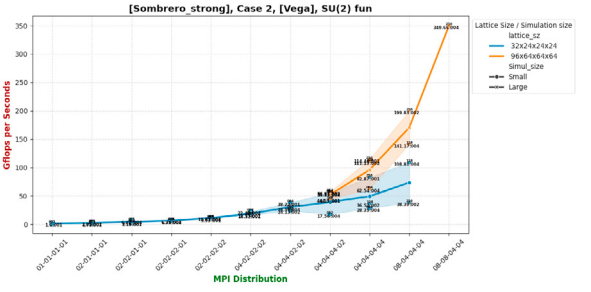


Fig. A.71. b). `Vega-CPU`, (`case-2:` $SU(2)$ `fun`).

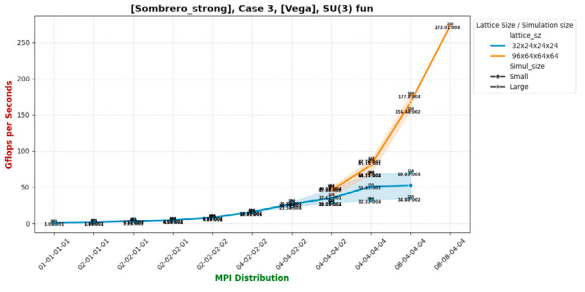


Fig. A.72. c). `Vega-CPU`, (`case-3:` $SU(3)$ `fun`).

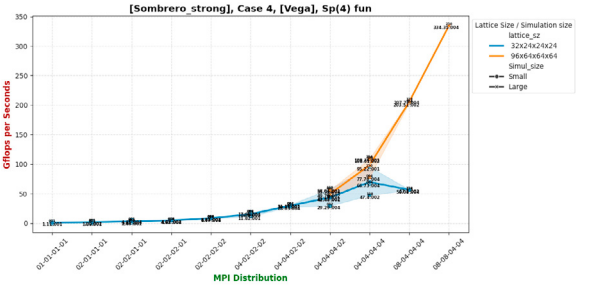


Fig. A.73. d). `Vega-CPU`, (`case-4:` $Sp(4)$ `fun`).

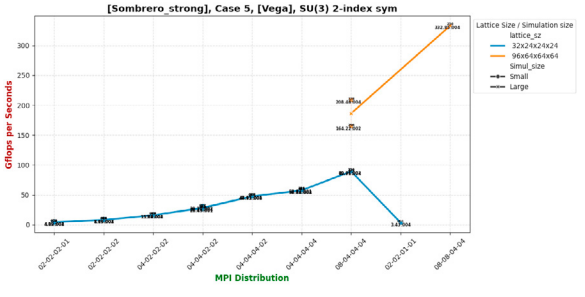


Fig. A.74. c). `Vega-CPU`, (`case-5:` $SU(3)$ `adj`).

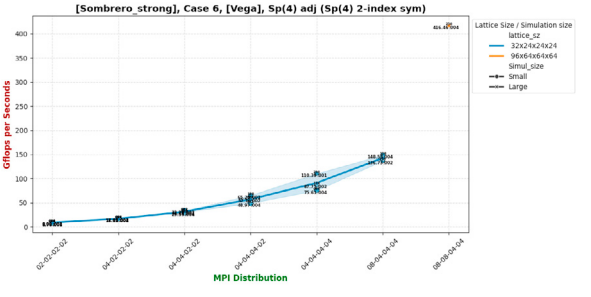


Fig. A.75. d). `Vega-CPU`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.76. Plots for the six different representation within `SOMBRERO` on each node. `mpi_distribution` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Vega-CPU` for the `Strong` cases.

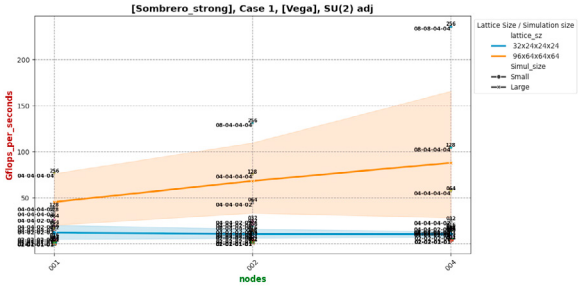


Fig. A.77. a). `Vega-CPU`, (`case-1:` $SU(2)$ `adj`).

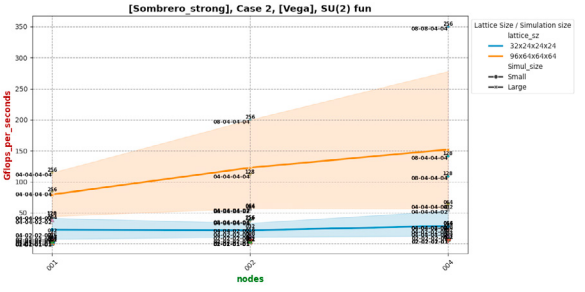


Fig. A.78. b). `Vega-CPU`, (`case-2:` $SU(2)$ `fun`).

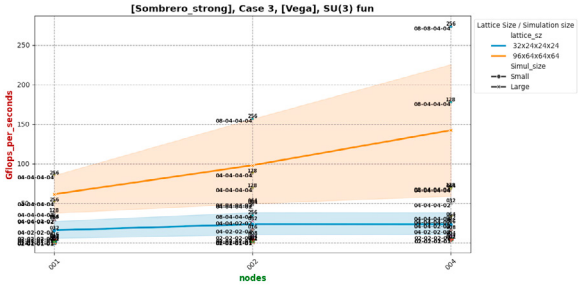


Fig. A.79. c). `Vega-CPU`, (`case-3:` $SU(3)$ `fun`).

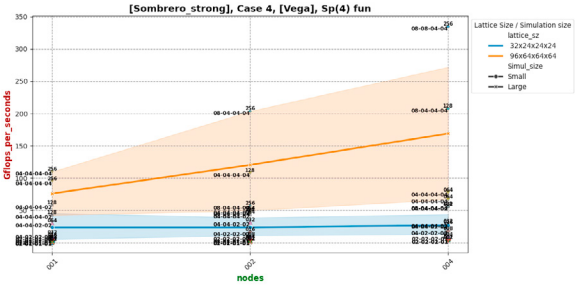


Fig. A.80. d). `Vega-CPU`, (`case-4:` $Sp(4)$ `fun`).

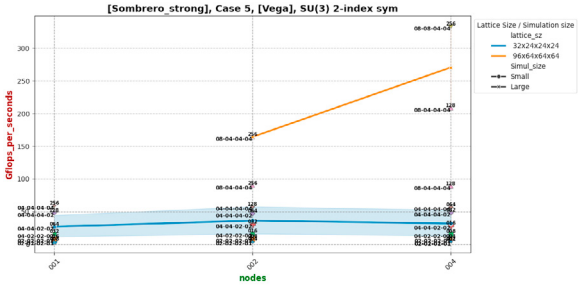


Fig. A.81. c). `Vega-CPU`, (`case-5:` $SU(3)$ `adj`).

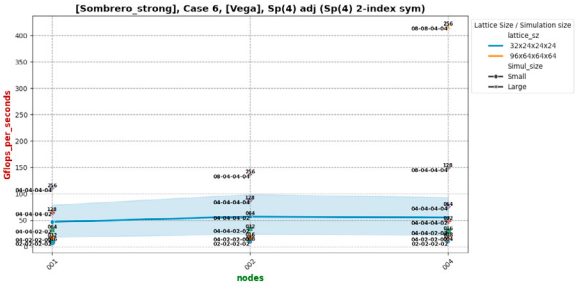


Fig. A.82. d). `Vega-CPU`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.83. Plots for the six different representation within `SOMBRERO` on each node. `nodes` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Vega-CPU` for the `Strong` cases.

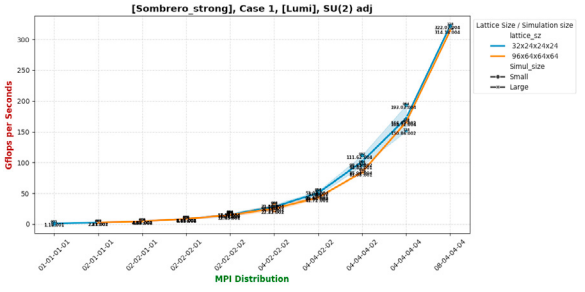


Fig. A.84. a). `Lumi-C`, (`case-1:` $SU(2)$ `adj`).

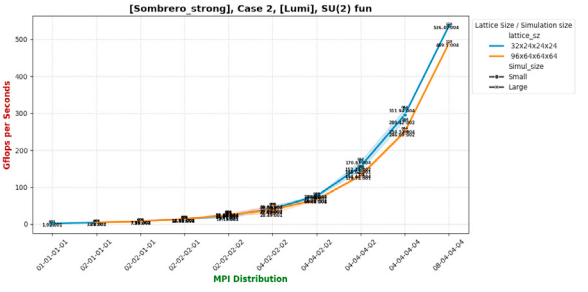


Fig. A.85. b). `Lumi-C`, (`case-2:` $SU(2)$ `fun`).

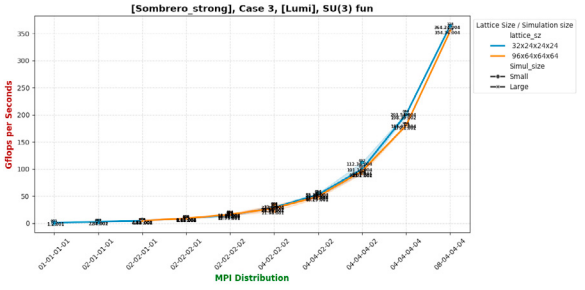


Fig. A.86. c). `Lumi-C`, (`case-3:` $SU(3)$ `fun`).

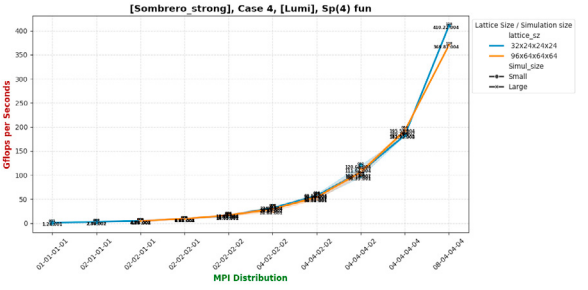


Fig. A.87. d). `Lumi-C`, (`case-4:` $Sp(4)$ `fun`).

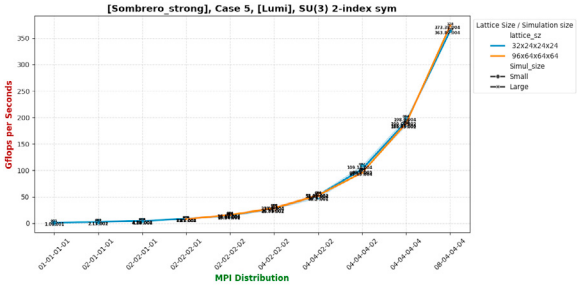


Fig. A.88. c). `Lumi-C`, (`case-5:` $SU(3)$ `adj`).

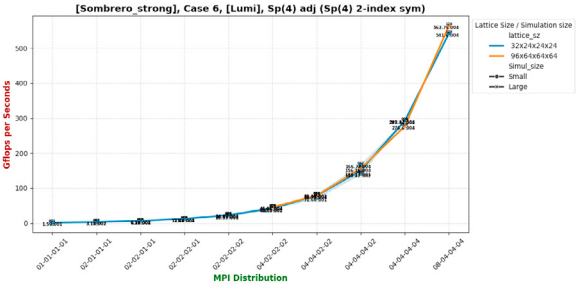


Fig. A.89. d). `Lumi-C`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.90. Plots for the six different representation within `SOMBRERO` on each node. `mpi_distribution` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Lumi-C` for the `Strong` cases.

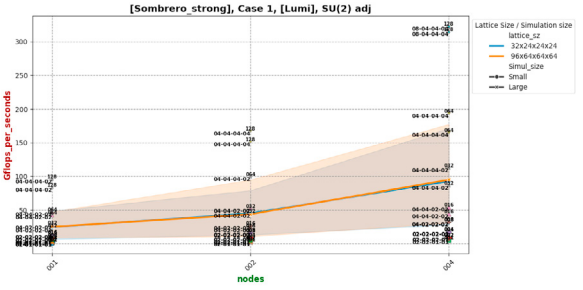


Fig. A.91. a). `Lumi-C`, (`case-1:` $SU(2)$ `adj`).

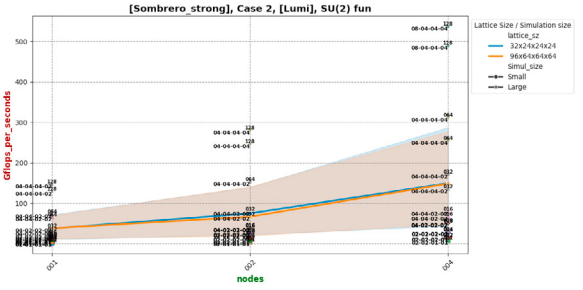


Fig. A.92. b). `Lumi-C`, (`case-2:` $SU(2)$ `fun`).

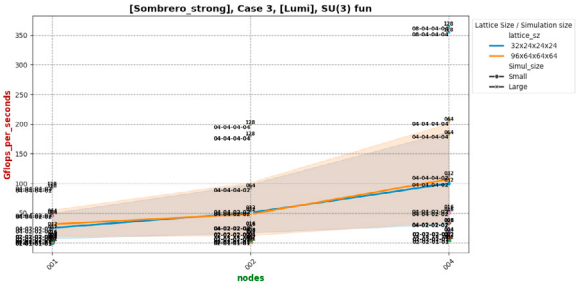


Fig. A.93. c). `Lumi-C`, (`case-3:` $SU(3)$ `fun`).

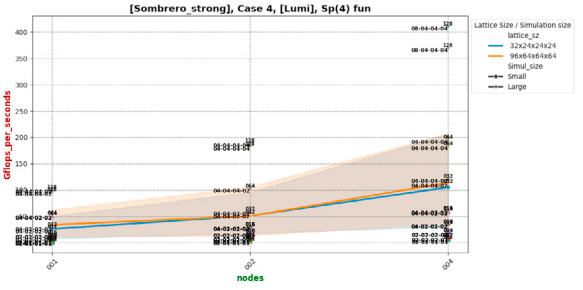


Fig. A.94. d). `Lumi-C`, (`case-4:` $Sp(4)$ `fun`).

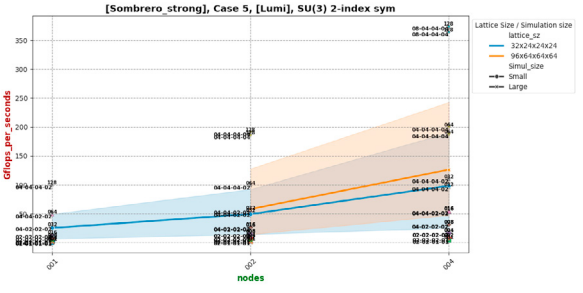


Fig. A.95. c). `Lumi-C`, (`case-5:` $SU(3)$ `adj`).

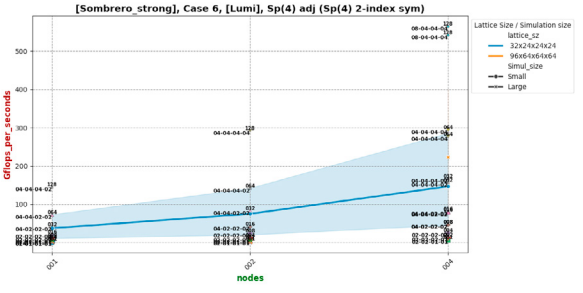


Fig. A.96. d). `Lumi-C`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.97. Plots for the six different representation within `SOMBRERO` on each node. `nodes` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Lumi-C` for the `Strong` cases.

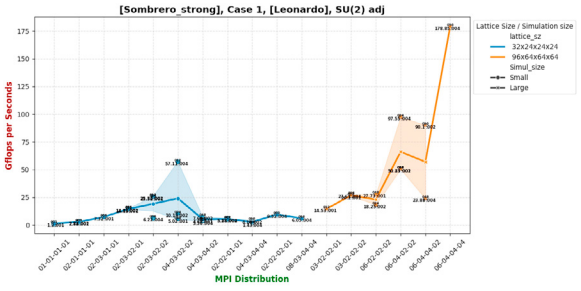


Fig. A.98. a). Leonardo-DCGP, (case-1: $SU(2)$ adj).

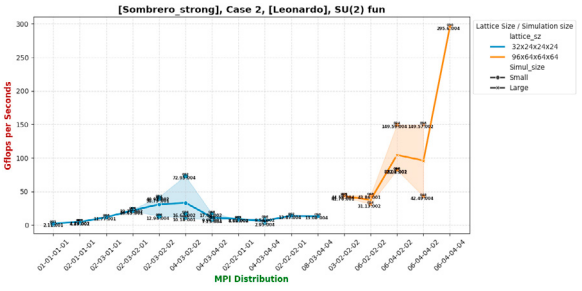


Fig. A.99. b). Leonardo-DCGP, (case-2: $SU(2)$ fun).

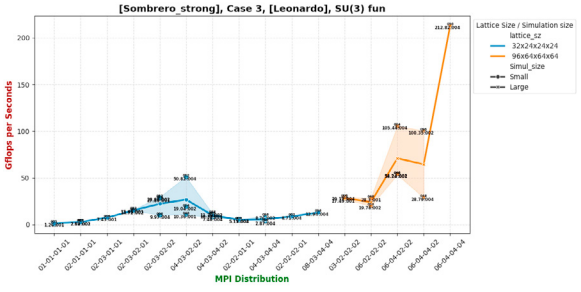


Fig. A.100. c). Leonardo-DCGP, (case-3: $SU(3)$ fun).

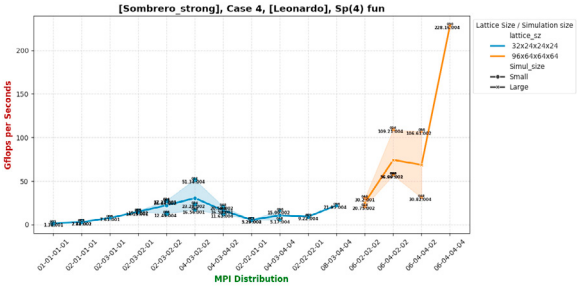


Fig. A.101. d). Leonardo-DCGP, (case-4: $Sp(4)$ fun).

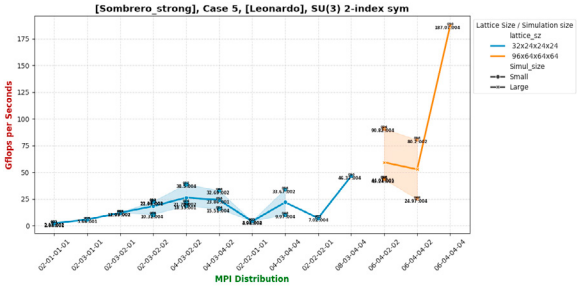


Fig. A.102. c). Leonardo-DCGP, (case-5: $SU(3)$ adj).

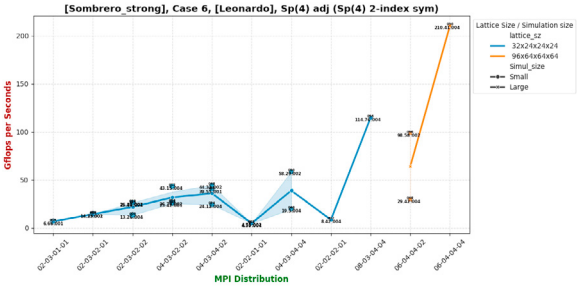


Fig. A.103. d). Leonardo-DCGP, (case-6: $Sp(4)$ adj).

Fig. A.104. Plots for the six different representation within SOMBRERO on each node. mpi_distribution versus GFlops per seconds (s) on $n_{\text{nodes}} = \{1, 2, 4\}$ on Leonardo-DCGP for the Strong cases.

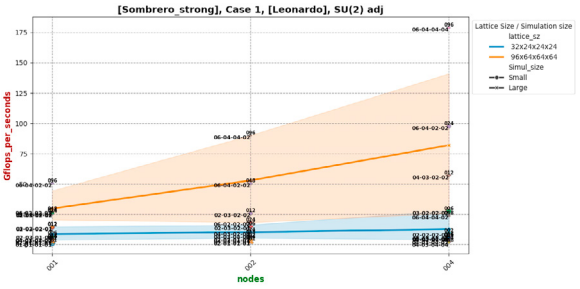


Fig. A.105. a). `Leonardo-DCGP`, (`case-1:` $SU(2)$ `adj`).

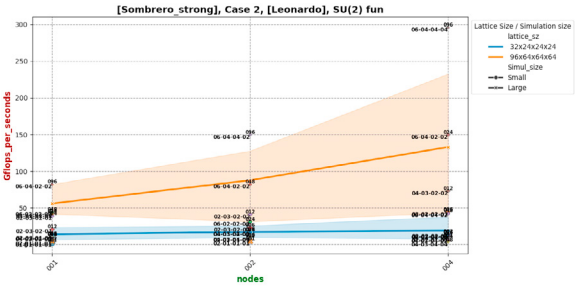


Fig. A.106. b). `Leonardo-DCGP`, (`case-2:` $SU(2)$ `fun`).

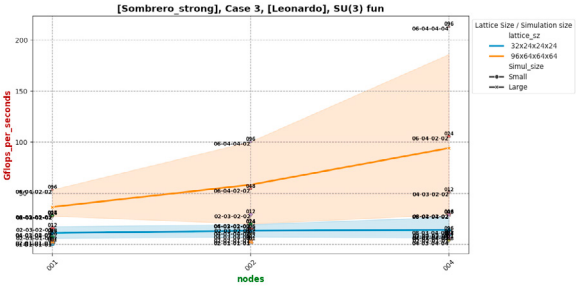


Fig. A.107. c). `Leonardo-DCGP`, (`case-3:` $SU(3)$ `fun`).

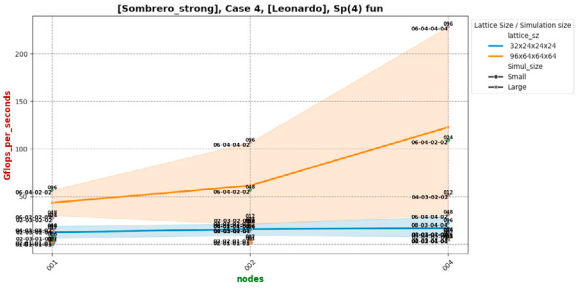


Fig. A.108. d). `Leonardo-DCGP`, (`case-4:` $Sp(4)$ `fun`).

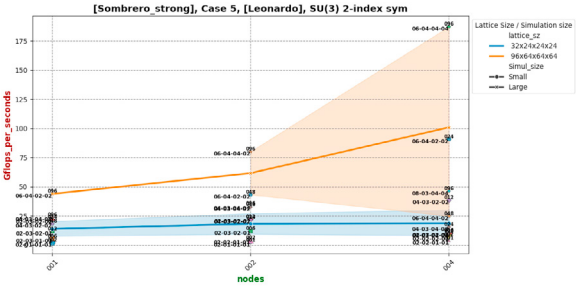


Fig. A.109. c). `Leonardo-DCGP`, (`case-5:` $SU(3)$ `adj`).

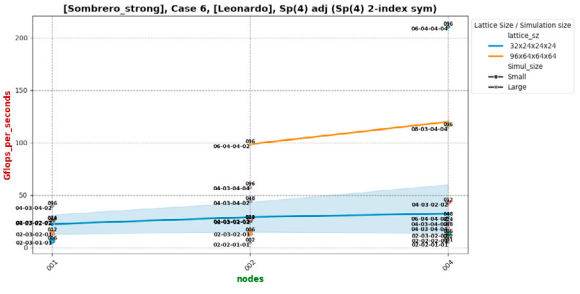


Fig. A.110. d). `Leonardo-DCGP`, (`case-6:` $Sp(4)$ `adj`).

Fig. A.111. Plots for the six different representation within `SOMBRERO` on each node. `nodes` versus `GFlops per seconds (s)` on $n_{\text{nodes}} = \{1, 2, 4\}$ on `Vega-CPU` for the `Strong` cases.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 40–51

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

accLB: A High-Performance Lattice Boltzmann Code for Multiphase Turbulence on Multi-Gpu Architectures

Marco Lauricella[a], Aritra Mukherjee[b], Luca Brandt[b,c], Sauro Succi[a,d,e], Daulet Izbassarov[f], Andrea Montessori[g,]

[a]*Istituto per le Applicazioni del Calcolo, Consiglio Nazionale delle Ricerche, Via dei Taurini 19, Rome, 00185, Italy*
[b]*Department of Energy and Process Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway*
[c]*Department of Environment, Land and Infrastructure Engineering (DIATI), Politecnico di Torino, Torino, Italy*
[d]*Center for Life Nano- & Neuro-Science, Fondazione Istituto Italiano di Tecnologia, Viale Regina Elena 295, Rome, 00161, Italy*
[e]*Department of Physics, Harvard University, 17 Oxford St., Cambridge, 02138, MA, USA*
[f]*Finnish Meteorological Institute, Erik Palmenin aukio 1, Helsinki, 00560, Finland*
[g]*Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University, Via Vito Volterra, Rome, 00146, Italy*

Abstract

In this work, we present accLB, a high-performance Fortran-based lattice Boltzmann (LB) solver tailored to multiphase turbulent flows on multi-GPU architectures. The code couples a conservative phase-field formulation of the Allen–Cahn equation with a thread-safe, regularized LB method to capture complex interface dynamics. Designed from the ground up for HPC environments, accLB employs MPI for distributed memory parallelism and OpenACC for GPU acceleration, achieving excellent portability and scalability on leading pre-exascale systems such as Leonardo and LUMI. Benchmark tests demonstrate strong and weak scaling efficiencies on multiple GPUs. Physical validation includes direct numerical simulations of homogeneous isotropic turbulence (HIT). Further, we examine bubble-laden HIT and observe a transition to a -3 energy scaling, as in experiments and theoretical predictions, due to bubble-induced dissipation, along with enhanced small-scale intermittency. These results highlight accLB as a robust and scalable platform for the simulation of multiphase turbulence in extreme computational regimes.

Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day

Keywords: Turbulent flows; Lattice-Boltzmann method; Phase-field method; GPU computing; OpenACC directives

1. Introduction

Multiphase turbulent flows play a central role in a wide array of natural phenomena and industrial processes, from cloud formation and ocean–atmosphere interactions to chemical reactors and spray combustion [20, 40, 14, 39]. Accurately modeling these flows is particularly challenging due to the complex interplay between turbulence and dispersed phases, which span a vast range of spatial and temporal scales [4, 7]. Even single-phase turbulence exhibits

E-mail address: andrea.montessori@uniroma3.it

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day
10.1016/j.procs.2025.08.231

multiscale dynamics, from the large energy-containing eddies down to the small-scale dissipative motions at the Kolmogorov scale. In multiphase systems, the complexity deepens as interface phenomena push the smallest relevant scales down to the molecular level.

Capturing these coupled multiscale features with high fidelity requires direct numerical simulations (DNS), which are notoriously computationally demanding—especially when resolving deformable fluid interfaces. As such, the development of scalable numerical solvers tailored for high-performance computing (HPC) platforms is essential to advance our understanding of turbulent multiphase flows.

The advent of exascale computing has opened unprecedented opportunities for tackling such grand-challenge simulations. According to the 64th *Top500* list, the leading exascale systems—El Capitan, Frontier, and Aurora—are hosted in the United States, while Europe is home to four pre-exascale machines among the global top ten: HPC6 (Italy), Alps (Switzerland), LUMI (Finland), and Leonardo (Italy). These machines are built on heterogeneous architectures, combining powerful CPUs with GPU accelerators such as AMD GPUs (HPC6 and LUMI) or NVIDIA GPUs (Alps and Leonardo), pushing the boundaries of performance and energy efficiency.

Modern HPC systems, particularly those leveraging GPU-based architectures, have significantly expanded the scope of direct numerical simulations in computational fluid dynamics (CFD). Several open-source CFD solvers, including AFiD [45], STREAmS [2], CaNS [6], TLBfind [33, 34], and LBcuda [3], have been optimized for NVIDIA GPUs using CUDA Fortran. However, maintaining performance portability across different GPU vendors remains a pressing challenge. To overcome this, recent efforts have embraced directive-based programming models, such as OpenACC, which offer a path toward cross-platform compatibility. Codes like FluTAS [8], CaNS-Fizzy [25], and the pseudo-spectral DNS solver ported by Roccon [38] exemplify this approach. Most recently, URANOS-2.0 [9] showcased a portable OpenACC-based implementation for both AMD and NVIDIA platforms.

In this work, we present accLB, a high-performance, thread-safe lattice Boltzmann solver specifically developed to simulate multiphase turbulent flows on modern GPU-accelerated supercomputers. The solver combines a conservative phase-field formulation based on the Allen–Cahn equation with a regularized LBM scheme, enabling accurate tracking of complex interface dynamics under large density and viscosity contrasts. Its architecture employs hybrid MPI–OpenACC parallelism, ensuring both scalability and portability across heterogeneous computing environments.

We validate the solver's performance and accuracy through extensive benchmarks on EuroHPC pre-exascale systems Leonardo and LUMI, demonstrating both strong and weak scaling up to 64 GPUs and achieving sustained throughputs exceeding 150 GLUPS(Giga Lattice Updates Per Second). Direct numerical simulations of single- and two-phase homogeneous isotropic turbulence confirm the robustness and fidelity of the accLB framework [24] in capturing the rich dynamics of multiphase turbulence.

2. Methods

2.1. Navier-Stokes equations for multiphase flows with interfaces

The multiphase system analyzed in this study is described by the continuity and momentum equations, which are numerically solved using the lattice Boltzmann method (LBM). These equations read:

$$\partial_t \rho + \partial_\alpha(\rho u_\alpha) = 0, \tag{1}$$

$$\partial_t(\rho u_\alpha) + \partial_\beta(\rho u_\alpha u_\beta) = -\partial_\alpha p + \partial_\beta\left(\rho\nu[\partial_\beta u_\alpha + \partial_\alpha u_\beta]\right) + F^s_\alpha, \tag{2}$$

where u_α denotes the velocity field, ρ the fluid density, p the macroscopic pressure, ν the kinematic viscosity, and F^s_α the surface tension force. Greek indices represent Cartesian tensor components.

The surface tension force F^s_α is expressed as the divergence of the capillary stress tensor, $F^s_\alpha = \partial_\beta\left(\gamma(\delta_{\alpha\beta} - n_\alpha n_\beta)\right)$, where n_α is the local interface normal and γ the surface tension coefficient.

2.1.1. Allen-Cahn Equation for Interface Tracking

The interface between the two fluid phases is tracked using a conservative phase-field formulation based on the Allen-Cahn equation. The scalar order parameter ϕ, which varies smoothly between 0 and 1 across the interface, evolves according to:

$$\partial_t \phi + u_\alpha \partial_\alpha \phi = D\partial_\alpha \partial_\alpha \phi - \kappa \partial_\alpha (\phi(1-\phi) n_\alpha), \tag{3}$$

where D denotes the interface diffusivity and $\kappa = 4D/\delta$, with δ representing the interface width. The interface is implicitly located at the iso-surface $\phi = \phi_0 = 0.5$. The equilibrium profile across an interface centered at x_0 is given by [21], $\phi(x,y,z) = \phi_0 \pm \frac{\phi_H - \phi_L}{2} \tanh\left(\frac{|x-x_0|+|y-y_0|+|z-z_0|}{\delta}\right)$.

This phase-field equation is tightly coupled to the hydrodynamic fields via the advection term.

To accurately resolve interface dynamics in the presence of strong advection, the convective term in Eq. (4) is discretized using a fifth-order Weighted Essentially Non-Oscillatory (WENO-5) scheme [19]. This high-order scheme minimizes numerical dissipation while capturing sharp gradients and reducing spurious oscillations, making it well-suited for turbulent multiphase flows.

2.2. Thread-safe lattice Boltzmann model

The LBM provides an efficient solver for the Navier-Stokes equations, particularly for complex or multiphase flows. In this work, we employ a thread-safe and regularized LBM framework, as described in [30, 24].

The second-order accurate discrete lattice Boltzmann equation over a set of q velocity directions is given by [42, 23]:

$$f_i(x_\alpha + c_{i\alpha}\Delta t, t + \Delta t) = f_i(x_\alpha, t) + \omega(f_i^{eq} - f_i) + (1 - \frac{\omega}{2}) S_i, \tag{4}$$

where ω is a relaxation rate, f_i are the distribution functions, f_i^{eq} their equilibrium counterparts, and S_i denotes an external forcing term incorporated through the Guo forcing scheme [16]:

$$S_i = w_i \left(\frac{c_{i\alpha} - u_\alpha}{c_s^2} + \frac{c_{i\beta} u_\beta}{c_s^4} c_{i\alpha} \right) F_\alpha, \tag{5}$$

being F_α the total body force acting on the fluid.

The relaxation time, $\tau = 1/\omega$, controls the kinematic viscosity through the relation $\nu = c_s^2(\tau - 0.5)$ [42, 23, 29].

Defining $f_i^{neq} = f_i - f_i^{eq} + 0.5 S_i$ and substituting in Eq. 4, the post-collision update can be reformulated as [11, 12]:

$$f_i(x_\alpha + c_{i\alpha}\Delta t, t + \Delta t) = f_i^{eq} + (1-\omega) f_i^{neq} + \frac{1}{2} S_i. \tag{6}$$

In the thread-safe implementation, both equilibrium and non-equilibrium components are reconstructed directly from macroscopic fields, avoiding direct access to full distribution functions and preventing race conditions due to non-local memory operations [30, 31]. The equilibrium distribution reads, $f_i^{eq} = w_i \left(p^* + \frac{c_{i\alpha} u_\alpha}{c_s^2} + \frac{(c_{i\alpha} c_{i\beta} - c_s^2 \delta_{\alpha\beta}) u_\alpha u_\beta}{2 c_s^4} \right)$, where $p^* = p/(\rho c_s^2)$ is the dimensionless pressure. The non-equilibrium contribution is reconstructed up to second order as $f_i^{neq} = \frac{c_{i\alpha} c_{i\beta} a_{1,\alpha\beta}^{(2)}}{2 c_s^4}$, where $a_{1,\alpha\beta}^{(2)}$ denotes the second order Hermite coefficient of non-equilibrium [26]. Macroscopic vari-

ables are obtained from the distribution functions using $p^*(x_\alpha, t) = \sum_i f_i(x_\alpha, t)$ and $u_\alpha(x_\alpha, t) = \sum_i f_i(x_\alpha, t)c_{i\alpha} + \frac{1}{2}\sum_i S_i$: The second order Hermite coefficient, $a^{(2)}_{1,\alpha\beta}$, is assessed by the sum over the discrete non-equilibrium populations f_i^{neq} weighted on the second order Hermite polynomial $\mathcal{H}^{(2)}_{i,\alpha\beta} = c_{i,\alpha}c_{i,\beta} - c_s^2\delta_{\alpha\beta}$ obtaining:

$$a^{(2)}_{1,\alpha\beta}(x_\alpha, t) = \sum_i f_i^{neq}(x_\alpha, t)(c_{i,\alpha}c_{i,\beta} - c_s^2\delta_{\alpha\beta}). \tag{7}$$

where, as shown in Ref. [26], apart from higher-order smallness in the Mach number (Ma), the second order coefficient macroscopically reads: $a^{(2)}_{1,\alpha\beta} = -c_s^2\tau\left[\partial_\alpha\left(u_\beta\right) + \partial_\beta\left(u_\alpha\right)\right] + O(\mathrm{Ma}^3)$.

2.3. Extension for high-density and viscosity contrasts

To accurately simulate multiphase flows with strong density and viscosity contrasts, additional force contributions are introduced into the momentum equation. The total force acting on the system is given by [10]:

$$F_\alpha = F^s_\alpha + F^p_\alpha + F^\nu_\alpha, \tag{8}$$

where F^s_α is the surface tension force, computed via a phase-field-based formulation[21], $F^s_\alpha = -\gamma\kappa\nabla\phi|\nabla\phi|$.

The pressure-induced force reads, $F^p_\alpha = -p^*c_s^2\partial_\alpha\rho$, where $\rho = \phi\rho_l + (1-\phi)\rho_g$ is the local mixture density. Given that the pressure is defined as $p = p^*\rho c_s^2$, its gradient contains implicit terms that require explicit correction via F^p_α.

Lastly, the viscous correction force is given by $F^\nu_\alpha = -\frac{\nu\omega}{c_s^2\Delta t}\left[\sum_i(f_i - f_i^{eq})c_{i\alpha}c_{i\beta}\right]\partial_\alpha\rho$:

Spatial derivatives appearing in the force terms and phase-field evolution equation are discretized using lattice-based stencils, $\partial_\alpha\Psi = \frac{1}{c_s^2}\sum_i w_i\Psi(x_\alpha + c_{i\alpha})c_{i\alpha}$ and $\partial_\alpha\partial_\beta\Psi = \frac{1}{c_s^2}\left(\sum_{i\neq 0} w_i\Psi(x_\alpha + c_{i\alpha}) - w_0\Psi(x_\alpha)\right)$

This modeling framework enables robust and efficient simulations of multiphase flows with large physical property mismatches while maintaining numerical stability and thread safety through the use of regularization techniques.

3. Implementation

The accLB code is developed in Fortran 95, utilizing modules to keep the code organized and reduce clutter. Variables related to specific features (such as fluids and phase-field) or methods (like time integrators) are grouped into separate modules for better structure. Portability and readability are key design goals of accLB, ensuring the code can run across multiple architectures and is accessible for contributors despite the complexity of the underlying model.

From its inception, accLB has been designed with parallel computing platforms in mind. It uses the Message Passing Interface (MPI), the standard for inter-node communication. However, a corresponding serial version of accLB is available and can be selected during compilation.

The parallelization of the hydrodynamic solver in accLB is based on domain decomposition [15], where the simulation domain is divided into one-, two-, or three-dimensional blocks distributed across MPI tasks. The block decomposition is selected at run-time, without the need to recompile the code, allowing users to optimize performance based on the global grid size and the number of available compute cores. For cubic domains, three-dimensional decomposition typically yields the best performance by minimizing the surface area involved in inter-process communication. The current implementation employs a halo region of two lattice spacings, which is necessary to calculate the derivative terms in the WENO5 scheme. This halo width is flexible and could be easily extended in future developments to accommodate more advanced algorithms.

The accLB code leverages OpenACC directives to enable execution on GPU devices, allowing for efficient offloading of compute-intensive tasks. This approach facilitates parallel acceleration with minimal code modifications, ensuring portability across different hardware architectures. By utilizing OpenACC, the code can fully utilize available GPU resources to enhance performance significantly, particularly for large-scale simulations. OpenACC also

ensures broad compatibility across various GPU platforms, including devices from NVIDIA, AMD, and other vendors that support the standard. This flexibility allows the code to remain portable and adaptable to evolving hardware environments without requiring extensive rewrites. Finally, OpenACC enables compilation and execution on purely CPU-based architectures as well. The code exploits a hybrid shared-distributed memory parallelization strategy in such cases, combining OpenACC and MPI to scale across multiple CPU cores and nodes efficiently. This unified parallel framework allows direct performance comparison across different high-performance computing platforms, such as CPU-based versus GPU-based clusters, aiding in performance benchmarking and resource optimization.

The time-marching implementation of the lattice-Boltzmann equation Eq. (4) exploits the reformulation given in Eq. (6) to facilitate completely independent computations at each computational node, effectively eliminating race conditions. This design includes the thread-safe paradigm introduced by Montessori et al. [32, 31, 30]. This approach enhances data locality and removes memory dependencies during the propagation step by reconstructing the post-collision distribution functions through Hermite projection. Such a strategy is particularly advantageous for parallel computing architectures as it ensures safe and efficient execution across multiple processing units without the need for complex synchronization mechanisms. Moreover, this method significantly reduces the memory footprint, enabling the simulation of large-scale fluid dynamics problems with improved performance and stability, achieving a balance between computational efficiency and the accuracy required for simulating complex fluid behaviors. Thus, the primary variables in accLB include the 27 distribution functions alongside with the ten recovered macroscopic fields (dimensionless pressure $p*$, velocity vector u_α, and six Hermite coefficient terms $a^{(2)}_{1,\alpha\beta}$ of the corresponding symmetric tensor). The distribution functions can be stored using one of two competing memory layouts: array of structures (AoS) or structure of arrays (SoA), regardless of the underlying memory order (row-major as in C or column-major as in Fortran) [43]. The accLB code employs the SoA layout, which has demonstrated superior performance over the AoS configuration. In particular, the SoA layout enables a unit-stride access pattern during the streaming step, ensuring coalesced memory access on GPU architectures. This choice significantly improves memory efficiency, reducing bandwidth-related bottlenecks and making it the optimal strategy for high-performance implementations.

4. Performances

To evaluate performance, we simulate two different benchmark cases: 1) a single-component fluid and 2) a two-component system containing a single droplet of heavier density dispersed in a lighter continuous phase (density contrast equal to 100), both with a kinematic viscosity equal to 0.005 in lattice units.

In the present work, we report both strong and weak scaling to assess the parallel performance of the code, where strong scaling provides information on how the execution time decreases with an increasing number of GPUs for a fixed problem size, reflecting the efficiency in accelerating a single workload. Instead, weak scaling analyzes how well the code maintains a constant execution time when the problem size grows proportionally with the number of GPUs, indicating scalability to larger workloads. The two scaling analyses provide a comprehensive view of parallel efficiency and the ability of accLB to handle increasingly demanding simulations.

Hence, the simulation domain is fixed as a cubic box with 512 lattice points per side in the strong scaling benchmarks. In the weak scaling simulations, we consider a cubic sub-domain of side 512 lattice points for each GPU device (fixed sub-domain size), adopting a linear decomposition along the z-axis. Thus, we obtain an overall simulation box with sides 512, 512, and $512*n_p$ with n_p denoting the number of GPU devices for the weak scaling simulations.

Efficiency is assessed using the Giga Lattice Updates Per Second (GLUPS) metric, defined as:

$$\text{GLUPS} = \frac{L_x L_y L_z}{10^9 t_s}, \tag{9}$$

where t_s is the run (wall-clock) time (in seconds) per single time step iteration, while L_x, L_y, and L_z are the domain sizes.

In the present work, we define the speedup, S_p, in terms of GLUPS ratio as follows:

$$S_p = \frac{\text{GLUPS}_p}{\text{GLUPS}_s}, \tag{10}$$

where GLUPS_s denotes the Giga Lattice Updates Per Second for the code running on a single GPU and GLUPS_p represents the Giga Lattice Updates Per Second for the code running on n_p GPU cards. It is worth highlighting that the above definition of S_p differs from the typical expression given in the literature [17]. Finally, the parallel efficiency, E_p, reads:

$$E_p = \frac{S_p}{n_p}. \tag{11}$$

The benchmark simulations were conducted on two different GPU-based architectures: the pre-exascale Leonardo HPC system at CINECA in Italy, equipped with four NVIDIA Ampere™ A100 Tensor Core GPU devices per computing node, each GPU card with 64 GB of HBM2e memory, and the LUMI pre-exascale supercomputer, hosted at CSC in Finland, based on the HPE Cray EX architecture, each compute node equipped with four AMD Instinct™ MI250X GPUs, where each GPU device contains two Graphics Compute Dies (GCDs), with each die providing 64 GB of HBM2e memory, for a total of 128 GB per GPU. The latter architecture yields eight GPU compute units per node, enabling highly parallel and memory-intensive workloads.

In all the benchmarks executed on the pre-exascale Leonardo HPC, the source code was compiled using the Nvidia Compiler, version 24.3, in conjunction with OpenMPI 4.1.6, an open-source implementation of the Message Passing Interface (MPI) standard. Instead, for the benchmark runs on the LUMI pre-exascale HPC, we used the HPE Cray Fortran Compiler, version 17.0.1, along with the Cray-provided MPI implementation based on MPICH, version 8.1.29.

Strong scaling results are presented in Figure 1 in terms of the speedup, S_p and parallel efficiency, E_p, for the single- and double-component systems, evaluated with the MPI decomposition that yields the best performance in terms of GLUPS over the two different GPU-based HPC infrastructures. Note that each AMD Instinct™ MI250X card provides two GPU units, and the performance estimators are evaluated as a function of the number of GPU units. The benchmark data indicate that the impact of parallel communication increases as the computational sub-domain of each GPU device diminishes.

To better illustrate the effect of communication overhead, we also report weak scaling results, where the sub-domain size per MPI process (i.e., per GPU device) remains constant. In Figure 1, we show weak scaling performance using a fixed cubic sub-domain of side 512 lattice points per GPU. This scaling study employs a linear decomposition along the z-axis and demonstrates excellent parallel efficiency, obtaining a peak performance value of 154.94 GLUPS for single-component and 105.48 GLUPS for two-component systems on 64 GPU devices in weak-scaling on NVIDIA Ampere™ A100 GPU-based cluster, versus 92.63 GLUPS and 65.67 GLUPS for the single- and double-component systems, respectively, for the AMD Instinct™ MI250 GPU-based cluster.

5. Results

5.1. Single-Component Homogeneous Isotropic Turbulence: Energy and Pressure Spectra and Single Point Statistics

We evaluate the accuracy of the *accLB* code [31, 24] in performing fully developed turbulent flow simulations by analyzing single-component homogeneous isotropic turbulence (HIT), focusing on energy and pressure spectra along with selected single-point statistics. We simulate three separate cases of increasing Taylor-scale Reynolds numbers, $Re_\lambda = 150$, 230 and 370, on a 512^3 uniform Cartesian grid with unit spacing, $\Delta x = 1$.

The case at $Re_\lambda = 150$ can be effectively considered as a Direct Numerical Simulation (DNS), where the Kolmogorov length scale is approximately equal to the grid spacing, Δx, ensuring adequate resolution of the smallest

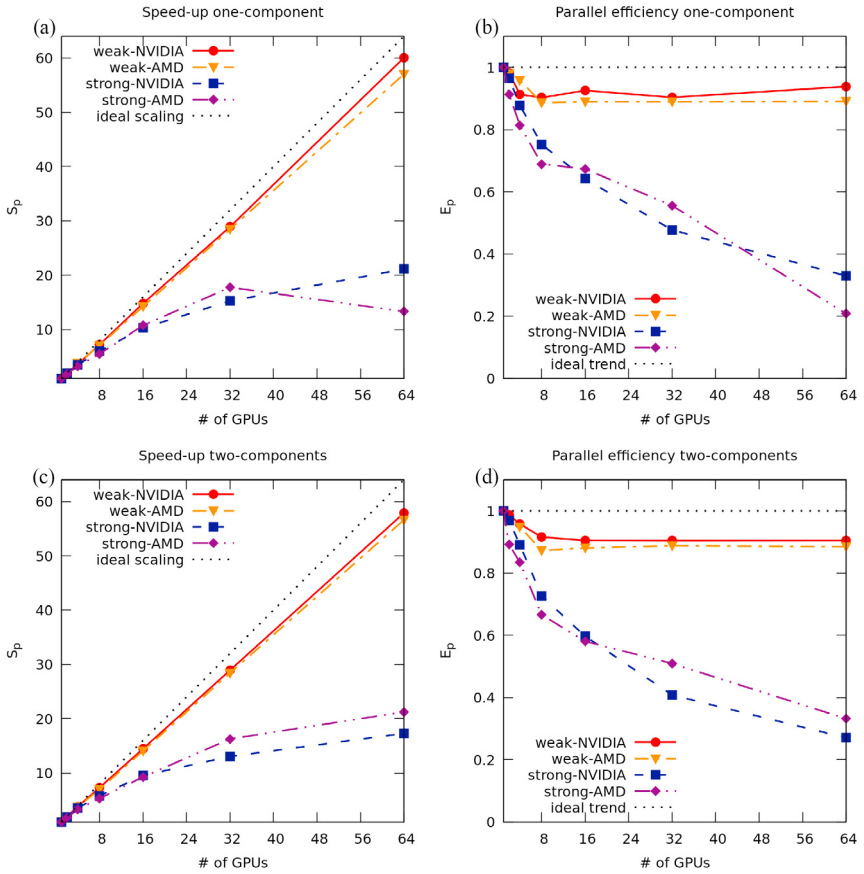


Fig. 1. Panels (a) and (b) show the speed up S_p and parallel efficiency E_p for the single component case measured in strong and weak scaling on the NVIDIA Ampere™ A100 and AMD Instinct™ MI250 GPU-based HPC infrastructures. Panels (c) and (d) report the corresponding data for the two-component benchmark case.

turbulent scales. Conversely, the simulations at $Re_\lambda = 230$ and 370 lie beyond the strict DNS regime, with a Kolmogorov scale of approximately $0.4\Delta x$ and $0.15\Delta x$, respectively.

The homogeneous isotropic turbulent state in all three cases is maintained through large-scale energy injection via the ABC (Arnold–Beltrami–Childress) forcing method. This deterministic, divergence-free forcing scheme introduces a time-independent velocity field that acts as a steady source of energy at the largest scales. Specifically, the forcing takes the form:

$$\mathbf{f}(\mathbf{x}) = A \begin{pmatrix} \sin(k_f z) + \cos(k_f y) \\ \sin(k_f x) + \cos(k_f z) \\ \sin(k_f y) + \cos(k_f x) \end{pmatrix},$$

where A is the forcing amplitude and k_f the forcing wavenumber, set respectively to $2 \cdot 10^{-7}$ (in simulation units) and 4π. The ABC forcing guarantees a sustained turbulent regime by balancing viscous dissipation with energy input, without introducing significant anisotropy, thereby providing an efficient and widely adopted method to achieve statistically stationary HIT in periodic domains [35].

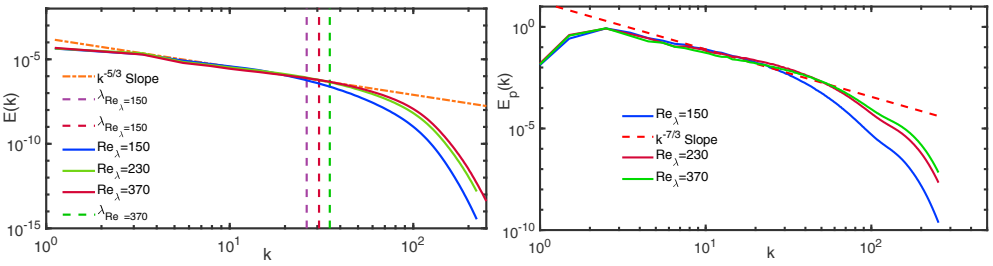


Fig. 2. Panel (a) shows the energy spectra for Re_λ, 150, 230 and 370. The expected $-5/3$ power-law in the inertial range is well captured across a broad range of wavenumbers. Panel (b) displays the corresponding pressure spectra, which follow the $-7/3$ scaling in the inertial range.

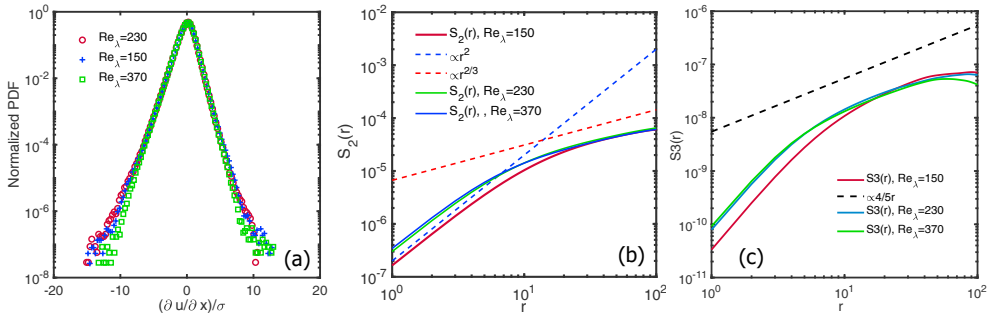


Fig. 3. Panel (a): Normalized probability density function (PDF) of the longitudinal velocity gradients. Panel (b): Second-order structure function, compared with the theoretical r^2 scaling in the dissipation range and $r^{2/3}$ Kolmogorov scaling in the inertial range. Panel (c): Third-order structure function, compared with the theoretical r^2 scaling in the dissipation range and $r^{2/3}$ Kolmogorov scaling in the inertial range.

Figure 2(a) presents the energy spectra for the cases under scrutiny, compared to the classical $-5/3$ slope predicted for the inertial subrange [36, 13]. The observed agreement serves as initial validation of the thread-safe LB solver's capability to reproduce energy transfer mechanisms in single-component turbulent flows.

In figure 2(b), the pressure spectra are plotted and compared to the theoretical $-7/3$ scaling law [44]. In both panels, the simulation at higher Reynolds number exhibits a broader spectrum, indicative of an extended inertial range and enhanced scale separation. As expected, deviations from the $-5/3$ and $-7/3$ power laws emerge at large scales, near the Taylor length scale, which is marked by vertical dashed lines in the plots.

Figure 3 shows single-point statistical features of the homogeneous isotropic turbulent field under examination. Panel (a) displays the probability density function (PDF) of the longitudinal velocity gradient at increasing Taylor Reynolds numbers.

As shown in panel (a), the PDFs deviate significantly from a Gaussian distribution and show negative skewness, $S \approx -0.42$ ($S = \langle(\partial u/\partial x)^3\rangle/\langle(\partial u/\partial x)^2\rangle^{3/2}$), and a flatness coefficient $F \approx 5.25$ ($F = \langle(\partial u/\partial x)^4\rangle/\langle(\partial u/\partial x)^2\rangle^2$), both in line with prior studies [18]. In the physics of HIT, negative skewness points to the presence of long-tails in the statistical distribution of the velocity increments, highlighting the presence of anisotropic effects at small scales, typically caused by persistent vortex stretching, while the elevated flatness signals the presence of intermittent rare-events, i.e., intense fluctuations in the velocity gradients [27]. As also pointed out in [41], while velocity fluctuations in three-dimensional homogeneous and isotropic turbulence have a symmetric distribution, it is known since Kolmogorov's seminal paper [22] that the longitudinal velocity differences are asymmetric.

Panel (b) reports the second-order structure function, $S_2(r) = \langle[u(x_i + r_i) - u(x_i)]^2\rangle$, benchmarked against two theoretical power laws [27]: a r^2 scaling at small r, typical of the dissipation range, and the Kolmogorov $r^{2/3}$ scaling in the inertial range, representative of energy transfer across larger scales. Panel (c) reports the value of the third-order invariant $S_3(r)$ across scales. In classical Kolmogorov theory, $S_3(r)$ follows a linear scaling in the inertial range (i.e., $S_3(r) = -\frac{4}{5}\varepsilon r$, where ε is the energy dissipation rate), implying that the energy transfer rate is scale invariant within the inertial range, i.e., that the turbulence exhibits a form of self-similarity across scales [36, 27, 5]. Both statistics

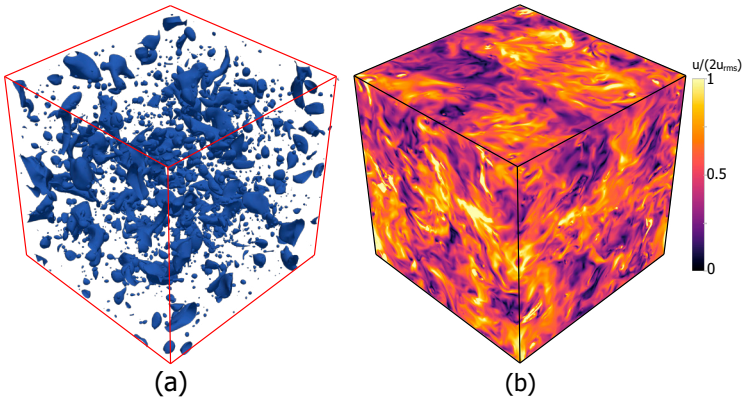


Fig. 4. Instantaneous snapshots of the bubble field panel (a) and velocity field (b) at $Re_\lambda = 114$ and 6% bubble volume fraction. Snapshots are taken at the statistical steady state.

show agreement with the expected theoretical behavior and tend to deviate from the theoretical behavior as r increases to values close to the integral length scale [18].

5.2. *Bubble-Laden Homogeneous Isotropic Turbulence: Effect of the dispersed phase on turbulence modulation*

In this section, we investigate the effect of a dispersed, buoyant gas phase on the turbulence. Specifically, we examine how bubbles influence the energy spectrum at selected wavelengths and explore this behavior across a range of Taylor Reynolds numbers. The bubble volume fraction is fixed at 6% of the total domain, which remains 512^3, as in the single-phase case (see figure 4, reporting two snapshots of the bubble and velocity field taken at the statistical steady state). The computed values of Re_λ for the cases under consideration are 65, 86 and 114. The surface tension coefficient is set to $\gamma = 0.01$ (in lattice units) and is kept constant across all simulations. The turbulent Weber number, defined as $We_T = u_{rms}^2 \lambda \rho_L / \gamma$ is of order one in all simulations (in the formula u_{rms} is the root mean square value of the velocity fluctuations, λ the Taylor length scale and ρ_L the bulk density). The turbulent Froude number, $Fr_T = u_{rms} / \sqrt{gL_I}$, where L_I is the integral length scale, is approximately 2, indicating that while bubbles are influenced by turbulence, their buoyancy remains sufficiently strong to impose a preferential upward motion.

It is worth noting here that, in our simulations, the interface thickness is kept about 5 grid points (implying a Cahn number ($Ca = \delta/L$) to $O(10^{-2})$, being δ the interface thickness and L the size of the domain) and the grid spacing is of the order of η, ensuring that the interface is resolved with a sufficient fidelity relative to the smallest physical scales. Moreover, the focus of our analysis is on statistical observables such as energy spectra and PDFs of velocity increments, which are typically dominated by inertial-range dynamics. These observables are less sensitive to the fine details of the interface smoothing, especially given the moderate volume fraction (6%) and the relatively large separation between interface and inertial-range scales. As shown in Figure 5, the energy spectra reveal a clear transition in scaling behavior with increasing Re_λ, from the classical $-5/3$ Kolmogorov scaling to a steeper -3 power law, typical of 2D turbulence[1]. This trend aligns with experimental observations [28] and theoretical predictions [37]. The emergence of the steeper spectrum is a distinct signature of bubble-induced dissipation at intermediate wavenumbers. Notably, this effect is only observed in buoyant bubble–liquid systems, suggesting that buoyancy plays a critical role in enabling new pathways of energy transfer within homogeneous turbulence. The transition occurs around $k = 10$ (corresponding to $r \approx 35$), approximately one-third of the integral length scale.

Risso [37] analytically demonstrated that the -3 scaling in dispersed multiphase flows arises due to the presence of strong, statistically independent, and uniformly distributed bursts within the turbulent field. These bursts can be interpreted as intermittent events driven by the complex dynamics of bubbles interacting with the carrier phase. As bubbles deform, break up, and re-coalesce, they continuously reorganize the surrounding flow, thus altering the statistical properties of HIT. This behavior is clearly illustrated in Figure 6, which compares the probability density functions (PDFs) of velocity increments in the single-phase and bubble-laden cases. The simulation parameters are otherwise

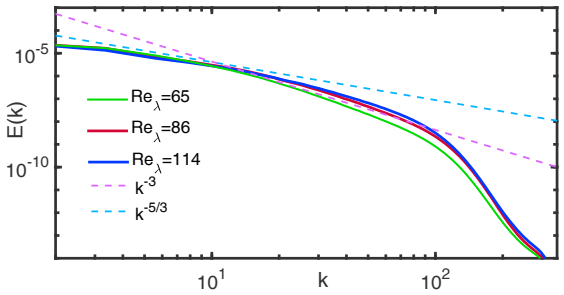


Fig. 5. Energy spectra for bubble-laden HIT at different Re_λ. The transition from $-5/3$ to -3 scaling, observed experimentally in [28], is captured in the intermediate wavenumber range.

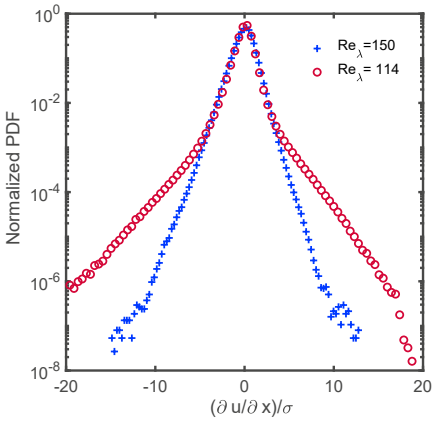


Fig. 6. Comparison of velocity increment PDFs between single-phase and bubble-laden HIT. The enhanced tails in the latter case indicate increased intermittency induced by bubble dynamics. In the figure, circles stand for the bubble-laden HIT while plusses for the single component case

identical, though the resulting Taylor Reynolds numbers differ slightly due to enhanced activity at small scales caused by the dispersed phase [7].

While both PDFs display the expected non-Gaussian features of homogeneous isotropic turbulence, the presence of bubbles significantly amplifies the tails of the distribution. This broadening indicates a higher frequency of intense velocity fluctuations, a hallmark of intermittency. The more pronounced tails in the bubble-laden case point to localized, energetic bursts and rare events arising from bubble dynamics. These findings underscore how dispersed bubbles modify the fine-scale structure of turbulence, intensifying gradients and increasing deviations from the Gaussian statistics typically seen in single-phase flows. The presence of strong intermittent events is further supported by the increase of the flatness factor by a factor 3, from approximately 5 to 16.

6. Conclusions

In this work we have presented accLB, a high-performance, thread-safe Lattice Boltzmann solver specifically tailored for the simulation of multiphase turbulent flows on modern GPU-based supercomputing architectures. By coupling a conservative phase-field formulation of the Allen–Cahn equation with a regularized LBM, accLB have shown to accurately capture the dynamics of fluid interfaces in the presence of turbulence and large density and viscosity contrasts.

The code architecture exploits hybrid parallelism via MPI and OpenACC, offering excellent portability and scalability across heterogeneous hardware platforms. Benchmark tests performed on the EuroHPC pre-exascale systems Leonardo and LUMI demonstrate strong and weak scaling performance up to 64 parallel GPUs, with sustained throughput exceeding 150 GLUPS in large-scale runs.

The physical capabilities of accLB are validated through simulations of single-phase and bubble-laden homogeneous isotropic turbulence (HIT). In the single-phase case, we recover the classical energy and pressure spectra scalings and turbulence invariants across scales. For the bubble-laden scenarios, our simulations capture the emergence of a -3 scaling in the energy spectrum, as predicted by theory and observed in experiments, along with amplified small-scale intermittency. These findings confirm that accLB is a reliable tool to investigate turbulence modulation in complex multiphase systems.

In summary, accLB provides a scalable and accurate platform for high-fidelity turbulence simulations in multiphase flows, enabling the investigation of complex physical phenomena across a broad range of scales with high computational efficiency.

Acknowledgements

A.M. acknowledges fundings from the Italian Government through the PRIN (Progetti di Rilevante Interesse Nazionale) Grant (MOBIOS) ID: 2022N4ZNH3 -CUP: F53C24001000006 and computational support of CINECA through the ISCRA B project DODECA (HP10BUBFIL). M.L. acknowledges the support of the Italian National Group for Mathematical Physics (GNFM-INdAM). M.L. and S.S. acknowledge the support from the European Research Council under the ERCPoC Grant No. 101187935 (LBFAST). D.I. acknowledges financial support from the Research Council of Finland (Grant number: 354620). We acknowledge CSC, Finland for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through CSC, Finland, Benchmark Access mode.

References

[1] R Benzi and S Succi. Two-dimensional turbulence with the lattice boltzmann equation. *Journal of Physics A: Mathematical and General*, 23 (1):L1, 1990.

[2] Matteo Bernardini, Davide Modesti, Francesco Salvadore, and Sergio Pirozzoli. Streams: A high-fidelity accelerated solver for direct numerical simulation of compressible turbulent flows. *Computer Physics Communications*, 263:107906, 2021. ISSN 0010-4655.

[3] Fabio Bonaccorso, Marco Lauricella, Andrea Montessori, Giorgio Amati, Massimo Bernaschi, Filippo Spiga, Adriano Tiribocchi, and Sauro Succi. Lbcuda: A high-performance cuda port of lbsoft for simulation of colloidal systems. *Computer Physics Communications*, 277:108380, 2022. ISSN 0010-4655.

[4] Luca Brandt and Filippo Coletti. Particle-laden turbulence: Progress and perspectives. *Annual Review of Fluid Mechanics*, 54(Volume 54, 2022):159–189, 2022. ISSN 1545-4479.

[5] M Briscolini, P Santangelo, S Succi, and R Benzi. Extended self-similarity in the numerical simulation of three-dimensional homogeneous flows. *Physical Review E*, 50(3):R1745, 1994.

[6] Pedro Costa, Everett Phillips, Luca Brandt, and Massimiliano Fatica. Gpu acceleration of cans for massively-parallel direct numerical simulations of canonical fluid flows. *Computers and Mathematics with Applications*, 81:502–511, 2021. ISSN 0898-1221. Development and Application of Open-source Software for Problems with Numerical PDEs.

[7] Marco Crialesi-Esposito, Sergio Chibbaro, and Luca Brandt. The interaction of droplet dynamics and turbulence cascade. *Communications Physics*, 6(1):5, 2023.

[8] Marco Crialesi-Esposito, Nicolò Scapin, Andreas D. Demou, Marco Edoardo Rosti, Pedro Costa, Filippo Spiga, and Luca Brandt. Flutas: A gpu-accelerated finite difference code for multiphase flows. *Computer Physics Communications*, 284:108602, 2023. ISSN 0010-4655.

[9] Francesco De Vanna. Enhancing performance of high-speed engineering flow computations: The uranos case study. *Procedia Computer Science*, 255:23–32, 2025. ISSN 1877-0509. Proceedings of the Second EuroHPC user day.

[10] Abbas Fakhari, Travis Mitchell, Christopher Leonardi, and Diogo Bolster. Improved locality of the phase-field lattice-boltzmann model for immiscible fluids at high density ratios. *Physical Review E*, 96(5):053301, 2017.

[11] Yongliang Feng, Pierre Boivin, Jérôme Jacob, and Pierre Sagaut. Hybrid recursive regularized lattice boltzmann simulation of humid air with application to meteorological flows. *Physical Review E*, 100(2):023304, 2019.

[12] Yongliang Feng, Pierre Boivin, Jérôme Jacob, and Pierre Sagaut. Hybrid recursive regularized thermal lattice boltzmann model for high subsonic compressible flows. *Journal of Computational Physics*, 394:82–99, 2019.

[13] Uriel Frisch and Andrei Nikolaevich Kolmogorov. *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995.

[14] Wojciech W Grabowski and Lian-Ping Wang. Growth of cloud droplets in a turbulent environment. *Annual review of fluid mechanics*, 45(1): 293–324, 2013.

[15] William D Gropp. Parallel computing and domain decomposition. In *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 349–361. Publ by Soc for Industrial and Applied Mathematics Publ, 1992.

[16] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. Discrete lattice effects on the forcing term in the lattice boltzmann method. *Physical review E*, 65(4):046308, 2002.

[17] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[18] Takashi Ishihara, Toshiyuki Gotoh, and Yukio Kaneda. Study of high–reynolds number isotropic turbulence by direct numerical simulation. *Annual Review of Fluid Mechanics*, 41:165–180, 2009. doi: 10.1146/annurev.fluid.010908.165203.

[19] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted eno schemes. *Journal of Computational Physics*, 126(1):202–228, 1996. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1996.0130. URL https://www.sciencedirect.com/science/article/pii/S0021999196901308.

[20] B. Jähne and H. Haußecker. Air-water gas exchange. *Annual Review of Fluid Mechanics*, 30(Volume 30, 1998):443–468, 1998. ISSN 1545-4479.

[21] Junseok Kim. A continuous surface tension force formulation for diffuse-interface models. *Journal of computational physics*, 204(2):784–804, 2005.

[22] Andrei Nikolaevich Kolmogorov. Dissipation of energy in the locally isotropic turbulence. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 434(1890):15–17, 1991.

[23] Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. The lattice boltzmann method. *Springer International Publishing*, 10(978-3):4–15, 2017.

[24] Marco Lauricella, Adriano Tiribocchi, Sauro Succi, Luca Brandt, Aritra Mukherjee, Michele La Rocca, and Andrea Montessori. Thread-safe multiphase lattice boltzmann model for droplet and bubble dynamics at high density and viscosity contrasts. *arXiv preprint arXiv:2501.00846*, 2025.

[25] Giandomenico Lupo, Peter Wellens, and Pedro Costa. Cans-fizzy: A gpu-accelerated finite difference solver for turbulent two-phase flows. *arXiv preprint arXiv:2502.04189*, 2025.

[26] Orestis Malaspinas. Increasing stability and accuracy of the lattice boltzmann scheme: recursivity and regularization. *arXiv preprint arXiv:1505.06900*, 2015.

[27] WD McComb. Theory of turbulence. *Reports on Progress in Physics*, 58(10):1117, 1995.

[28] Julian Martinez Mercado, Daniel Chehata Gomez, Dennis Van Gils, Chao Sun, and Detlef Lohse. On bubble clustering and energy spectra in pseudo-turbulence. *Journal of fluid mechanics*, 650:287–306, 2010.

[29] Andrea Montessori and Giacomo Falcucci. *Lattice Boltzmann modeling of complex flows for engineering applications*. Morgan and Claypool Publishers, 2018.

[30] Andrea Montessori, Marco Lauricella, Adriano Tiribocchi, Mihir Durve, Michele La Rocca, Giorgio Amati, Fabio Bonaccorso, and Sauro Succi. Thread-safe lattice boltzmann for high-performance computing on gpus. *Journal of Computational Science*, 74:102165, 2023.

[31] Andrea Montessori, Michele La Rocca, Giorgio Amati, Marco Lauricella, Adriano Tiribocchi, and Sauro Succi. High-order thread-safe lattice boltzmann model for high performance computing turbulent flow simulations. *Physics of Fluids*, 36(3), 2024.

[32] Andrea Montessori, Luiz A Hegele, and Marco Lauricella. A thread-safe lattice boltzmann model for multicomponent turbulent jet simulations. *AIAA Journal*, 63(3):1005–1012, 2025.

[33] Francesca Pelusi, Matteo Lulli, Mauro Sbragaglia, and Massimo Bernaschi. Tlbfind: a thermal lattice boltzmann code for concentrated emulsions with finite-size droplets. *Computer Physics Communications*, 273:108259, 2022.

[34] Francesca Pelusi, Stefano Ascione, Mauro Sbragaglia, and Massimo Bernaschi. Analysis of the heat transfer fluctuations in the rayleigh–bénard convection of concentrated emulsions with finite-size droplets. *Soft Matter*, 19(37):7192–7201, 2023.

[35] O. Podvigina and A. Pouquet. On the non-linear stability of the 1:1:1 abc flow. *Physica D: Nonlinear Phenomena*, 75(4):471–508, 1994. ISSN 0167-2789. doi: https://doi.org/10.1016/0167-2789(94)00031-X. URL https://www.sciencedirect.com/science/article/pii/016727899400031X.

[36] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[37] Frédéric Risso. Theoretical model for k- 3 spectra in dispersed multiphase flows. *Physics of fluids*, 23(1), 2011.

[38] Alessio Roccon. A gpu-ready pseudo-spectral method for direct numerical simulations of multiphase turbulence. *Procedia Computer Science*, 240:17–30, 2024. ISSN 1877-0509. Proceedings of the First EuroHPC user day.

[39] Gaetano Sardina, Francesco Picano, Luca Brandt, and Rodrigo Caballero. Continuous growth of droplet size variance due to condensation in turbulent clouds. *Phys. Rev. Lett.*, 115:184501, Oct 2015.

[40] Laurier L. Schramm, Elaine N. Stasiuk, and D. Gerrard Marangoni. Surfactants and their applications. *Annu. Rep. Prog. Chem., Sect. C: Phys. Chem.*, 99:3–48, 2003.

[41] Katepalli R Sreenivasan, Kartik P Iyer, and Ashvin Vinodh. Asymmetry of velocity increments in turbulence. *Physical Review Research*, 4(4): L042002, 2022.

[42] S. Succi. *The Lattice Boltzmann equation: For complex states of flowing matter*. Oxford University Press, 2018.

[43] S. Succi, G. Amati, M. Bernaschi, G. Falcucci, M. Lauricella, and A. Montessori. Towards exascale lattice boltzmann computing. *Computer and Fluids*, 181:107–115, 2019. doi: https://doi.org/10.1016/j.compfluid.2019.01.005.

[44] Sipei Zhao, Eva Cheng, Xiaojun Qiu, Ian Burnett, and Jacob Chia chun Liu. Pressure spectra in turbulent flows in the inertial and the dissipation ranges. *The Journal of the Acoustical Society of America*, 140:4178, 2016. doi: 10.1121/1.4968881.

[45] Xiaojue Zhu, Everett Phillips, Vamsi Spandan, John Donners, Gregory Ruetsch, Joshua Romero, Rodolfo Ostilla-Mónico, Yantao Yang, Detlef Lohse, Roberto Verzicco, Massimiliano Fatica, and Richard J.A.M. Stevens. Afid-gpu: A versatile navier–stokes solver for wall-bounded turbulent flows on gpu clusters. *Computer Physics Communications*, 229:199–210, 2018. ISSN 0010-4655.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 52–61

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Multi-GPU porting of a phase-change cascaded lattice Boltzmann method for three-dimensional pool boiling simulations

Alessandro Gabbana[a,b,*], Linlin Fei[c], Xander M. de Wit[d], Ziqi Wang[d], Daniel Livescu[a], Federico Toschi[d,e]

[a]*Computational Physics and Methods Group (CCS-2), Los Alamos National Laboratory, Los Alamos, 87545 New Mexico, USA*
[b]*Center for Nonlinear Studies (CNLS), Los Alamos National Laboratory, Los Alamos, 87545 New Mexico, USA*
[c]*Key Laboratory of Thermo-Fluid Science and Engineering of Ministry of Education, School of Energy and Power Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China*
[d]*Department of Applied Physics and Science Education, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands*
[e]*CNR-IAC, I-00185 Rome, Italy*

Abstract

The Lattice Boltzmann method (LBM) has proven effective in simulating phase-change phenomena, such as melting, solidification, evaporation, and boiling. In this work, we develop a highly parallelized multi-GPU implementation of LBM for three-dimensional pool boiling simulations. The code is based on the OpenACC programming model, which enables the code to be deployed efficiently on multi-core CPUs, GPUs, and potentially other accelerators, without the need for architecture-specific rewrites. To support large-scale simulations, the domain is decomposed and distributed across multiple compute nodes using MPI. We demonstrate that the code exhibits excellent scaling properties, with ideal strong-scaling running with up to 256 GPUs on the MareNostrum5 cluster.

Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day

Keywords: Pool Boiling; Lattice Boltzmann Method; OpenACC; Multi-GPU;

1. Introduction

Boiling is a highly efficient heat transfer mechanism with widespread applications, from everyday uses such as cooking to critical industrial systems such as nuclear reactors, heat exchangers, and electronic cooling devices. Despite its ubiquity, the boiling process remains a subject of intense research because of its inherently multiscale and multiphysics nature, which involves complex interactions between fluid dynamics, phase change, and heat transfer [1].

Among the different boiling configurations, pool boiling, where buoyancy is the sole driving force for fluid motion, offers a relatively simple setup to study the fundamental mechanisms of bubble nucleation, growth, departure, and

* Corresponding author.
E-mail address: agabbana@lanl.gov

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day
10.1016/j.procs.2025.08.232

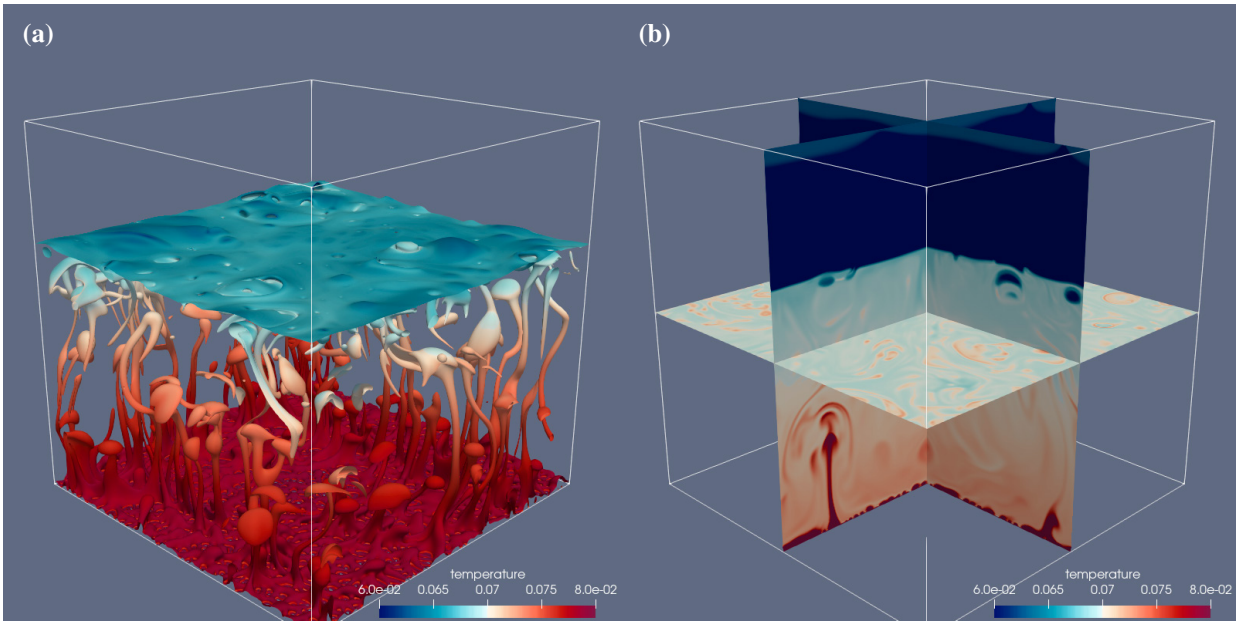


Fig. 1. Snapshot from a pool boiling simulation on a 1024^3 domain, in the nucleate boiling regime. (a) Iso-surface representation of the liquid-vapor interface, colored by the local temperature field. (b) Cross-sectional slices of the temperature field.

coalescence. Understanding these phenomena is essential to improve heat transfer efficiency and predict performance limits, such as the critical heat flux (CHF), in engineering systems. However, accurately capturing these dynamics requires high-fidelity numerical simulations that resolve both the microscale vapor-liquid interfaces and the macroscale thermal convection.

To address this challenge, numerical methods like the lattice Boltzmann method (LBM) have emerged as promising tools due to their algorithmic simplicity, scalability, and ability to naturally handle complex boundary conditions. In particular, the pseudopotential multiphase LBM, enhanced by thermal coupling via finite-difference solvers and non-ideal equations of state, has proven effective in simulating phase-change phenomena, including nucleate boiling and transitions to film boiling. The model developed by Fei et al. [2], which extends the hybrid LBM to fully three-dimensional domains using the cascaded collision operator for improved numerical stability, has successfully reproduced the entire boiling curve and boiling regimes with hundreds of interacting bubbles.

Despite these advances, most existing numerical studies on boiling remain limited to two-dimensional simulations or coarse three-dimensional models, which significantly restrict their applicability to real-world scenarios. Furthermore, the computational cost of resolving a fully 3D boiling process at realistic scales has, until recently, remained prohibitive [3].

In this work, we take a major step forward by developing a highly parallelized multi-GPU implementation of LBM for three-dimensional pool boiling simulations. By leveraging massively parallel GPU architectures, our code enables simulations involving millions of grid points, offering the possibility of simulations with unprecedented fidelity and throughput, making it possible to quantitatively study bubble nucleation, bringing us closer to quantitatively matching experimental observations and validating theoretical models. A qualitative example is shown in Figure 1, where we provide the snapshot from a simulation in the nucleate boiling regime on a 1024^3 domain.

The remainder of this paper is organized as follows. In Section 2 we provide a short overview on the numerical method. Section 3 outlines the key aspects of porting the LBM code to multi-GPU clusters. Performance results and scalability analysis are presented in Section 4. Finally, Section 5 offers concluding remarks and discusses possible future developments.

2. Numerical Method

In this section we provide details on the numerical methods employed for the simulation of pool boiling. We couple two methods, with a hybrid cascaded lattice Boltzmann model (CLBM) addressing the liquid-vapor phase-change process, and a finite difference method for the evolution of the temperature field, and with the two components coupled via a non-ideal equation of state [2].

2.1. Solution of the flow field

The lattice Boltzmann method (LBM) has proven to be a highly efficient and flexible mesoscopic approach for simulating a wide variety of fluid flows [4]. Unlike traditional Navier-Stokes solvers which explicitly discretize the macroscopic equations, LBM is a mesoscopic approach that models the evolution of particle distribution functions over a discrete lattice. At its core, LBM solves a discretized form of the Boltzmann equation:

$$f_i(\mathbf{x} + \mathbf{e}_i\delta t, t + \delta t) - f_i(\mathbf{x}, t) = \Omega(f_i) + F_i, \tag{1}$$

where f_i denotes the particle distribution function along the i-th discrete velocity direction $\mathbf{e}_i$, Ω is the collision operator, F_i is a source term accounting for internal and external forces, and δt is the time-step size. The left-hand side represents the streaming step, which moves information from one lattice node to its neighbors, while the right-hand side captures local changes in the particle distributions due to collisions and forcing effects.

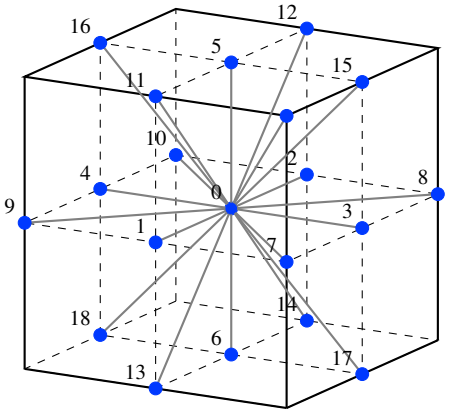


Fig. 2. D3Q19 velocity stencil used for the discretization of the velocity space in the lattice Boltzmann method.

We use the D3Q19 model (see Fig. 2) for the discretization of the continuum velocity space, employing a set of 19 discrete velocities, defined as:

$$\mathbf{e}_i = [|\, e_{ix} \,\rangle, |\, e_{iy} \,\rangle, |\, e_{iz} \,\rangle], \quad i = 0, \ldots, 18,$$

with the specific components given by:

$$\begin{aligned} |\, e_{ix} \,\rangle &= [0, 1, -1, 0, 0, 0, 0, 1, -1, 1, -1, 1, -1, 1, -1, 0, 0, 0, 0]^\top, \\ |\, e_{iy} \,\rangle &= [0, 0, 0, 1, -1, 0, 0, 1, 1, -1, -1, 0, 0, 0, 0, 1, -1, 1, -1]^\top, \\ |\, e_{iz} \,\rangle &= [0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0, 1, 1, -1, -1, 1, 1, -1, -1]^\top. \end{aligned} \tag{2}$$

Here, $|\cdot\rangle$ denotes a 19-dimensional column vector. The lattice speed is defined as $c = \delta x/\delta t = 1$, and the lattice sound speed is $c_s = 1/\sqrt{3}$, with $\delta x = \delta t = 1$.

In this study, we employ a central-moment-based LBM, which offers enhanced numerical stability and the possibility to independently tune the relaxation rate of the different transport parameters. In CLBM the post-collision state is defined as

$$f_i^* = \Omega(f_i) = f_i^{\mathrm{eq}} + \left(1 - \mathbf{M^{-1}N^{-1}SNM}\right)\left(f_i - f_i^{\mathrm{eq}}\right) \tag{3}$$

where f_i^{eq} is the (discrete) local equilibrium distribution, $\mathbf{M}$ is the transformation matrix mapping the particle distributions to moment space, $\mathbf{N}$ is the transformation matrix shifting from raw moments to central moments, and $\mathbf{S}$ is the relaxation matrix, dictating the relaxation rate of the different moments. Therefore, in the CLBM framework, the collision operator is defined in terms of a set of raw and central moments of the particle distribution functions f_i, respectively

$$k_{mnp} = \left\langle f_i \, e_{ix}^m e_{iy}^n e_{iz}^p \right\rangle, \quad \tilde{k}_{mnp} = \left\langle f_i \, (e_{ix} - u_x)^m (e_{iy} - u_y)^n (e_{iz} - u_z)^p \right\rangle, \tag{4}$$

where m, n, p are non-negative integers and u_x, u_y, u_z are the components of the macroscopic velocity vector $\mathbf{u}$.

Following the work of Fei et al. [5], we consider the following non-orthogonal set of central moments:

$$\left|\tilde{T}_i\right\rangle = [\tilde{k}_{000}, \tilde{k}_{100}, \tilde{k}_{010}, \tilde{k}_{001}, \tilde{k}_{110}, \tilde{k}_{101}, \tilde{k}_{011}, \tilde{k}_{200}, \tilde{k}_{020}, \tilde{k}_{002}, \tilde{k}_{120}, \tilde{k}_{102}, \tilde{k}_{210}, \tilde{k}_{201}, \tilde{k}_{012}, \tilde{k}_{021}, \tilde{k}_{220}, \tilde{k}_{202}, \tilde{k}_{022}]^\top, \tag{5}$$

with

$$|T_i\rangle = \mathbf{M}\,|f_i\rangle = [T_0, T_1, ..., T_{18}]^\top, \quad \left|\tilde{T}_i\right\rangle = \mathbf{N}\,|T_i\rangle = \mathbf{NM}\,|f_i\rangle = [\tilde{T}_0, \tilde{T}_1, ..., \tilde{T}_{18}]^\top. \tag{6}$$

This choice leads to a sparse transformation matrix [5], which allows for enhanced computational efficiency.

The collision process can then be expressed as the independent relaxation of each central moment towards its equilibrium counterpart as:

$$\left|\tilde{T}_i^*\right\rangle = \left|\tilde{T}_i\right\rangle - \mathbf{S}\left(\left|\tilde{T}_i\right\rangle - \left|\tilde{T}_i^{eq}\right\rangle\right) + (\mathbf{I} - \mathbf{S}/2)\delta t\,|C_i\rangle\,, \tag{7}$$

where $|C_i\rangle$ represents the forcing terms in moment space.

The choice for the set of moments in Eq. 5 implies the following block-diagonal form for the relaxation matrix:

$$\mathbf{S} = \text{diag}\left\{\text{s}_0, \text{s}_1, \text{s}_1, \text{s}_1, \text{s}_\nu, \text{s}_\nu, \text{s}_\nu, \begin{bmatrix} s_+ & s_- & s_- \\ s_- & s_+ & s_- \\ s_- & s_- & s_+ \end{bmatrix}, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_4, \text{s}_4, \text{s}_4\right\}, \tag{8}$$

with $s_+ = (s_{2b} + 2s_2)/3$, $s_- = (s_{2b} - s_2)/3$, enabling independent control of normal stress components and bulk viscosity [5].

The equilibrium central moments are matched to those of the continuous Maxwell-Boltzmann distribution:

$$\left|\tilde{T}_i^{eq}\right\rangle = [\rho, 0, 0, 0, 0, 0, 0, \rho c_s^2, \rho c_s^2, \rho c_s^2, 0, 0, 0, 0, 0, 0, \rho c_s^4, \rho c_s^4, \rho c_s^4]^\top. \tag{9}$$

The forcing term in moment space is:

$$|C_i\rangle = [0, F_x, F_y, F_z, 0, 0, 0, 0, 0, 0, F_x c_s^2, F_x c_s^2, F_y c_s^2, F_z c_s^2, F_y c_s^2, F_z c_s^2, 0, 0, 0]^\top. \tag{10}$$

After the collision step, the post-collision distribution functions are reconstructed by:

$$\left|f_i^*\right\rangle = \mathbf{M}^{-1}\mathbf{N}^{-1}\left|\tilde{T}_i^*\right\rangle,$$

and propagated via the streaming step:

$$f_i(\mathbf{x} + \mathbf{e}_i\delta t, t + \delta t) = f_i^*(\mathbf{x}, t). \tag{11}$$

The macroscopic variables are recovered as:

$$\rho = \sum_i f_i, \quad \rho\mathbf{u} = \sum_i f_i \mathbf{e}_i + \frac{\delta t}{2}\mathbf{F}. \tag{12}$$

The kinematic viscosity ν and bulk viscosity ξ are related to the relaxation parameters via:

$$\nu = \left(\frac{1}{s_2} - \frac{1}{2}\right)c_s^2\delta t, \quad \xi = \frac{2}{3}\left(\frac{1}{s_{2b}} - \frac{1}{2}\right)c_s^2\delta t. \tag{13}$$

In order to incorporate phase-change phenomena, the pseudopotential model is adopted [6, 7], with the interaction force:

$$\mathbf{F}_{\text{int}} = -G\psi(\mathbf{x}) \sum_i w(|\mathbf{e}_i|^2)\psi(\mathbf{x} + \mathbf{e}_i\delta t)\mathbf{e}_i. \tag{14}$$

The pseudopotential ψ is defined by:

$$\psi = \sqrt{\frac{2(p_{\text{EOS}} - \rho c_s^2)}{Gc^2}}, \tag{15}$$

where $G = -1$ ensures the positivity of the expression under the square root.

To address thermodynamic inconsistency, Li et al. [8, 9] proposed modifying the forcing term:

$$|C_i\rangle = [0, F_x, F_y, F_z, 0, 0, 0, \eta, \eta, \eta, F_x c_s^2, F_x c_s^2, F_y c_s^2, F_z c_s^2, F_y c_s^2, F_z c_s^2, 0, 0, 0]^\top, \tag{16}$$

where:

$$\eta = \frac{2\sigma |\mathbf{F}_{\text{int}}|^2}{\psi^2 (s_e^{-1} - 0.5)\delta t}, \tag{17}$$

with σ used for tuning η. By applying a Chapman-Enskog analysis, the following macroscopic equations are recovered in the low-Mach limit:

$$\begin{aligned} &\partial_t \rho + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) = 0, \\ &\partial_t (\rho \mathbf{u}) + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}\mathbf{u}) = -\boldsymbol{\nabla}(\rho c_s^2) + \boldsymbol{\nabla} \cdot \left[\rho \nu (\boldsymbol{\nabla}\mathbf{u} + (\boldsymbol{\nabla}\mathbf{u})^\top) - \tfrac{2}{3}\rho\nu(\boldsymbol{\nabla}\cdot\mathbf{u})\mathbf{I}\right] + \boldsymbol{\nabla}\left[\rho\xi(\boldsymbol{\nabla}\cdot\mathbf{u})\right] + \mathbf{F} - 2G^2c^4\sigma\boldsymbol{\nabla}\cdot\left(|\boldsymbol{\nabla}\psi|^2\mathbf{I}\right) \end{aligned} \tag{18}$$

where the final term arises from the thermodynamic correction via η. Note that for $\sigma = 0$, the standard Navier-Stokes equations are recovered.

2.2. Solution of the Temperature Field

The temperature evolution in the liquid-vapor phase-change process is governed by the following equation [10]:

$$\frac{\partial T}{\partial t} = -\mathbf{u} \cdot \boldsymbol{\nabla} T + \boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T) - \frac{T}{\rho c_v} \left(\frac{\partial p_{\text{EOS}}}{\partial T}\right)_\rho \boldsymbol{\nabla} \cdot \mathbf{u}, \tag{19}$$

where T is the temperature, $\mathbf{u}$ is the fluid velocity, λ is the thermal conductivity, ρ is the density, and c_v is the specific heat at constant volume. The term involving $\partial p_{\text{EOS}}/\partial T$ accounts for the thermodynamic coupling due to the non-ideal equation of state.

To numerically solve Eq. 19, spatial derivatives must be discretized appropriately. In this work, a central difference scheme based on lattice directions is employed to approximate both the gradient and Laplacian operators [11]. The discretization is given as:

$$\begin{aligned} \frac{\partial \phi}{\partial x_\alpha} &= \frac{1}{c_s^2 \delta t} \sum_i \omega\left(|\mathbf{e}_i|^2\right) e_{i\alpha}\, \phi\left(\mathbf{x} + \mathbf{e}_i \delta t\right), \\ \frac{\partial^2 \phi}{\partial x_\alpha \partial x_\alpha} &= \frac{2}{c_s^2 \delta t} \sum_i \omega\left(|\mathbf{e}_i|^2\right) \left[\phi\left(\mathbf{x} + \mathbf{e}_i \delta t\right) - \phi(\mathbf{x})\right], \end{aligned} \tag{20}$$

where $\alpha \in \{x, y, z\}$ denotes the spatial coordinate direction, ϕ is a scalar physical quantity (e.g., temperature), $\mathbf{e}_i$ are the discrete lattice velocities, and ω is a weight function associated with the magnitude of $\mathbf{e}_i$.

In order to conserve the heat flux, particularly across the liquid-vapor interface where sharp gradients may occur, the diffusion term $\boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T)$ is discretized using a finite volume method [12]. This approach provides better accuracy and stability when handling discontinuities and steep gradients, which are common in phase-change phenomena.

For temporal discretization, the right-hand side of Eq. 19 is denoted as $K(T)$, representing all spatially discretized contributions to the temperature evolution. The time integration is carried out using a fourth-order Runge-Kutta (RK4) method:

$$\begin{aligned} T^{t+\delta t} &= T^t + \tfrac{\delta t}{6}(h_1 + h_2 + h_3 + h_4), \\ h_1 &= K(T^t), \\ h_2 &= K\left(T^t + \tfrac{\delta t}{2} h_1\right), \\ h_3 &= K\left(T^t + \tfrac{\delta t}{2} h_2\right), \\ h_4 &= K\left(T^t + \delta t\, h_3\right). \end{aligned} \tag{21}$$

2.3. Non-ideal equation of state

The pseudopotential CLBM described in Section 2.1 is coupled with the finite-difference temperature solver introduced in Section 2.2 through a non-ideal equation of state. Specifically, temperature variations affect the local thermodynamic pressure p_{EOS}, which in turn modifies the interparticle interaction force in the pseudopotential CLBM via Eqs. 14 and 15.

To model the non-ideal thermodynamic behavior of the fluid, various equations of state—such as the van der Waals, Carnahan-Starling, or Peng-Robinson EOS—can be incorporated into the framework through the square-root form of the pseudopotential defined in Eq. 15. Following a previous study [2], the Peng-Robinson EOS is employed:

$$p_{\text{EOS}} = \frac{\rho RT}{1 - b\rho} - \frac{a\varphi(T)\rho^2}{1 + 2b\rho - b^2\rho^2}, \tag{22}$$

where the temperature-dependent function $\varphi(T)$ is defined as:

$$\varphi(T) = \left[1 + \left(0.37464 + 1.54226\varpi - 0.26992\varpi^2\right)\left(1 - \sqrt{T/T_c}\right)\right]^2,$$

and the parameters $a = 2/49$, $b = 2/21$ have been established following the analysis in Refs. [13, 9], with $\varpi = 0.344$ and $R = 1$. The corresponding critical temperature and pressure are computed as $T_c = 0.07292$ and $p_c = 0.05957$, respectively.

Furthermore, in the simulation of pool boiling phenomena, the motion of vapor bubbles and surrounding liquid is significantly influenced by buoyancy forces arising from density differences. The buoyancy force is modeled as:

$$\mathbf{F}_b = -(\rho - \bar{\rho})g\hat{\mathbf{k}}, \tag{23}$$

where $\bar{\rho}$ is the average density in the computational domain, g is the gravitational acceleration, and $\hat{\mathbf{k}}$ is the unit vector in the vertical (gravity) direction.

As a result, the total body force acting on the system is given by the sum of the interparticle interaction force and the buoyancy force:

$$\mathbf{F} = \mathbf{F}_{\text{int}} + \mathbf{F}_b.$$

This coupling strategy ensures that both temperature-induced pressure variations and gravitational effects are consistently incorporated into the fluid dynamics, enabling accurate simulation of liquid-vapor phase-change processes such as boiling and condensation.

3. Implementation

To achieve simultaneously high performance and portability across heterogeneous computing architectures, our LBM code is implemented using the OpenACC programming model. OpenACC enables incremental parallelization of existing C/C++ and Fortran codebases and supports a broad range of accelerators and CPUs. The primary goal of this implementation is to ensure that the same codebase can be deployed efficiently on NVIDIA GPUs, multi-core CPUs, and potentially other accelerators, without requiring architecture-specific modifications.

The core compute routines, such as streaming, collision, macroscopic moment computation, boundary conditions and the Runge-Kutta solver for the temperature fields, are offloaded to the accelerator using OpenACC directives. Loop-level parallelism is expressed through the kernels construct, allowing the compiler to generate optimized code for the target architecture.

To support parallel updates of all grid points during both streaming and collision operations, we store two copies of the lattice for the particle distribution functions f_i, following the so-called A-A pattern. The lattice data is transferred to the accelerator memory at the beginning of the time-stepping loop, and an OpenACC data region is used to minimize communication between the host and the device, restricting data transfers to essential operations such as disk writes and MPI communications.

The code adopts a Structure of Arrays (SoA) data layout, which aligns with memory coalescing requirements on GPUs and supports efficient vectorization on CPUs. In this layout, the particle distribution functions are stored with

spatial indices contiguously in memory. This organization enhances memory bandwidth utilization, which is a key performance factor in LBM algorithms due to their memory-bound nature. Although more sophisticated data structures for LBM have been proposed, offering performance benefits on architectures with large cache hierarchies [14], they often introduce additional complexity. For the sake of simplicity and maintainability, we opt not to adopt them in the current implementation.

To support large-scale simulations, the computational domain is decomposed and distributed across multiple compute nodes using MPI. For simplicity, we adopt a one-dimensional domain decomposition, where each MPI rank handles a subdomain. Halo regions (ghost layers) are exchanged at each time step to maintain data consistency across subdomain boundaries. The code is designed for multi-node execution, with one MPI process assigned to each GPU when running on GPU clusters.

Inter-process communication is implemented using non-blocking MPI routines, enabling the overlap of computation and communication. Specifically, the interior nodes of each subdomain are updated while halo data is exchanged asynchronously. This overlap is explicitly managed by partitioning the domain into interior and halo regions and launching separate OpenACC kernels for each subdomain. When possible, the kernels are executed on different CUDA streams to further reduce potential overheads introduced by small kernels reserved to the preparation of buffers used in MPI communications and kernels operating on the boundary layers.

We leverage a CUDA-aware MPI library, which allows communication buffers to reference device memory directly. This enables efficient pipelining of data movement between host and device memory, reducing overhead and improving scalability, and also enables seamless use of NVLink for direct GPU-to-GPU data transfers between devices on the same node.

4. Results

In this section we provide a brief overview of the performances of the code.

All results are obtained using a realistic physical configuration that replicates the pool boiling setup illustrated in Fig. 1. We perform three-dimensional simulations where the bottom plate is maintained at a fixed temperature, with a small perturbation applied at $t = 0$ to trigger bubble nucleation. At the top boundary, we impose a constant pressure condition along with a Neumann boundary condition for velocity, enforcing that the normal derivative of the velocity field is set to zero. Periodic boundary conditions are applied in the remaining directions. The initial condition consists of a domain half-filled with liquid and half with vapor. The flow field is solved using the D3Q19 LBM (cf. Fig. 2), while the temperature field is evolved using a fourth-order Runge-Kutta method.

We begin by analyzing the performance obtained on a single GPU. As a representative case, we consider the performance of the two main computational kernels: streaming and collide. The streaming kernel is responsible for moving pseudo-particles between neighboring grid points and is thus entirely memory-bound. On a single NVIDIA A100 GPU, this kernel achieves a bandwidth of 1245.31 GB/s, which corresponds to approximately 62% of the theoretical peak. In contrast, the collide kernel is the most compute-intensive component of the code. On the same A100 GPU, it reaches 2745.43 GFLOP/s, about 28% of the theoretical peak performance, comparing well against results of previous LBM implementations from the literature [15, 16]. These figures nearly double when running on a single H100 GPU at the MareNostrum 5 supercomputer (BSC), achieving 2659.13 GB/s for the streaming kernel and 5616.93 GFLOP/s for the collide kernel. These figures show that the code allows to extract a significant fraction of the performance peak on some of the latest GPU cards.

Moving now to the evaluation of the multi-GPU implementation, Fig. 3a reports on the strong scalability and corresponding parallel efficiency achieved on the MareNostrum 5 cluster.

We consider three representative domain sizes: a small domain of size $1536 \times 512 \times 512$, a medium domain of $3072 \times 1024 \times 1024$, and a large domain of $6144 \times 2048 \times 2048$. In the strong scalability test, the domain is partitioned along the x-dimension as L_x/p, where p is the number of GPUs. We evaluate performance for $p = 4, 8, \ldots, 512$ GPUs.

We compare two code variants: one where communication and computation are overlapped, and another where they are executed synchronously. The plot clearly shows the benefit of overlapping, which brings performance close to ideal scaling up to 256 GPUs. At $p = 512$, execution times of the two variants converge, suggesting that the execution time becomes dominated by the cost of MPI communications. Since this overhead is roughly constant

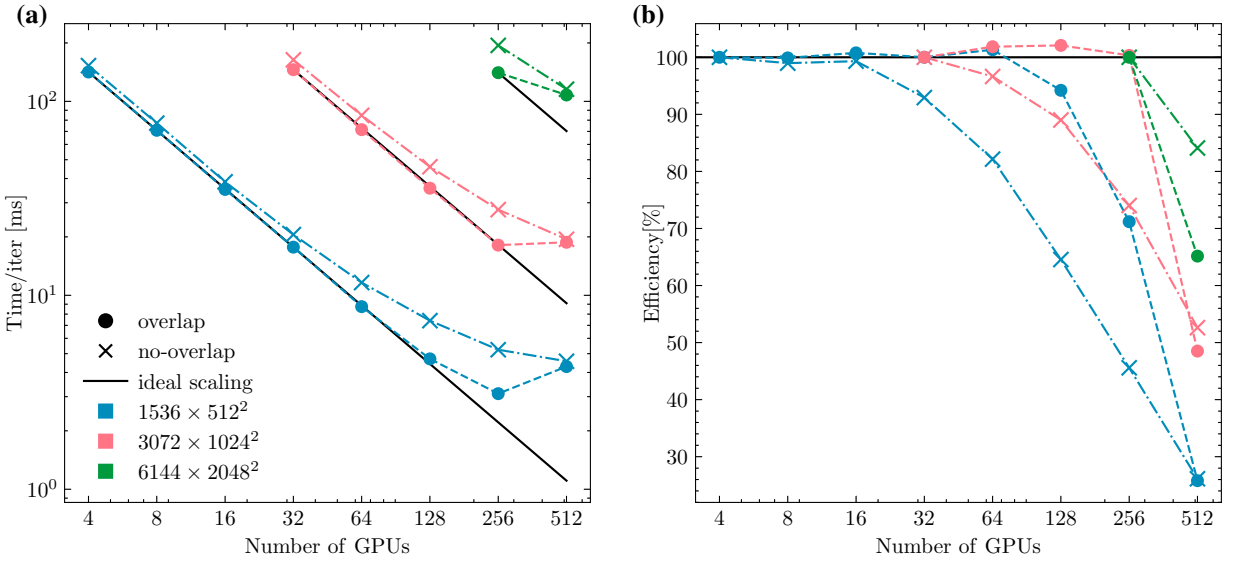


Fig. 3. (a) Strong scalability results for three domain sizes on the MareNostrum 5 cluster, using up to 512 GPUs. Two code variants are compared: with and without overlapping of communication and computation. (b) Parallel efficiency corresponding to the strong scaling test, with baseline GPU counts of 4, 32, and 256 for the small, medium, and large domains, respectively.

with respect to p under 1D partitioning, scalability plateaus. Higher-dimensional domain decompositions could help alleviate this bottleneck, albeit at the cost of added code complexity.

In Fig. 3b, we show the corresponding parallel efficiency, defined as:

$$E(p) = \frac{T(b)}{T(p)} \frac{p}{b} \tag{24}$$

where $T(p)$ is the execution time using p GPUs, and b is the minimum number of GPUs required to fit a given problem size in memory (4 for the small domain, 32 for the medium, 256 for the large).

Remarkably, for the medium-size domain, efficiency remains close to 100% up to 256 GPUs. Minor super-linear speedups can be observed and are attributed to runtime kernel scheduling on CUDA streams. These effects are difficult to predict *a priori* and depend on available GPU resources. When the workload per GPU decreases (at higher p), the potential for concurrent kernel execution increases, which may yield slight performance boosts.

5. Conclusions and future work

We have presented a multi-GPU implementation of a Lattice Boltzmann Method (LBM) code designed for large-scale simulations of pool boiling.

The code exhibits excellent strong scaling, reaching near-ideal performance up to 256 GPUs on representative three-dimensional domains. This opens up the possibility of performing high-resolution numerical studies of the complex dynamics of boiling at scales that were previously computationally prohibitive.

Extended details on the implementation, validation strategy, and performance analysis will be provided in an extended version of this work. Moreover we plan to use the code for studying in detail the transition between boiling regimes as a function of wall superheat, the spatiotemporal evolution and distribution of bubble footprint areas, and the influence of geometrical and thermophysical properties on boiling curves. Work along these lines is in progress.

Acknowledgments

We acknowledge HPC-Europe computational facilities and allocation on the MareNostrum 5 supercomputer (BSC, Spain) through grant EHPC-EXT-2023E02-035, and the Karolina cluster (IT4I, Czech Republic) through grant

EHPC-DEV-2023D04-032. A.G. gratefully acknowledges the support of the U.S. Department of Energy through the LANL/LDRD Program under project number 20240740PRD1 and the Center for Non-Linear Studies for this work.

Appendix A. Scaling performance Data

For completeness, in Table A.1 we report the data presented in Fig. 3.

Table A.1. Scaling performance: time per iteration [ms], speedup, and parallel efficiency for different problem sizes and GPU counts

nGPUs	nNodes	No overlap			With overlap		
		Time [ms]	Speedup	Efficiency [%]	Time [ms]	Speedup	Efficiency [%]
Small problem size: $1536 \times 512 \times 512$							
4	1	152.87	1.00	100.0	141.69	1.00	100.0
8	2	77.23	1.98	99.0	70.92	2.00	100.0
16	4	38.48	3.97	99.3	35.15	4.03	100.8
32	8	20.56	7.44	92.9	17.71	8.00	100.0
64	16	11.63	13.14	82.1	8.74	16.21	101.3
128	32	7.40	20.66	64.6	4.70	30.15	94.0
256	64	5.24	29.17	45.6	3.11	45.56	71.2
512	128	4.57	33.44	26.1	4.29	33.04	25.8
Medium problem size: $3072 \times 1024 \times 1024$							
32	8	163.74	1.00	100.0	145.61	1.00	100.0
64	16	84.72	1.93	96.7	71.48	2.04	102.0
128	32	45.99	3.56	89.0	35.66	4.08	102.0
256	64	27.65	5.92	74.0	18.14	8.03	100.4
512	128	19.45	8.42	52.6	18.76	7.76	48.5
Large problem size: $6144 \times 2048 \times 2048$							
256	64	194.34	1.00	100.0	140.57	1.00	100.0
512	128	115.54	1.68	84.0	107.87	1.30	65.0

Appendix B. EuroHPC Supercomputers

In this appendix, we provide detailed hardware specifications for the supercomputing systems employed in this study.

Karolina Cluster – Accelerated Compute Nodes

The accelerated partition of the Karolina Cluster, hosted at IT4Innovations National Supercomputing Center (Czech Republic), comprises 72 GPU-accelerated nodes equipped with NVIDIA A100 GPUs. This system was primarily utilized for early-stage benchmarking and development of the code.

- **Total Nodes:** 72
- **Processor:** 2× AMD EPYC™ 7763 (2.45 GHz, 64 cores each) per node
- **CPU Cores per Node:** 128 (9,216 cores in total)
- **Main Memory:** 1,024 GB DDR4 3200 MT/s per node
- **GPU Configuration:** 8× NVIDIA A100 GPUs per node (320 GB HBM2 memory per node)
- **Peak Performance:** 361.27 TFLOPS
- **Interconnect:** 4× 200 Gb/s InfiniBand ports per node

MareNostrum 5 ACC (Accelerated Partition)

The accelerated partition of MareNostrum 5, operated by the Barcelona Supercomputing Center (Spain), consists of 1120 nodes based on the latest Intel Sapphire Rapids processors and NVIDIA Hopper GPUs. This system was used for large-scale performance evaluation and production runs.

- **Total Nodes:** 1,120
- **Processor:** 2× Intel Sapphire Rapids 8460Y+ (2.3 GHz, 40 cores each) per node
- **CPU Cores per Node:** 80
- **Main Memory:** 512 GB DDR5 per node
- **GPU Configuration:** 4× NVIDIA Hopper GPUs with 64 GB HBM2 memory each
- **Storage:** 480 GB NVMe (scratch space)
- **Interconnect:** 4× NDR200 ports (800 Gb/s per node)
- **Peak Performance:** 260 PFLOPS
- **Network Topology:** Fat-tree topology with full non-blocking 160-node islands and 2:1 contention between islands

References

[1] V. Dhir, Boiling heat transfer, Annual review of fluid mechanics 30 (1) (1998) 365–401. doi:10.1146/annurev.fluid.30.1.365.

[2] L. Fei, J. Yang, Y. Chen, H. Mo, K. H. Luo, Mesoscopic simulation of three-dimensional pool boiling based on a phase-change cascaded lattice Boltzmann method, Physics of Fluids 32 (10) (2020) 103312. doi:10.1063/5.0023639.

[3] H. Jiang, Y. Liu, H. Chu, A review of numerical investigation on pool boiling, Journal of Thermal Analysis and Calorimetry 148 (17) (2023) 8697–8745. doi:10.1007/s10973-023-12292-0.

[4] S. Succi, The Lattice Boltzmann Equation: For Complex States of Flowing Matter, OUP Oxford, 2018. doi:10.1093/oso/9780199592357.001.0001.

[5] L. Fei, K. H. Luo, Q. Li, Three-dimensional cascaded lattice boltzmann method: Improved implementation and consistent forcing scheme, Physical Review E 97 (5) (2018) 053309. doi:10.1103/PhysRevE.97.053309.

[6] X. Shan, H. Chen, Lattice boltzmann model for simulating flows with multiple phases and components, Physical Review E 47 (3) (1993) 1815. doi:10.1103/PhysRevE.47.1815.

[7] X. Shan, H. Chen, Simulation of nonideal gases and liquid-gas phase transitions by the lattice boltzmann equation, Physical Review E 49 (4) (1994) 2941. doi:10.1103/PhysRevE.49.2941.

[8] Q. Li, K. H. Luo, X. Li, Forcing scheme in pseudopotential lattice boltzmann model for multiphase flows, Physical Review E 86 (1) (2012) 016709. doi:10.1103/PhysRevE.86.016709.

[9] Q. Li, K. H. Luo, X. Li, Lattice boltzmann modeling of multiphase flows at large density ratio with an improved pseudopotential model, Physical Review E 87 (5) (2013) 053301. doi:10.1103/PhysRevE.87.053301.

[10] Q. Li, Q. Kang, M. M. Francois, Y. He, K. H. Luo, Lattice boltzmann modeling of boiling heat transfer: The boiling curve and the effects of wettability, International Journal of Heat and Mass Transfer 85 (2015) 787–796. doi:10.1016/j.ijheatmasstransfer.2015.01.136.

[11] T. Lee, C.-L. Lin, A stable discretization of the lattice boltzmann equation for simulation of incompressible two-phase flows at high density ratio, Journal of Computational Physics 206 (1) (2005) 16–47. doi:10.1016/j.jcp.2004.12.001.

[12] R. Huang, H. Wu, N. A. Adams, Lattice boltzmann model with self-tuning equation of state for multiphase flows, Physical Review E 99 (2) (2019) 023303. doi:10.1103/PhysRevE.99.023303.

[13] P. Yuan, L. Schaefer, Equations of state in a lattice boltzmann model, Physics of Fluids 18 (4) (2006) 042101. doi:10.1063/1.2187070.

[14] E. Calore, A. Gabbana, S. F. Schifano, R. Tripiccione, Optimization of lattice boltzmann simulations on heterogeneous computers, The International Journal of High Performance Computing Applications 33 (1) (2019) 124–139. doi:10.1177/1094342017703771.

[15] E. Calore, A. Gabbana, J. Kraus, E. Pellegrini, S. Schifano, R. Tripiccione, Massively parallel lattice–boltzmann codes on large gpu clusters, Parallel Computing 58 (12) (2016) 1–24. doi:10.1016/j.parco.2016.08.005.

[16] E. Calore, A. Gabbana, J. Kraus, S. F. Schifano, R. Tripiccione, Performance and portability of accelerated lattice Boltzmann applications with OpenACC, Concurrency and Computation: Practice and Experience 28 (12) (2016) 3485–3502. doi:10.1002/cpe.3862.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 62–71

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Hierarchical Dynamic Load Balancing Strategy for a *p*-adaptive Discontinuous Galerkin Compressible LES Solver

Paolo Valvo[a,*], Antonella Abbà[a]

[a]*DAER Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano Via la Masa 34, Milan, 20161, Italy*

Abstract

The current work proposes a dynamic hierarchical load balancing strategy for a *p*-adaptive local discontinuous Galerkin compressible LES solver. Firstly, the load balancing problem is properly reformulated as a graph partitioning problem. Then, the graph partitioning step is solved exploiting the partitioning capabilities of the *Zoltan* library. Finally, this approach is validated on a classic benchmark for LES simulations, that is, the flow past a square cylinder. The hardware-informed load balancing capabilities and the overall performance of the proposed strategy are discussed. Lastly, the influence of load balancing on the results is investigated. Results demonstrate the goodness of the current framework, that allows for dynamically balanced adaptive simulations of complex and unsteady turbulent flows.

Keywords: Hierarchical Load Balancing, Discontinuous Galerkin, LES;

1. Introduction

The discontinuous Galerkin (DG) methods are one of the emerging methods in CFD. One of their main characteristics is their reduced stencil, that allows for highly local formulations and all the implied advantages. Remarkably, such schemes allow for the straightforward implementation of adaptive strategies, as demonstrated by the plenty of published works about the design of good indicators to drive *p*-, *h*- and *hp*-adaptation. Although remarkable results have been achieved concerning the optimum placement of degrees of freedom, this is only a part of the problem. Indeed, adaptation alters the computational effort over time, especially with dynamic adaptation and unsteady flows. This is a potential problem for any parallel implementation, since the best parallel efficiency is achieved by distributing as evenly as possible the overall computational effort among the different tasks. For this reason, in the last years some authors have begun to introduce some load balancing strategies in their codes. For example, both [9] and [3] employs the serial *METIS* library to perform load balancing at run-time in their ILES *p*-adaptive solvers. In particular, [3] divide the load among the parallel tasks by assigning a portion of the computational grid to each task, basically formulating

* Corresponding author.
E-mail address: paolo.valvo@polimi.it

1877-0509
10.1016/j.procs.2025.08.233

the load balancing problem as a grid partitioning problem. The grid is then described as a graph and the problem is addressed through the multi-weight multilevel graph partitioner of *METIS*. In order to avoid the potential bottleneck due to the serialization of the partitioning process, subsequent works [26, 14] on p-adaptive DG ILES solvers always reformulate the problem as a graph partitioning problem, but address it through *ParMETIS*, the parallel version of *METIS*. For the sake of completeness, [17] proposes a different parallel solution to the load balancing problem. This approach is based on the capabilities of the *Charm++* system to auto-instrument a code to measure the effective CPU load and consequently move the data objects. However, code instrumentation is potentially detrimental to performance, so this idea is not considered in the current work. Notice that the researches [9, 3, 26, 14] basically adopt the same partitioning approach, only exploring different possible weights for the vertices of the graph. To the authors' knowledge, the only contribution to explore different graph partitioning methods in the framework of adaptive DG LES/ILES solvers is [15], who proposes a two-step partitioning where geometric and multilevel connectivity-based methods from *METIS*/*ParMETIS*, *Zoltan* and *GeMPa* are combined together.

This two-step strategy is a peculiar case of what is better known in literature as hierarchical partitioning. In particular, this concept was first introduced by [23] and, as pointed out by [16, 18], it can significantly improve the quality of partitions. Indeed, it allows for partitions aware of the hardware locality and then of the communication buses. In this way, the resulting partition can be optimized from both the perspectives of load and communication imbalance. Moreover, it also improves the overall quality of partitions with respect to a single step partition, which can perform badly when a large number of partitions are required [16]. Notice that [15] focuses only on parallel scalability, marginally exploring these potential advantages.

In light of this discussion, the present work introduces a dynamic load balancing strategy with full hierarchical partitioning capabilities inside a p-adaptive local DG (LDG) scheme. Following the previous activity of the present research group [2, 24, 1], this result is practically achieved by embedding the *Zoltan* toolkit [6] inside the p-adaptive LDG solver *dg-comp*, implemented in the *FEMilaro* Fortran library [13]. The remainder of this paper briefly recalls the mathematical formulation in section 2, then presents the load balancing strategy in section 3 and discusses the validation of the current implementation in section 4.

2. Math formulation

This section briefly recalls the mathematical formulation behind the *dg-comp* solver. The present discretization is based on Cockburn et al. [8] and Bassi and Rebay [4]. Its details have been extensively discussed in previous works [2, 24], to which the interested reader is referred for further details. In particular, the *dg-comp* LDG formulation searches the solution in the vector space $\mathcal{V}_h = \{v_h \in L^2(\Omega) : v_h\big|_K \in \mathbb{P}^q(K), \forall K \in T_h\}$, where T_h is a tessellation of tetrahedra of the computational domain Ω and K is an element of this tessellation, while $\mathbb{P}^q(K)$ is the polynomial space of order q with compact support K. The basis functions used to describe $\mathcal{V}_h$ are, for each element K, the orthonormal Legendre polynomials defined over K. The resulting weak form of the compressible Navier-Stokes equation is:

$$\begin{aligned}
&\partial_t \int_K \mathbf{U} v_h d\Omega - \int_K \mathbf{F}^e \cdot \nabla v_h d\Omega + \int_K (\mathbf{F}^v - \mathbf{F}^{SGS}) \cdot \nabla v_h d\Omega + \\
&\qquad + \int_{\partial K} \mathbf{F}^e \cdot \mathbf{n}_{\partial K} v_h d\Sigma - \int_{\partial K} (\mathbf{F}^v - \mathbf{F}^{SGS}) \cdot \mathbf{n}_{\partial K} v_h d\Sigma = 0 \\
&\int_K \boldsymbol{\mathcal{G}} \cdot \mathbf{r}_h d\Omega + \int_K \boldsymbol{\Phi} \nabla \cdot \mathbf{r}_h d\Omega - \int_{\partial K} \boldsymbol{\Phi} \mathbf{n}_{\partial K} \cdot \mathbf{r}_h d\Sigma = 0
\end{aligned} \tag{1}$$

In equation (1), $\mathbf{r}_h \in \mathcal{V}_h^d$, with d the dimension of the domain. Furthermore, $\mathbf{F}^e = \mathbf{F}^e(\mathbf{U})$, $\mathbf{F}^v = \mathbf{F}^v(\mathbf{U}, \boldsymbol{\mathcal{G}})$ and $\mathbf{F}^{SGS} = \mathbf{F}^{SGS}(\mathbf{U}, \boldsymbol{\mathcal{G}})$ are, respectively, the convective (or Eulerian), the diffusive (or viscous) and the subgrid fluxes. Here, $\boldsymbol{\mathcal{G}} = \nabla \mathbf{U}$, $\mathbf{U} = [\rho, \rho e^{tot}, \rho \mathbf{u}]$, $e^{tot} = e + \|\mathbf{u}\|^2/2$, and $\boldsymbol{\Phi} = [T, \mathbf{u}]$, where ρ is the density, $\mathbf{u}$ the velocity, e the internal energy and T the temperature. In general, *dg-comp* offers both LES and ILES capabilities. In this work, an explicit LES approach is adopted. As a consequence, all the above quantities must be intended in a Favre-filtered sense [12], where the LES filter is implicitly determined by the numerical discretization as the projection of the exact solution in

the polynomial space where the numerical solution is searched [2]. In particular, this work adopts the turbulent closure of Smagorinsky for modeling $\mathbf{F}^{SGS}$. Concerning the fluxes on the surfaces ∂K, $\mathbf{F}^{e}$ is estimated by the approximated Riemann solver of Rusanov [7] up to a minor modification, since the sound speed is computed using the state variables in place of the ROE ones. Instead, $\mathbf{F}^{v}$ and $\mathbf{F}^{SGS}$ are modeled with a centered scheme as in [4]. The resulting equation is advanced in time with the explicit five-stages fourth-order strong stability preserving Runge-Kutta method (SSPRK54) [21].

Concerning p-adaptivity, the adaptation process is driven by the estimate of the amount of under-resolved turbulence [24]. The selected indicator is based on the second order structure function $D_{ij}(\mathbf{x}, \mathbf{r})$ and is reported in equation (2). Here, $\langle \cdot \rangle$ denotes the expected value operator.

$$Ind_{SF}(K) = \sqrt{\sum_{ij} [D_{ij}(K) - D_{ij}^{ISO}]^2} \quad \text{where} \quad D_{ij}(\mathbf{x}, \mathbf{r}) = \langle [u_i(\mathbf{x} + \mathbf{r}, t) - u_i(\mathbf{x}, t)][u_j(\mathbf{x} + \mathbf{r}, t) - u_j(\mathbf{x}, t)] \rangle \tag{2}$$

The indicator is calculated for all elements at each $dt_{indicator}$ time instant and averaged over a time interval of width dt_{adapt}. Every dt_{adapt} the solution is adapted according to a series of thresholds ϵ_i, with $i = 1, \ldots, p_{max}$, such that if $Ind_{SF}(K) > \epsilon_{p+1}$ the degree of K is increased by one, while if $Ind_{SF}(K) < \epsilon_p$ is decreased by one.

3. Hierarchical load balancing strategy

The *dg-comp* solver is based on a pure MPI programming model, where the load is distributed among the various MPI tasks by assigning to each task a portion of the computational grid. Thus, the load balancing problem is naturally reformulated as a grid partitioning and then as a graph partitioning problem. As discussed in section 1, the classic approach in the literature generates a graph G based on the grid connectivity information. In particular, each vertex v_i of G describes an element K_i of the grid and the edges e_{ij} of G link K_i to and only to its neighboring elements K_j. This idea is sketched in figure 1. Moreover, each vertex v_i is equipped with one or more weights w_i, which describe the computational cost of K_i. However, notice that this strategy produces partitions that are balanced only from the load viewpoint, completely neglecting the communication cost between partitions. For this reason, the present work describes the grid with a hypergraph G' based on the stencil of the numerical method. The various hyperedges then qualitatively describe the communication pattern implied by the numerical method and, if they are equipped with a proper weights, such a description is even quantitative. In this case, connectivity-based partitioning schemes minimize both load and communication imbalance. In the peculiar case of a LDG scheme, the stencil is very reduced. Indeed, the computation of the solution on the element K_i requires data only from the neighboring elements and K_i itself. Then, the hypergraph G' basically collapses to the graph G, up to the edge's weights w_{ij}^{e} and the grid sides with periodic boundary conditions. Indeed, elements on these sides communicate across the periodic boundary. Then, the proposed graph G' has some additional edges that link these physically disjoint but logically connected pairs of elements. In addition, the description G' also attributes to each v_i the coordinates of the baricenter of K_i. In this way, the graph partitioning step is solvable both with geometric and connectivity-based methods. Concerning the choice of weights, w_i is the number of Gauss quadrature points on the element K_i, while w_{ij}^{e} the number of quadrature points on the surface $K_i \cap K_j$. Indeed, a previous study by this research group [25] shows that these weights are accurate enough to estimate the effective CPU load and the total exchanged data between the various tasks.

The partitioning of G' is performed through a hierarchical partitioning strategy. In particular, said N_{tasks} the number of MPI tasks and then of desired partitions. The hierarchical strategy does not generate all partitions in a single step but performs a recursive split of G'. Then, at the first step s_1 np_1 subgraphs are generated, then at the subsequent step s_2 each subgraph is divided into np_2 partitions and so on until N_{tasks} partitions are generated. Moreover, also the overall set of MPI tasks is recursively divided in groups of tasks and each group is attributed to a given subgraph. In particular, these groups always contain tasks with subsequent MPI rank. For example, said $N_{tasks}^{s_1} = N_{tasks}/np_1$, at the s_1 step the tasks are divided into np_1 groups, where the j-th group contains the tasks of rank $jN_{tasks}^{s_1}, 1+jN_{tasks}^{s_1}, \ldots, (j+1)N_{tasks}^{s_1}-1$, $j = 0, \ldots, np_1 - 1$. Of course, np_i must be selected to get no remainder for all splits of the groups of tasks. If such

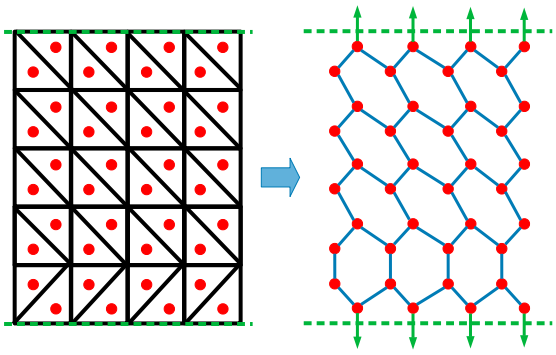


Fig. 1: Example of a $2D$ grid with two periodic boundary sides, in green. On the left, the original grid with the baricenter of each element marked in red. On the right, its associated graph G'. The red dots are the vertices, located at the same position of the baricenter of the corresponding elements. The blue lines are the edges that would appears also in G, while the green arrows are the extra edges due to periodic boundary conditions.

a strategy is properly combined with CPU binding of MPI tasks at run-time, a hardware-informed load balancing strategy is obtained. The concept is better discussed in section 4.

This result is practically obtained embedding the *Zoltan* toolkit within the *dg-comp* solver. In particular, at each partitioning step one of four native *Zoltan* methods can be employed. The first three are geometric algorithms, based only on the coordinates of v_i. They consider only w_i, then the partitions are not optimized for communication. In detail, these methods are the Recursive Coordinate Bisection (RCB) [5], its variant the Recursive Inertial Bisection (RIB) [22] and a Hilbert-based space filling curve approach (HSFC) [10, 19]. The fourth method is the *Zoltan* native connectivity-based graph partitioning algorithm (here, GRAPH) [11]. Differently from geometric methods, connectivity-based algorithms perform a multi-constrained optimization, minimizing both load and communication imbalances. In particular, to avoid communication imbalance the method minimizes the communication volume V_{comm}, which for a given partition P_k is defined as the sum of w^e_{ij} of the edges which cross ∂P_k, i.e., which connect P_k to other partitions. In practice, this quantity estimates the amount of data a partition exchanges with the other partitions. This definition is summarized in the equation (3).

$$V_{comm} = \sum_{e_{ij} \cap \partial P_k \neq \emptyset} w^e_{ij} \tag{3}$$

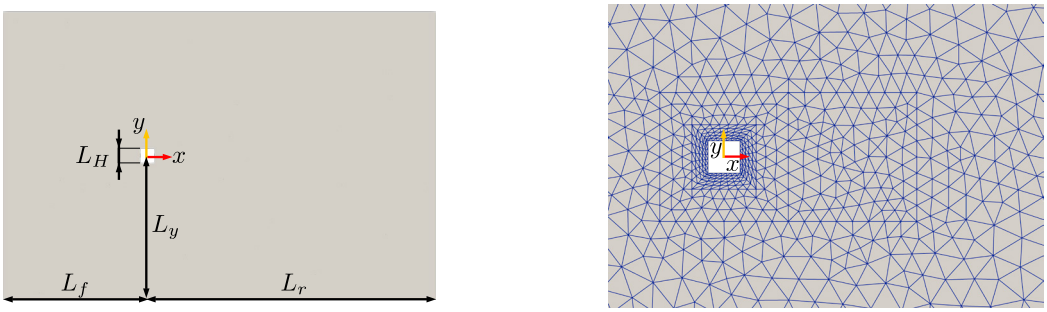
In the GRAPH case, the options coded inside *dg-comp* imply a multilevel graph partitioning, with an agglomerative inner product matching coarsening strategy and an approximate Fiduccia-Mattheyses refinement algorithm.

4. Square cylinder simulations

4.1. Setup of the simulation

The proposed load balancing strategy is validated on a classic benchmark for LES simulations, i.e., the flow past a square cylinder. Due to the strong unsteadiness of the wake, this test case is very suitable also to validate adaptive strategies and then load balancing techniques. In particular, the selected flow conditions are a Reynolds number of 22000 and a Mach number of 0.3.

The computational domain is sketched in figure 2a. The origin of the reference frame is in the center of the cylinder and the dimensions of the domain are, in dimensionless units, $L_H = 1$, $L_y = 10$, $L_f = 10$ and $L_r = 20$. This domain is extruded in the z direction, from $z = 2$ to $z = -2$ and meshed with an unstructured grid of 23816 thetraedra, more refined near the cylinder. A zoom of the grid is reported in figure 2b. The simulation over this domain requires periodic



(a) Domain

(b) Slice of the mesh at $z = 2$.

Fig. 2: The computational domain with its characteristic dimensions and the mesh.

conditions along the z direction, Neumann slip wall conditions on the top and bottom sides and unitary Dirichlet conditions on the front and rear sides. In order to better enforce Dirichlet conditions, a sponge layer of thickness 2 is employed on the front and 4 on the rear. The simulation is advanced in time with a time step $dt = 0.0004$, which implies a maximum CFL of 0.24. Concerning the initial conditions, a preliminary simulation is advanced for 30 time units with uniform polynomial degree $p = 2$, starting from a uniform flow field with a Gaussian damping on velocity around the cylinder. The resulting flow field is the initial condition for all the other simulations of this work.

4.2. Numerical results

The present simulations run on *Leonardo* supercomputer, hosted by CINECA at Bologna, Italy. The employed partition of *Leonardo* is composed of nodes equipped with two 56-cores Intel Xeon Platinum 8480+ processors. All simulations are run on three nodes. The resulting 336 MPI tasks are sequentially bound to the hardware. The run time layout is schematically depicted in figure 3. In order to demonstrate the effectiveness of the hierarchical load

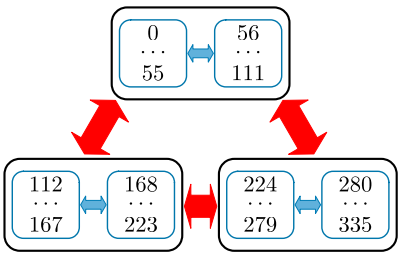


Fig. 3: Hardware layout and run-time CPU binding of MPI tasks. The black rectangles the nodes, the blue ones the processors and the arrows describe the data buses. The numbers are the MPI ranks of the tasks according to the adopted CPU binding.

balancing proposed in section 3, a set of ten simulations is performed. Starting from the initial conditions described in section 4.1, the simulation is advanced in time for 60 time units. The first simulation is with uniform polynomial degree $p = 4$ and uniform load distribution. The second simulation uses the same fixed load distribution but performs p-adaptation, varying the polynomial degree from $p = 2$ to $p = 4$. The indicator thresholds are $\epsilon_{p=3} = 5e-4$ and $\epsilon_{p=4} = 1e-2$, the indicator is computed each $dt_{indicator} = 4dt$ and the solution is adapted each $dt_{adapt} = 0.05$. This setup has already been validated by this research group in [1]. The other simulations perform both adaptation and dynamic load balancing, repartitioning the grid and then redistributing the loads every $dt_{repart} = dt_{adapt}$. In particular, a three-step hierarchical load balancing is performed, creating 3 and 2 partitions with the GRAPH method at the first two levels and 56 partitions at the third level, using one among RCB, RIB, HSFC and GRAPH. Thus, this approach divides the grid into three parts at the first level, attempting to minimize both the load and the communication imbalances.

Then, each subgrid is split into two further parts, always minimizing the load and the communication imbalances. Finally, each second-level subgrid is divided again into 56 parts, considering or not the communication cost according to the selected method. Moreover, a similar splitting scheme is applied to MPI tasks. In particular, at the first level the first subgrid is assigned to tasks with MPI ranks 0-111, the second to ranks 112-223 and the third to 224-335. At the second level, each of these groups of tasks is divided again, assigning to the first second-level subgrid the first half of the ranks and to the second the remaining part. Finally, a subgrid per task is assigned with the last partitioning step. As a consequence, when this strategy is combined with the CPU binding exposed in figure 3, a hardware-informed load balancing is obtained. Indeed, the first level minimizes the node-to-node communication, while the second level the processor-to-processor communication within the same node. These simulations are compared with both the references and the single-step application of the four partitioning methods, that is, asking for 336 partitions at the same time with one of RCB, RIB, HSFC or GRAPH.

All these simulations have an equivalent level of accuracy, as discussed later in the paper. For now, consider only the performances of these simulations in terms of elapsed times and communication volumes. In particular, consider two kinds of volumes. The first one is the group communication volume V_{comm}^{i-j}, that is, the communication volume computed considering all partitions from i to j as a single partition. In this context, they are of interest V_{comm}^{0-111}, $V_{comm}^{112-223}$ and $V_{comm}^{224-335}$. These are the communication volumes associated with the partitions resulting from the first hierarchical step and then represent the amount of data a node exchanges with the other two. Secondly, consider the metrics V_{comm}^{intra}, which expresses the amount of exchanged data between the two processors on a single node. For the first node, this quantity is computed as $V_{comm}^{intra1} = (V_{comm}^{0-55} + V_{comm}^{56-111} - V_{comm}^{0-111})/2$. Similar definitions are possible for the second and the third node. This metrics is preferred to the communication volume of the resulting partition assigned to a single processor, as V_{comm}^{0-55} or V_{comm}^{56-111}, since this latter choice would not distinguish between the intra-node and inter-node communications of a processor. These data are summarized in table 1 according to the following naming scheme: identify with p4 and ADnoLB the $p = 4$ and adaptive references and with the names of the partitioning method the other ones. In hierarchical cases, the name is the method of the last partitioning step with the prefix HIER. For example, HIER-RCB denotes the simulation in which the 6 groups of 56 partitions at the last partitioning step are computed with the RCB method.

Table 1: Load balancing performances. The group communication volume is the communication volume considering the partitions assigned to a group of tasks as a single partition. In this case, V_{comm}^{0-111}, $V_{comm}^{112-223}$ and $V_{comm}^{224-335}$ are the communication volumes of the nodes. Instead, V_{comm}^{intra1} is the intra-node communication volume for node 1, i.e., the sum of all w_{ij}^{e} of the edges connecting the two partitions assigned to the different processors of node 1. Communication volumes are reported for the last time instant.

Simulation	Elapsed time [s]			Group communication volume [-]			Intra-node volume [-]
	Computation	Balancing	total	V_{comm}^{0-111}	$V_{comm}^{112-223}$	$V_{comm}^{224-335}$	V_{comm}^{intra1}
p4	18568	0	18568	13128	18912	12432	5844
ADnoLB	16866	0	16866	7155	9315	5166	3861
RCB	9446	839	10285	14142	22935	10491	6996
RIB	9634	912	10546	12630	24120	14490	4344
HSFC	9924	827	10751	20460	9114	20712	1824
GRAPH	-	-	-	-	-	-	-
HIER-RCB	9886	864	10750	4164	1944	2532	1848
HIER-RIB	9860	882	10742	4626	2538	2262	2040
HIER-HSFC	9899	848	10747	4047	2064	2583	1866
HIER-GRAPH	10364	989	11353	4350	2535	2253	1539

Looking at the degrees of freedom, the p4 simulation has 883455 $dofs$, while the adaptive simulations are around 456000 $dofs$, that is, a reduction of about -48% with respect to p4. Then, in the ideal case, the adaptive simulations should approximately halve their total elapsed time. However, it is rather evident that, without load balancing, the obtained gain is much smaller. Indeed, the reduction in the elapsed time is just -9.8% for ADnoLB. Instead, all dynamically balanced simulations almost achieve this result, with the exception of the GRAPH case. Indeed, this

simulation does not converge due to an internal failure of *Zoltan*, that likely depends on the peculiar options used to invoke the main partitioning routine. In particular, the options coded in *dg-comp* ask for a full partitioning of the graph from scratch, which is a rather demanding task. Invoking the GRAPH method with less severe requirements may avoid this failure. However, it is worth noting that the HIER-GRAPH partitioning scheme converges, stressing how hierarchical partitioning is generally more robust than the single step application of a certain partitioning algorithm when a large number of partitions is required [16].

Furthermore, the proposed hierarchical procedure successfully minimizes the data communication along some given data buses. Considering the node-to-node communication, RCB, RIB and HSFC have much larger node communication volumes V_{comm}^{0-111}, $V_{comm}^{112-223}$ and $V_{comm}^{224-335}$ than their hierarchical counterparts, even ten times larger. Then, the first step of the hierarchical procedure, which generates the subgrids for the groups of tasks 0-111, 112-223 and 224-335, successfully minimizes the node-to-node communication. In the same way, the intra-node communications between the two processors on the same node are minimized by the second hierarchical step. Indeed, looking at table 1 V_{comm}^{intra1} is reduced by the hierarchical strategy, even if the advantage is smaller than at the node level. Comparable results are found for the other nodes. Thus, the second hierarchical step actually minimizes the intra-node communication. Then, the proposed coupling of run-time CPU binding and hierarchical load balancing successfully results in a hardware-informed load balancing strategy, reducing the data communication along all the main data buses.

Concerning the total elapsed time, hierarchical partitioning is slightly slower than the single-step approaches for the RCB and RIB cases, while it is almost the same for HSFC and HIER-HSFC. This suggests that, by virtue of its high locality, the present LDG scheme is not communication-limited when simulations are executed on supercomputers as *Leonardo*. Indeed, these machines have the state of the art of communication buses, with very large bandwidths and low latency. However, different situations may produce different results. This could likely be the case of a computation executed on different remote nodes that are hosted at different locations and must communicate between themselves through, e.g., internet.

Finally, consider the accuracy of the current computations. Table 2 reports the Strouhal number and statistics for the force coefficients. The p4 and ADnoLB simulations agree, legitimizing the adaptation parameters selected for

Table 2: Accuracy of numerical results. The Strouhal number is estimated as a weighted average of the highest contributions of the Fourier transform of the C_L time history. The mean results for all the performed simulations are expressed with their 90% confidence interval.

Simulation	Mean drag coefficient [-]	RMS force coefficients [-]		Strouhal number
	$\langle C_D \rangle$	RMS_{C_D}	RMS_{C_L}	St
p4	2.445	0.1531	1.366	0.1398
ADnoLB	2.3818	0.1934	1.305	0.1378
RCB	2.3852	0.1934	1.321	0.1383
RIB	2.442	0.2187	1.410	0.1374
HSFC	2.4281	0.1793	1.417	0.1372
GRAPH	-	-	-	-
HIER-RCB	2.381	0.1903	1.289	0.1390
HIER-RIB	2.374	0.1975	1.308	0.1381
HIER-HSFC	2.441	0.2012	1.419	0.1333
HIER-GRAPH	2.412	0.1873	1.360	0.1383
Other LES data [20]	1.66 - 2.77	0.10 - 0.27	0.38 - 1.79	0.066 - 0.15

this work. Moreover, the numerical results are also in good agreement with data in the literature [20] at the same Reynolds number. Nevertheless, it is worth noting that ADnoLB results do not perfectly coincide with the other adaptive runs, even if, in principle, they are the same simulation. Indeed, a load balancing strategy redistributes the loads, but should not alter the results. However, this only is a purely theoretical requirement, that cannot be really satisfied. In fact, floating point arithmetic is non-commutative and the load balancing procedure unavoidably alters the order of computations, at least in some points. For example, *dg-comp* code at first performs all the computation on

a single partition, then a communication step takes place and after the remaining computations are completed. As a consequence, the repartition of the grid at every load balancing step modifies the order of some operations, introducing a small numerical disturbance. At first, the difference between two adaptive simulations is inappreciable. Nevertheless, if a very long simulation is performed, it is reasonable that, at a given instant, these differences may be amplified and become non-negligible for some points in the domain. Then, different simulations would slowly differ one from the other. This is exactly the present case. Indeed, the simulations start to differ after about 30 time units. This is rather evident in figure 4, which reports the time history of C_D for the p4, ADnoLB, RCB, RIB and HSFC cases. The other simulations are not reported in the figure, but exhibit the same behavior. This slow change from ADnoLB time history explains the differences in the $\langle C_D \rangle$ value in table 2, as well as the other discrepancies. The same result is found by

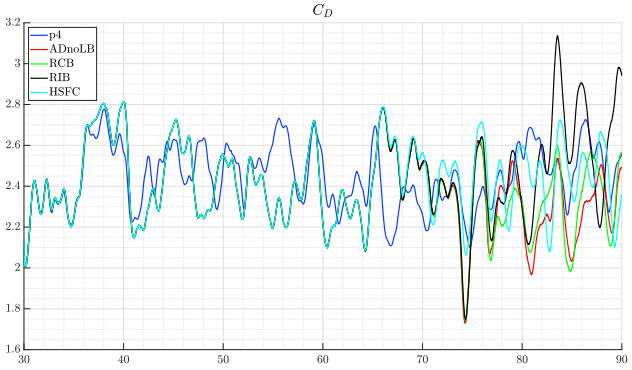
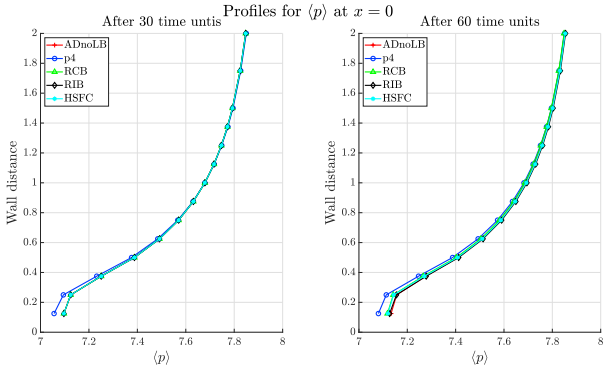
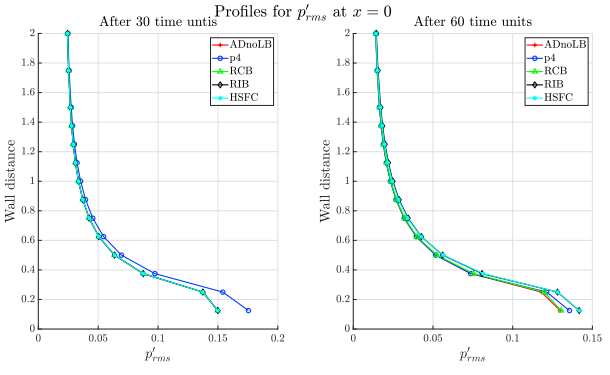


Fig. 4: Time history of the drag coefficient for p4, ADnoLB, RCB, RIB and HSFC simulations. The simulation starts from $t = 30$ since the $p = 2$ startup goes from $t = 0$ to $t = 30$.

looking at figure 5, which reports the profiles of the pressure statistics at $x = 0$ on the upper side of the cylinder. After 30 time units from the beginning of the simulation, the adaptive simulations still agree perfectly. Indeed, after this time interval all profiles except the one of p4 are overlapped, both for the mean pressure profile $\langle p \rangle$ in figure 5a and the RMS profile of $p - \langle p \rangle$ in figure 5b. Instead, at the end of the simulation, statistics show a small scattering. A small but no longer negligible difference arises. Similar results are obtained for all the other simulations and all the other meaningful quantities as, e.g., the statistics of density and velocity components.



(a) Mean pressure profiles after 30 and 60 time units.



(b) Profiles after 30 and 60 time units for the rms of $p' = p - \langle p \rangle$.

Fig. 5: Profiles of pressure statistics on the upper side of the cylinder, at $x = 0$. Data are reported for p4, ADnoLB, RCB, RIB and HSFC simulations.

This produces a concern about the reliability of the results. However, notice that a certain level of uncertainty is implicitly accepted considering p4 and ADnoLB simulations as equivalent. Thus, the observed numerical noise must be compared with this uncertainty in order to determine its importance. For this reason, consider two sets of data, one made of the p4 and ADnoLB results and the other made of all adaptive results including ADnoLB. Table 3 reports

the statistical comparison of the results of these two sets of simulations for the data in table 2. Observe that, except

Table 3: Statistical comparison between the accepted uncertainty and the numerical noise. For each considered set of simulations, the data in table 2 are elaborated statistically.

Set of simulations	Mean				Standard deviation σ			
	$\langle C_D \rangle$	RMS_{C_D}	RMS_{C_L}	St	$\langle C_D \rangle$	RMS_{C_D}	RMS_{C_L}	St
p4 and ADnoLB	2.414	0.1732	1.335	0.1388	0.045	0.0285	0.043	0.0014
Adaptive	2.407	0.1951	1.354	0.1374	0.029	0.0116	0.055	0.0018

for RMS_{CL}, the standard deviations of all the considered data are smaller for the second set of simulations. Then, on average, the variability of the results due to the load balancing scheme is smaller than the uncertainty accepted in the adaptive results. As a consequence, the load-balanced simulations can be considered equivalent to each other exactly as the p4 and the ADnoLB simulations are assumed statistically equivalent. The statistical meaningfulness of the results is conserved independently of the load balancing strategy, even if the time histories of instantaneous values can exhibit appreciable differences on long simulations.

5. Conclusions

The present work successfully designs a dynamic hierarchical load balancing strategy, that produces a hardware-informed load balancing when properly combined with CPU binding of MPI tasks. The strategy has been formulated in its details for a LDG-LES solver and practically implemented in the open source *p*-adaptive *dg-comp* solver of the *FEMilaro* library [13]. However, the proposed methodology is easily generalizable to other solvers and even to different numerical schemes.

Furthermore, the present work investigates the effects of a load balancing strategy on the results of the simulations. For long simulations the partitioning approach marginally affects the solution. Results cannot be expected to coincide any longer, but the discrepancy is limited and contained within the range of the accepted uncertainty. This ensures the statistical significance of the computations, independently of the adopted load balancing strategy.

In conclusion, the state of the art of load balancing is combined with one of the most promising emerging methods in CFD for the simulation of turbulent unsteady flows.

Acknowledgements

We acknowledge EuroHPC Joint Undertaking for awarding us access to LEONARDO at CINECA, Italy.

References

[1] Abba, A., Recanati, A., Tugnoli, M., Bonaventura, L., 2020. Dynamical p – adaptivity for LES of compressible flows in a high order DG framework. Journal of Computational Physics 420, 109720. doi:10.1016/j.jcp.2020.109720.

[2] Abbà, A., Bonaventura, L., Nini, M., Restelli, M., 2015. Dynamic models for large eddy simulation of compressible flows with a high order DG method. Computers & Fluids 122, 209–222. doi:10.1016/j.compfluid.2015.08.021.

[3] Bassi, F., Colombo, A., Crivellini, A., Fidkowski, K.J., Franciolini, M., Ghidoni, A., Noventa, G., 2020. Entropy-adjoint p-adaptive discontinuous Galerkin method for the under-resolved simulation of turbulent flows. AIAAJ 58, 3963–3977. doi:https://doi.org/10.2514/1.J058847.

[4] Bassi, F., Rebay, S., 1997. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. Journal of Computational Physics 131, 267–279. doi:https://doi.org/10.1006/jcph.1996.5572.

[5] Berger, M.J., Bokhari, S.H., 1987. A partitioning strategy for nonuniform problems on multiprocessors. IEEE Transactions on Computers C-36, 570–580. doi:10.1109/TC.1987.1676942.

[6] Boman, E.G., Catalyurek, U.V., Chevalier, C., Devine, K.D., 2012. The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering and coloring. Scientific Programming 20, 129–150. doi:10.1155/2012/713587.

[7] Chen, S.s., Yan, C., Xiang, X.h., 2018. Effective low-Mach number improvement for upwind schemes. Computers and Mathematics with Applications 75, 3737–3755. doi:10.1016/j.camwa.2018.02.028.

[8] Cockburn, B., Shu, C.W., 1998. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. SIAM Journal on Numerical Analysis 35, 2440–2463. doi:https://doi.org/10.1137/S0036142997316712.

[9] Colombo, A., Manzinali, G., Ghidoni, A., Noventa, G., Franciolini, M., Crivellini, A., Bassi, F., 11 June 2018. A p-adaptive implicit discontinuous Galerkin method for the under-resolved simulation of compressible turbulent flows, in: 6th European Conference on Computational Mechanics (ECCM 6) & 7th European Conference on Computational Fluid Dynamics (ECFD 7). Glasgow, UK.

[10] Dennis, J., 2003. Partitioning with space-filling curves on the cubed-sphere, in: Proceedings International Parallel and Distributed Processing Symposium, IEEE. doi:10.1109/IPDPS.2003.1213486. Conference held at Nice, France, 22-26 April 2003.

[11] Devine, K.D., Boman, E.G., Heaphy, R.T., Bisseling, R.H., Catalyurek, U.V., 2006. Parallel hypergraph partitioning for scientific computing, in: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, IEEE. doi:10.1109/IPDPS.2006.1639359. Conference held at Rhodes Island, Greece, 25-29 April 2006.

[12] Favre, A.J., 1965. The equations of compressible turbulent gases. Technical Report 1. Institut de mecanique statistique de la turbulence. Marseille, France. Annual summary report for Contract AF 61 (052)-772, Air Force office of scientific research.

[13] FEMilaro, a finite element toolbox, 2025. URL: https://bitbucket.org/mrestelli/femilaro/wiki/Home. available under GNU GPL v3. Results of the current paper are generated with the code under the Zoltan branch, version date 2025-04-24. The code will be merged in the master branch in the next future.

[14] Ghoreishi, R., Vermeire, B.C., 2024. Dynamic load balancing for vorticity-based polynomial adaptation of turbulent flows, in: AIAA SciTech 2024 Forum. doi:10.2514/6.2024-0912. Orlando, FL.

[15] Jang, Y., Martin, E., Chapelier, J.B., Couaillier, V., 2024. Two-level dynamic load-balanced p-adaptive discontinuous Galerkin methods for compressible CFD simulations. Computers & Mathematics with Applications 176, 165–178. doi:10.1016/j.camwa.2024.10.008.

[16] Kong, F., Stogner, R.H., Gaston, D.R., Peterson, J.W., Permann, C.J., Slaughter, A.E., Martineau, R.C., 2018. A general-purpose hierarchical mesh partitioning method with node balancing strategies for large-scale numerical simulations, in: 2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA), IEEE. pp. 65–72. doi:10.1109/ScalA.2018.00012. conference held at Dallas, TX, USA, 12 November 2018.

[17] Li, W., Pandare, A.K., Luo, H., Bakosi, J., Waltz, J., 2023. A parallel p-adaptive discontinuous Galerkin method for the Euler equations with dynamic load-balancing on tetrahedral grids. International Journal Numerical Methods in Fluids 95, 1913–1932. doi:https://doi.org/10.1002/fld.5231.

[18] Mohanamuraly, P., Staffelbach, G., 2020. Hardware locality-aware partitioning and dynamic load-balancing of unstructured meshes for large-scale scientific applications, in: Proceedings of the Platform for Advanced Scientific Computing Conference, ACM, New York, NY, United States. pp. 1–10. doi:10.1145/3394277.3401851. conference held at Geneve, Switzerland, 29 June-01 July 2020.

[19] Pilkington, J.R., Baden, S.B., 1996. Dynamic partitioning of non-uniform structured workloads with spacefilling curves. IEEE Transactions on Parallel and Distributed Systems 7, 288–300. doi:10.1109/71.491582.

[20] Rodi, W., Ferziger, J.H., Breuer, M., Pourquie´e, M., 1997. Status of large eddy simulation: Results of a workshop. Journal of Fluids Engineering 119, 248–262. doi:10.1115/1.2819128.

[21] Spiteri, R.J., Ruuth, S.J., 2003. A new class of optimal high-order strong-stability-preserving time discretization methods. SIAM Journal on Numerical Analysis 40, 469–491. doi:10.1137/S0036142901389025.

[22] Taylor, V.E., Nour-Omid, B., 1994. A study of the factorization fill-in for a parallel implementation of the finite element method. International Journal for Numerical Methods in Engineering 37, 3809–3823. doi:https://doi.org/10.1002/nme.1620372205.

[23] Teresco, J.D., Faik, J., Flaherty, J.E., 2006. Hierarchical partitioning and dynamic load balancing for scientific computation, in: Dongarra, J., Madsen, K., Waśniewski, J. (Eds.), Applied Parallel Computing. State of the Art in Scientific Computing. Springer Berlin Heidelberg, Berlin, Heidelberg. volume 3732, pp. 911–920. doi:10.1007/11558958_110. series title: Lecture Notes in Computer Science. First appeared in conference *PARA*, 20-23 June 2004, Lyngby, Denmark.

[24] Tugnoli, M., Abbà, A., Bonaventura, L., Restelli, M., 2017. A locally p-adaptive approach for large eddy simulation of compressible flows in a DG framework. Journal of Computational Physics 349, 33–58. doi:10.1016/j.jcp.2017.08.007.

[25] Valvo, P., Abbà, A., . A hardware-informed dynamic load balancing for a p-adaptive local discontinuos galerkin LES solver. Submitted to *Computer & Fluids* the 16th of April, 2025.

[26] Wang, L., Gobbert, M.K., Yu, M., 2020. A dynamically load-balanced parallel p-adaptive implicit high-order flux reconstruction method for under-resolved turbulence simulation. Journal of Computational Physics 417, 109581. doi:https://doi.org/10.1016/j.jcp.2020.109581.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 72–81

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Enabling Ginkgo as Numerics Backend in nekRS Employing A Loosely-Coupled Configuration File Concept

Yu-Hsiang Mike Tsai[a], Mathis Bode[b], Hartwig Anzt[a,c,*]

[a]*Technical University of Munich, Heilbronn, Germany*
[b]*Forschungszentrum Jülich, Germany*
[c]*The Innovative Computing Laboratory, The University of Tennessee, USA*

Abstract

In computational fluid dynamics (CFD), the choice of numerical methods can significantly impact the overall simulation runtime. While it is virtually impossible to know the optimal solver plus preconditioner configuration for every hardware and application setup, it is valuable for CFD engineers to have access to and evaluate different numerical methods to customize the setup for efficient execution. In this paper, we demonstrate how the Ginkgo high-performance numerical linear algebra library is integrated as a math toolbox into the nekRS state-of-the-art computational fluid dynamics simulation library to give CFD engineers access to a plethora of solvers and preconditioners CFD engineers. Using three application test cases, we demonstrate how picking numerical methods from the Ginkgo library can accelerate simulations on supercomputers featuring NVIDIA's Ampere GPUs and Grace Hopper superchips.

Keywords: Spectral Element Methods; GPU; linear solver

1. Introduction

Computational fluid dynamics (CFD) simulations drive a large portion of high-performance research and development. Based on a discretization of the Navier-Stokes equations, CFD simulations attempt to simulate the free-stream flow and the interaction between the flows and gases with applications ranging from airplanes, reactors and combustion to ocean and atmospheric flows as well as astrophysical problems. CFD simulations resolving all relevant physical scales are called direct numerical simulations (DNSs), but they are not possible for all applications due to computational cost. Reduced order methods (ROMs), such as Reynolds-averaged Navier Stokes (RANS) and large-eddy simulation (LES), can be used as cheaper alternatives. However, their accuracy strongly depends on the availability of

* Corresponding author.
E-mail address: hartwig.anzt@tum.de

1877-0509
10.1016/j.procs.2025.08.234

suitable closure models. For handling these models on supercomputers, CFD libraries need access to fast and robust numerical methods that are designed for efficient execution on modern hardware.

Virtually all modern supercomputers are GPU-centric. That means that a significant portion of the computing performance is provided by the GPUs of a system. For example, on the latest TOP500 list ranking the supercomputers according to their HPL performance on FP64 precision, only one of the top ten systems is not GPU-centric. Although the performance and memory bandwidth of GPUs have both grown quickly in recent years driven by applications such as artificial intelligence, the communication between host memory and GPUs' memory is still often a performance bottleneck. Thus, realizing all numerical computations of a CFD simulation on the GPUs of a system without any data movement between the GPUs and the CPUs has become the dominant approach in high-performance computing.

In this paper, we integrate the high-performance numerical linear algebra library Ginkgo as a math toolbox into the state-of-the-art CFD library nekRS. Ginkgo provides a plethora of performance-portable direct and iterative linear solvers, preconditioners, and algebraic multigrid (AMG) that are efficient in the solution of problems arising in LES and RANS turbulence models. After providing some background about the nekRS and Ginkgo libraries and their usage in Section 2, we detail in Section 3 how configuration files can be used to instruct nekRS to employ a specific solver + preconditioner combination from the Ginkgo. We then demonstrate in Section 4 that employing Ginkgo functionality in nekRS simulations can indeed render attractive runtime savings on GPU-centric supercomputers. In Section 5, we summarize the results and how the approach taken in the Ginkgo integration transfers to other simulation software stacks.

2. Background

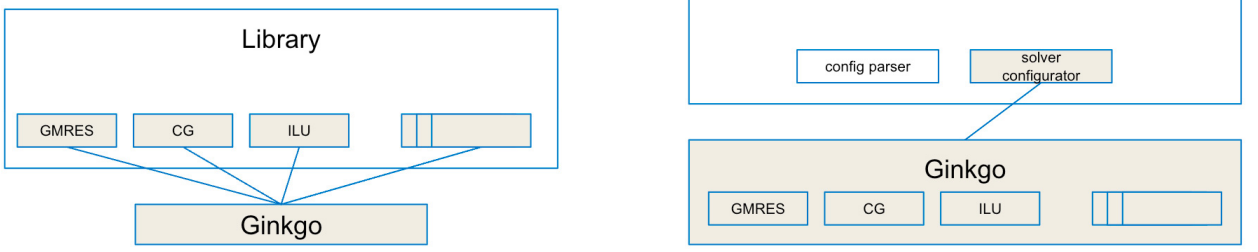
nekRS[15, 6] is a CFD library based on libparanumal[10, 3] and Nek5000[14]. It mainly uses OCCA[11] as a portability layer to support different vendors' hardware. nekRS starts from the spectral element method (SEM), which combines high accuracy with flexibility concerning flow geometry. A high-order SEM approximates the solution and data in terms of locally structured Nth-order tensor product polynomials on a set of globally unstructured elements. Thus, in addition to exponential convergence for smooth solutions with increasing polynomial order, which provides high accuracy at lower computational cost, it offers the flexibility to handle complex geometries via domain decomposition. As a result, nekRS is well suited for the efficient simulation of turbulence, where the number of grid points grows faster than quadratically with the Reynolds number when all flow features must be resolved.

nekRS highly optimizes the components for their simulation. They especially apply their domain knowledge to accelerate the solving process based on the problems' properties. nekRS usually uses p-multigrid as a preconditioner, which uses different approximate orders on the problem for different multigrid levels. In multigrid, nekRS needs a solver to solve the coarse problem of p-multigrid. Previously, nekRS used HYPRE[5, 17] and NVIDIA AmgX[13] as the coarse solver options. This paper introduces Ginkgo as another coarse solver option, which brings more solvers into the nekRS stack.

Ginkgo[1] is a software library focusing on the efficient solution of sparse linear systems on GPU-centric supercomputers. The software features multiple backends in hardware-native languages: CUDA for NVIDIA GPUs, HIP for AMD GPUs, and SYCL for Intel GPUs in [4]. Also, Ginkgo considers software sustainability to be of high priority, so all functionality is unit-tested against a sequential reference implementation and employs continuous integration (CI) to ensure the correctness of compilation and execution. Ginkgo contains a plethora of iterative solvers, including Krylov solvers, direct solvers, algebraic multigrid (AMG), and parallel preconditioners like block-Jacobi, incomplete LU factorization (ILU) and parallel variants (ParILU), ILU with thresholding, BDDC preconditioning, and batched routines for efficient data-parallel processing.

3. Leveraging Ginkgo in nekRS

Integrating Ginkgo as a math toolbox into a simulation software stack is one thing, but easing the use of Ginkgo's functionality in simulation runs is a totally different aspect that is equally important to facilitate the hassle-free toolbox usage for domain scientists who are primarily focused on the CFD applications. To facilitate the hassle-free usage of Ginkgo's functionality, we developed a configuration file concept that allows encoding all solver and preconditioner



(a) Library needs to expose the numerical methods available in the math toolbox, then configure the solver in the application code.

(b) A single interface to the toolbox is complemented with a configuration file that encodes the numerical method selection outside the source code.

Fig. 1: Math toolbox integration options.

Listing 1: CG solver from configuration file for selecting the numerical methods and configuring the parameters.

```
{
  "type": "solver::Cg",
  "criteria": [
    {"type": "Iteration", "max_iters": 4},
    {"type": "ImplicitResidualNorm",
     "reduction_factor": 1e-4}
  ]
}
```

settings in a file. Prior to introducing this concept, the simulation software needed to expose all of Ginkgo's solver and preconditioner options internally to the users, see Figure 1a. The configuration file concept removes this burden by outsourcing all solver and preconditioner configurations into a file. The simulation software only needs to include one interface to the math toolbox (in this case: Ginkgo) and extracts the specific numerical method configuration from the configuration file, see Figure 1b. This concept has several advantages: 1) It dramatically eases the maintenance of a library interface as new methods do not need to be exposed in the simulation software stack but simply become available as an option in the file configuration, and 2) it eases the toolbox usage for the domain scientists as they can easily swap solvers and preconditioners and options in the configuration file without even touching the code or re-compilation.

To provide an example for the configuration file usage, we configure the JSON in Listing 1 to select a Conjugate Gradient (CG) solver with an iteration limit of four iterations and a relative residual stopping criterion of 10^{-4}. Not significantly more complicated is the configuration of an AMG solver based on Parallel Graph Match (PGM[13]) as the coarsening method and a CG coarse grid solver for the bottom-level system of size smaller than 512 in Listing 2.

The nekRS domain scientist instructs nekRS to use Ginkgo by modifying the nekRS run file, see Listing 3: Specifying `ginkgo` in `coarseSolver` instructs nekRS to the generic interface to Ginkgo in the coarse solver of the nekRS multigrid, the path specified in the `[GINKGO]` section of the runfile provides the path to a Ginkgo configuration file like Listing 2. Domain scientists can quickly change the Ginkgo solver configuration by modifying the Ginkgo configuration file or changing the path to the configuration file in the nekRS run file.

4. Experiments Result

With Ginkgo being integrated into nekRS and the configuration file enabling the hassle-free use of Ginkgo's numerical methods in nekRS, the remaining question is whether domain scientists can benefit from utilizing Ginkgo functionality in nekRS simulations. To demonstrate the practical benefits the Ginkgo toolbox brings to nekRS, we focus on three scientifically relevant benchmark applications and their solution on the latest GPU-centric clusters at the Jülich Supercomputing Centre (JSC). The two supercomputers we run on are the JURECA-DC GPU system, which

Listing 2: AMG solver from configuration file for selecting the numerical methods and configuring the parameters.

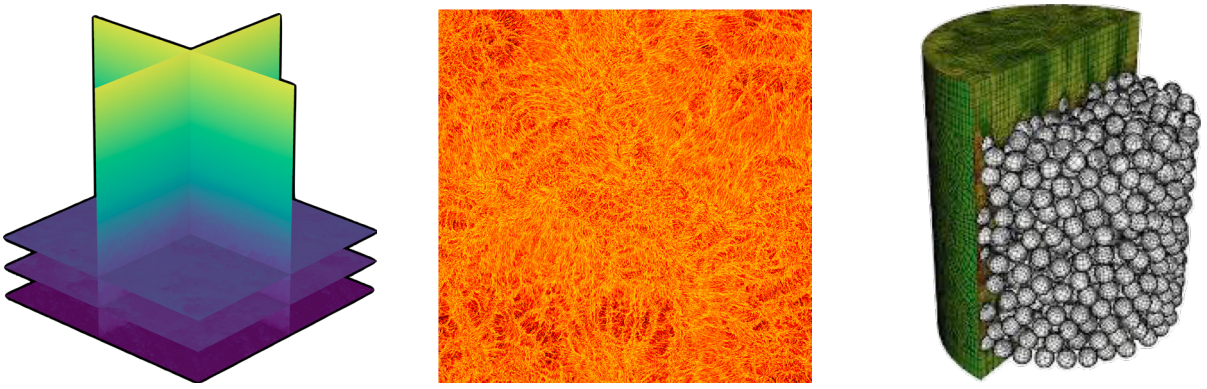
```
{
  "type": "solver::Multigrid",
  "min_coarse_rows": 512,
  "mg_level": [
    {
    "type": "multigrid::Pgm", "deterministic": true
    }
    ],
  "coarsest_solver": {
    "type": "solver::Cg",
    "criteria": [
      {"type": "Iteration", "max_iters": 4},
      {"type": "ImplicitResidualNorm",
       "reduction_factor": 1e-4}
    ]
  },
  "criteria": [
        {"type": "Iteration", "max_iters": 1}
  ]
}
```

Listing 3: Using Ginkgo as the coarse solver and set the ginkgo solver from the file.

```
[PRESSURE]
...
preconditioner = multigrid
coarseSolver = ginkgo
...
[GINKGO]
configFile = <path to cg.json or mg.json>
```

is based on 4 A100 GPUs hosted by 2 AMD EPYC 7742 CPUs per node, and the JEDI system, which is composed of 4 Grace Hopper superchips per node. Each Superchip integrates a 72-core Grace CPU and a Hopper GPU (96 GB HBM3) connected via NVLink-C2C (900 GB/s). JEDI is the development system for the first European Exascale supercomputer JUPITER. In fact, it features one future JUPITER rack of hardware. We compile the nekRS and Ginkgo codes with GCC 12.3.0, CUDA 12.2, and OpenMPI 4.1.6 with GPU-aware features on these machines.

Different CFD problems can be numerically very different. E.g. the pressure solve, which is usually the most critical part, can be simpler or more difficult to solve from different problems. The benchmark applications are 1) GABLS in Figure 2a - GEWEX Atmospheric Boundary Layer Study[8, 7] focusing on boundary layer turbulence and near surface process on the stable boundary layers over ground and under clear skies; 2) RBC in Figure 2b - Rayleigh-Bénard convection[16, 2] focusing on the natural convection in the container when heating the container; and 3) PB in Figure 2c - Pebble Bed[9] focusing on the flow past non-contacting, randomly-packed, equally sized spheres staked in cylindrical or annular container. Also, we collect the problem size and corresponding polynomial order in Table 1. Additionally, we collect the global size of the coarse problem from nekRS's multigrid which is solved by Ginkgo in the last column of Table 1. nekRS almost distributes the problem equally to GPUs, so each GPU handles $\approx$ {the total number of elements} / N elements when we run the problem on N GPUs. Because the problem in the pressure solve can be varied in terms of difficulty, we pick Ginkgo's CG and AMG solvers with the configurations listed in Listing 1 and Listing 2, respectively for the comparison against the nekRS-internal solvers. nekRS specifics single precision for the coarse solver, so Ginkgo's solver also uses single precision to set the solver.



(a) GABLS (b) RBC (c) PB from the paper[12]

Fig. 2: The application visualization.

	polynomial order	the total number of elements	global problem size from nekRS to Ginkgo
GABLS_64	8	262,144	266,240
GABLS_128	8	2,097,152	2,113,536
RBC_40	7	2,160,000	2,182,500
RBC_80	9	2,160,000	2,182,500
ann3344	7	1,080,003	1,238,974
cyl11k	8	3,575,076	4,096,583

Table 1: The application size

4.1. GABLS cases

We report the performance of GABLS_64 (64^3 discretization) and GABLS_128 (128^3 discretization) on JURECA-DC GPU in Figure 3 and Figure 4. The node count is chosen to fulfill the weak scaling requirement of the local problem size remaining constant for both node counts. The parallel efficiency for the weak scaling experiment is visualized in Figure 5. The GABLS benchmark cases are relatively easy to solve, and Ginkgo CG outperforms both Ginkgo AMG and nekRS smoother. For the largest node count, Ginkgo CG achieves a 1.1× speedup against nekRS smoother for GABLS_64 in Figure 3 and 1.12× speedup for GABLS_128 in Figure 4. In terms of parallel efficiency, Ginkgo and nekRS are comparable, reaching about 70% parallel efficiency, see Figure 5. We collect the memory consumption per GPU from the report of nekRS smoother settings:

- GABLS_64: 17.11 GB per GPU when using 16 A100, 9.83 GB per GPU when using 28 A100, and 4.93 GB per GPU when using 56 A100.
- GABLS_128: 19.61 GB per GPU when using 112 A100, 9.97 GB per GPU when using 220 A100, and 5.02 GB per GPU when using 440 A100.

Besides the performance results on A100 on JURECA, we also perform the same experiments on JEDI to assess the performance gains coming from NVIDIA's Grace Hopper architecture. We collect the performance data of GABLS_128 on both machines in Figure 6. Overall, we see a 2× speedup when moving from the A100-powered supercomputer to the Grace Hopper superchips system. This speedup matches the experience from the other applications on JEDI. Grace Hopper superchips equip more memory than A100, so we only use a half the number of machine nodes of JURECA-DC GPU for the experiments on Grace Hopper superchips. For example, we set 14 machine nodes (56 Grace Hopper superchips) on JEDI from 28 machine nodes (112 A100) on JURECA-DC GPU. Because we set the number based on the number of machine nodes, we set 27 machine nodes by rounding down from the number $\frac{55}{2}$. We set up 28 machine nodes on JEDI, which contains 112 Grace Hopper superchips, rather than 55 machine nodes

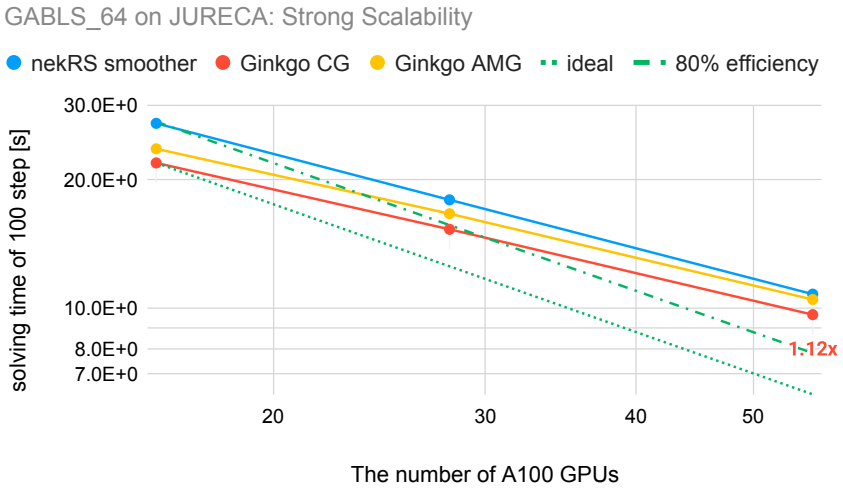


Fig. 3: The performance of GABLS_64 (64^3 discretization) on JURECA-DC GPU.

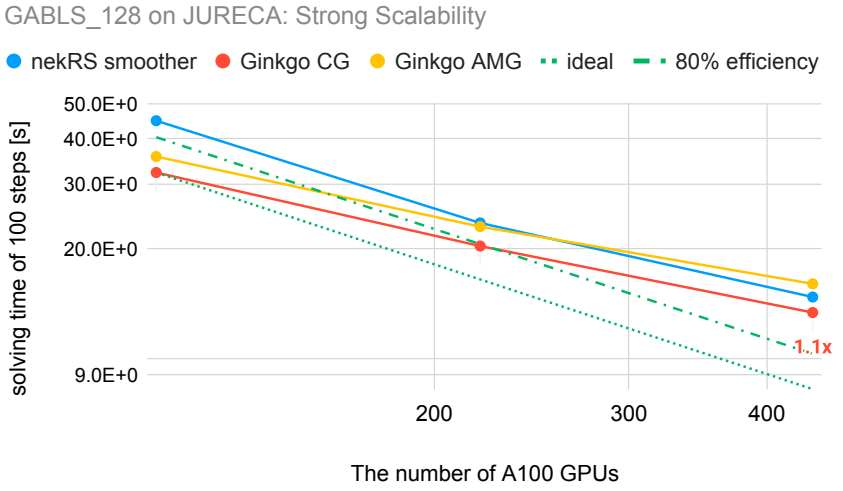


Fig. 4: The performance of GABLS_128 (128^3 discretization) on JURECA-DC GPU.

by reducing half from 110 machine nodes on JURECA-DC GPU because it is the maximum number of nodes we can request. We collect the memory consumption per GPU from the report of nekRS smoother settings:

- GABLS_128: 38.97 GB per GPU when using 56 Grace Hopper superchips, 20.30 GB per GPU when using 108 Grace Hopper superchips, and 19.61 GB per GPU when using 112 Grace Hopper superchips.

4.2. RBC cases

We next use the same hardware configurations for the RBC benchmark applications. The suffix number of RBC indicates the minimal machine node requirement on JURECA-DC GPU. RBC_40 uses polynomial order 7 and a Rayleigh number to 10^8, but RBC_80 uses polynomial order 9 and a Rayleigh number to 10^9. In this performance

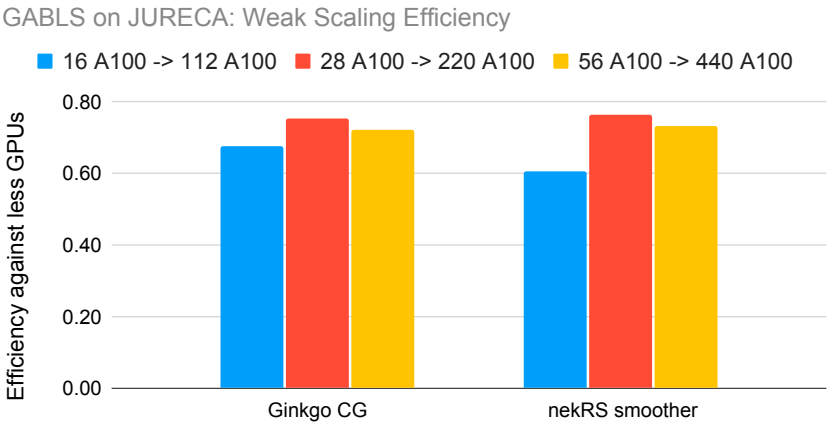


Fig. 5: The weak scaling efficiency of GABLS (from 64^3 to 128^3) on JURECA-DC GPU.

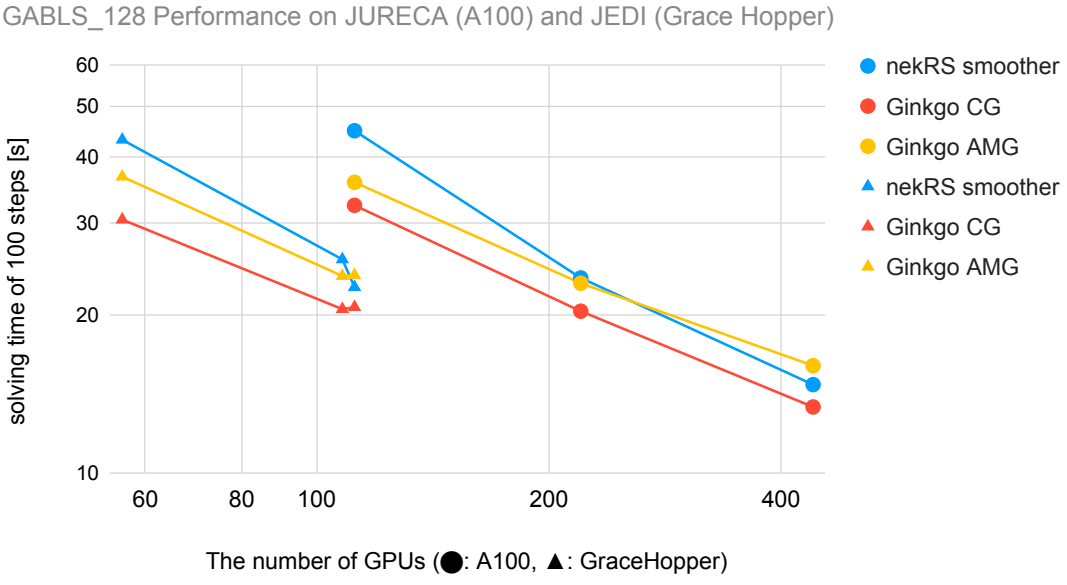


Fig. 6: The performance of GABLS_128 (128^3 discretization) on JURECA-DC GPU and JEDI.

experiment, we increment the node count by steps of 40 until we run on the full JURECA-DC GPU machine. In Figure 7 we see that all solvers perform similarly for the RBC_40 case. Ginkgo CG shows 1.15× speedup over nekRS smoother on 160 nodes (640 A100) on JURECA-DC GPU in Figure 8. For GABLS(128^3) Figure 4, nekRS smoother outperforms Ginkgo AMG for large GPU counts, but Ginkgo AMG starts outperforming nekRS smoother for large node counts for the RBC_80 benchmark application, see Figure 8. We collect the memory consumption per GPU from the report of nekRS smoother settings:

- RBC_40: 11.59 GB per GPU when using 160 A100, 5.89 GB per GPU when using 320 A100, 3.89 GB per GPU when using 480 A100, and 2.92 GB per GPU when using 640 A100.
- RBC_80: 11.34 GB per GPU when using 320 A100, 7.58 GB per GPU when using 480 A100, and 5.70 GB per GPU when using 640 A100.

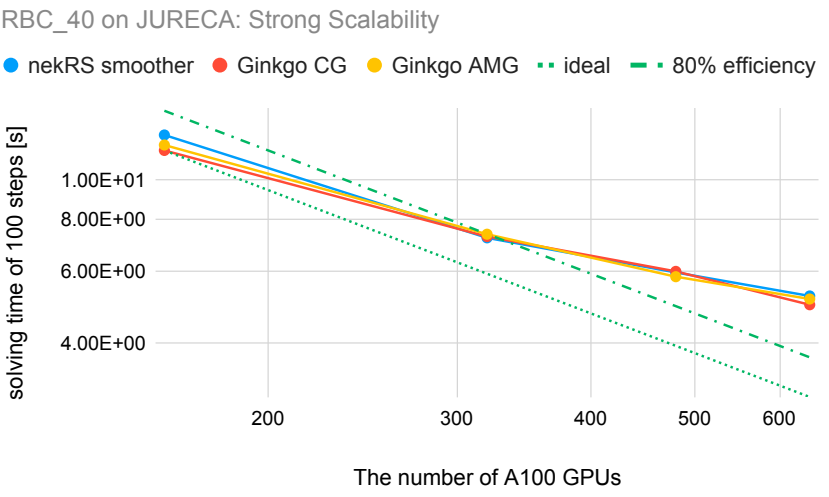


Fig. 7: The performance of RBC_40 on JURECA-DC GPU.

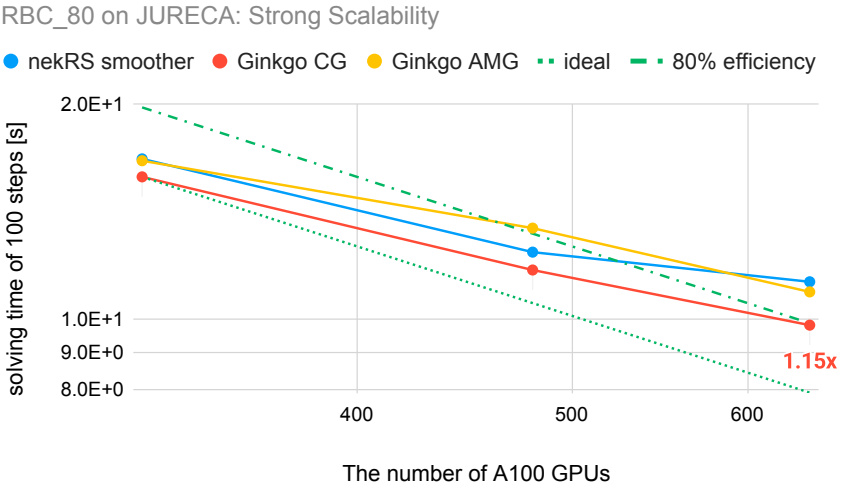


Fig. 8: The performance of RBC_80 on JURECA-DC GPU.

4.3. PB cases

We now address a numerically challenging problem coming from the PebbleBed benchmark application. The ann3344 problem contains 3,344 spheres in an annular container, and the cyl11k problem contains 11,145 spheres in a cylindrical container. The solver runtime visualization for the pebble bed series problem reveals that Ginkgo AMG largely outperforms Ginkgo CG and nekRS smoother: Ginkgo AMG outperforms nekRS smoother by 3× for the ann3344 test case and 8× for the larger cyl11k test case in Figure 9. These speedups stem from the significant convergence acceleration: the AMG reduces the iteration count by more than 8×, see Figure 10. Ginkgo CG seems superior to Gingko AMG and nekRS smoother for GABLS and RBC cases because the problems are relatively easy to solve. For PB cases, Ginkgo AMG shows clear advantages against Ginkgo CG and nekRS smoother. We collect the memory consumption per GPU from the report of nekRS smoother settings:

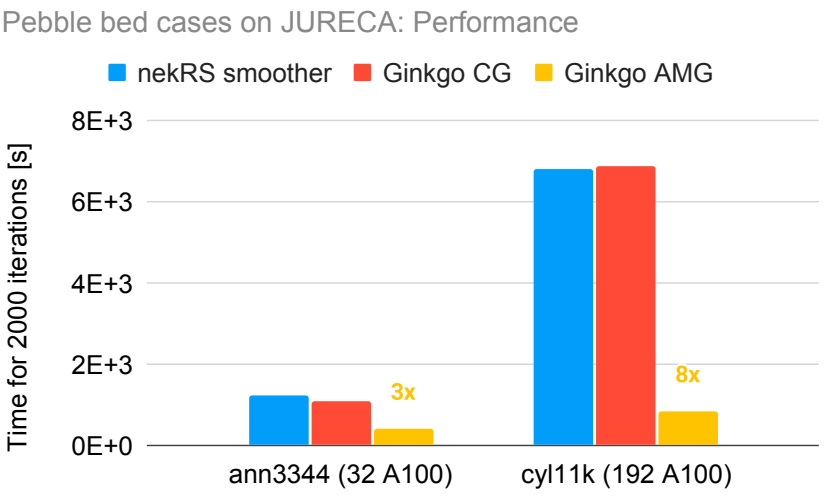


Fig. 9: The performance of pebble bed cases on JURECA-DC GPU.

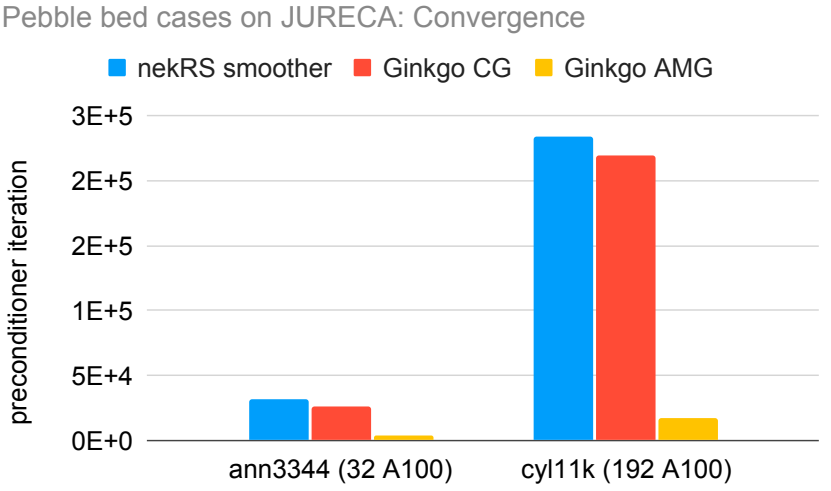


Fig. 10: The convergence of pebble bed cases on JURECA-DC GPU.

- ann3344: 36.37 GB per GPU when using 32 A100.
- cyl11k: 32.80 GB per GPU when using 192 A100.

5. Conclusion

Using the Ginkgo library as an example, we demonstrated how a lightweight integration into the nekRS library can give domain scientists easy access to a plethora of numerical methods in the Ginkgo toolbox. The use of a configuration file workflow enables quick testing of different methods and their suitability for a given application and hardware setup. We furthermore demonstrated that the CG and AMG methods available in Ginkgo through this lightweight interface can offer attractive runtime savings, particularly for numerically challenging problems. While we

do not claim that the performance we demonstrate on the latest GPU architecture from NVIDIA cannot be matched by other solvers or libraries, we are convinced that the integration strategy is a blueprint for other math toolbox integrations as it combines sustainability through low maintenance and productivity through the hassle-free usage by domain scientists.

Acknowledgements

This research is supported by the Inno4Scale project under Inno4scale-202301-099. Inno4Scale has received funding from the European High Performance Computing Joint Undertaking (JU) under grant agreement No 101118139. The JU receives support from the European Union's Horizon Europe Programme. The authors acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputers at Jülich Supercomputing Centre (JSC) and the JUPITER Research and Early Access Program (JUREAP).

References

[1] Anzt, H., Cojean, T., Flegar, G., Göbel, F., Grützmacher, T., Nayak, P., Ribizel, T., Tsai, Y.M., Quintana-Ortí, E.S., 2022. Ginkgo: A modern linear operator algebra framework for high performance computing. ACM Transactions on Mathematical Software (TOMS) 48, 1–33.

[2] Bode, M., Alvarez, D., Fischer, P., Frouzakis, C.E., Göbbert, J.H., Insley, J.A., Lan, Y.H., Mateevitsi, V.A., Min, M., Papka, M.E., et al., 2025. Deciphering boundary layer dynamics in high-rayleigh-number convection using 3360 gpus and a high-scaling in-situ workflow. arXiv preprint arXiv:2501.13240 .

[3] Chalmers, N., Karakus, A., Austin, A.P., Swirydowicz, K., Warburton, T., 2022. libParanumal: a performance portable high-order finite element library. URL: https://github.com/paranumal/libparanumal, doi:10.5281/zenodo.4004744. release 0.5.0.

[4] Cojean, T., Tsai, Y.H.M., Anzt, H., 2022. Ginkgo—a math library designed for platform portability. Parallel Computing 111, 102902.

[5] Falgout, R.D., Yang, U.M., 2002. hypre: A library of high performance preconditioners, in: International Conference on Computational Science, Springer. pp. 632–641.

[6] Fischer, P., Kerkemeier, S., Min, M., Lan, Y.H., Phillips, M., Rathnayake, T., Merzari, E., Tomboulides, A., Karakus, A., Chalmers, N., Warburton, T., 2022. Nekrs, a gpu-accelerated spectral element navier–stokes solver. Parallel Computing 114, 102982. URL: https://www.sciencedirect.com/science/article/pii/S0167819122000710, doi:https://doi.org/10.1016/j.parco.2022.102982.

[7] Holtslag, A., 2006. Gewex atmospheric boundary-layer study (gabls) on stable boundary layers. Boundary-Layer Meteorol 118, 234–246.

[8] Kosović, B., Curry, J.A., 2000. A large eddy simulation study of a quasi-steady, stably stratified atmospheric boundary layer. Journal of the atmospheric sciences 57, 1052–1068.

[9] Lan, Y.H., Fischer, P., Merzari, E., Min, M., 2021. All-hex meshing strategies for densely packed spheres. arXiv preprint arXiv:2106.00196 .

[10] libparanumal, . libparanumal. [Online] https://github.com/paranumal/libparanumal.

[11] Medina, D.S., St-Cyr, A., Warburton, T., 2014. Occa: A unified approach to multi-threading languages. arXiv preprint arXiv:1403.0968 .

[12] Merzari, E., Yuan, H., Min, M., Shaver, D., Rahaman, R., Shriwise, P., Romano, P., Talamo, A., Lan, Y.H., Gaston, D., et al., 2021. Cardinal: A lower-length-scale multiphysics simulator for pebble-bed reactors. Nuclear Technology 207, 1118–1141.

[13] Naumov, M., Arsaev, M., Castonguay, P., Cohen, J., Demouth, J., Eaton, J., Layton, S., Markovskiy, N., Reguly, I., Sakharnykh, N., et al., 2015. AmgX: A library for GPU accelerated algebraic multigrid and preconditioned iterative methods. SIAM Journal on Scientific Computing 37, S602–S626.

[14] nek5000, . nek5000. [Online] https://github.com/Nek5000/Nek5000.

[15] nekRS, . nekRS. [Online] https://github.com/Nek5000/nekRS.

[16] Samuel, R.J., Bode, M., Scheel, J.D., Sreenivasan, K.R., Schumacher, J., 2024. No sustained mean velocity in the boundary region of plane thermal convection. Journal of Fluid Mechanics 996, A49.

[17] Yang, U.M., et al., 2002. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics 41, 155–177.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 82–91

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

An efficient multigrid solver for finite element methods on multi-GPU systems

Chris N. Richardson[a,*], Igor A. Baratta[a], Joseph P. Dean[a], Adrian Jackson[b], Garth N. Wells[a]

[a]*Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, United Kingdom*
[b]*Edinburgh Parallel Computing Centre, Bayes Centre, 47 Potterrow, Edinburgh, EH8 9BT, United Kingdom*

Abstract

We present a matrix-free p-multigrid method for solving the Poisson problem using the finite element method on GPU-based supercomputers. The solver uses optimised matrix-free kernels to compute the action of arbitrary-degree finite element operators, matrix-free interpolation between finite element spaces of different polynomial degrees, and parallel vector updates, all operating on the GPU. On the coarsest level, we use an algebraic multigrid solver. We examine performance and scalability for the Poisson problem on a number of different meshes. Numerical experiments on the LUMI supercomputer show good parallel performance.

Keywords: Finite Element; Multigrid; GPU; HPC; iterative methods

1. Introduction

Solving elliptic and parabolic partial differential equation problems, such as heat conduction and elasticity, on GPU architectures poses a number of challenges. Firstly, implicit solution methods are required, which would usually involve the assembly of a sparse matrix. High-order finite element methods are usually preferred on GPUs over more commonly used low-order methods because the kernels have higher arithmetic intensity, which potentially allows a greater fraction of the floating point performance of the GPU to be utilised. However, the performance of sparse matrix implementations degrades with increasing polynomial degree, with an evaluation complexity of $O(p^d)$ per matrix row [8] for p-degree elements in d-dimensional space. Moreover, scalable solvers typically perform many matrix–vector products, and sparse matrix–vector products have very low arithmetic intensity, meaning that they only achieve a percent or two of peak compute performance on modern hardware [2]. To compound these challenges, large problems require efficient solvers with optimal cost complexity of $O(n)$, where n is the number of degrees-of-freedom (DoFs), to be tractable to solve. Multigrid-preconditioned Krylov subspace methods can provide optimal solvers, and

* Corresponding author. Tel.: +44 1223 765700
E-mail address: cnr12@cam.ac.uk

1877-0509
10.1016/j.procs.2025.08.235

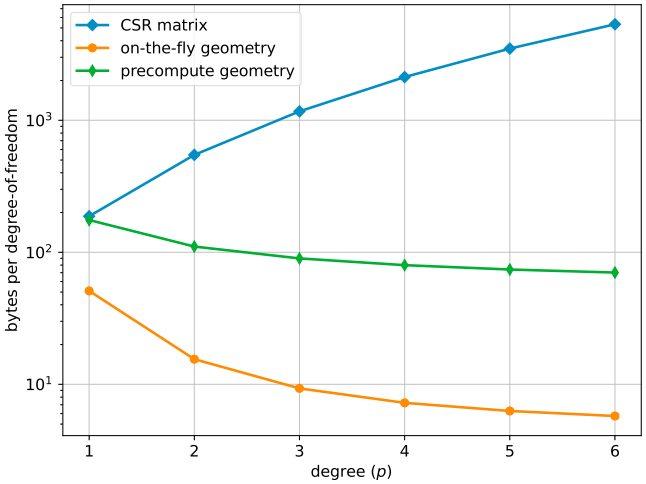


Fig. 1: Memory requirement per degree-of-freedom for different implementations of the Laplacian operator for Lagrange elements on hexahedral cells of different polynomial degrees. First-order cell geometry is assumed.

implementations of algebraic multigrid (AMG) methods for GPUs are available [9, 6]. However, the performance of AMG methods degrades at higher-orders; they become increasingly expensive and converge more slowly [2].

Matrix-free finite element methods have attracted considerable attention on GPU architectures to circumvent the issues associated with sparse matrices. The main operation in Krylov methods is the computation of a sparse matrix–vector product, i.e. computation of the *action* $y = Ax$ of a matrix (operator) A on a vector x. For finite element methods, the action of the operator can easily be computed directly without forming a sparse matrix. By exploiting the structure of tensor product elements and using sum-factorisation techniques, the evaluation complexity reduces to $O(dp)$ per DoF [8]. Moreover, the amount of data required is lower than for a sparse matrix for $p \geq 2$ [8]. To highlight differences in memory requirements for assembled sparse matrices and matrix-free methods, fig. 1 shows the storage required per DoF for the Poisson operator for (i) an assembled CSR matrix, (ii) computing the action of the Laplacian operator with lowest-order geometry cells and with the geometry data computed 'on-the-fly' and (iii) computing the action of the Laplacian with pre-computed geometry data stored at quadrature points. Note that at higher orders the memory required for an assembled matrix is significantly greater than the memory required for the matrix-free approach.

While matrix-free methods can allow GPUs to take advantage of the higher arithmetic intensity of higher-order elements whilst avoiding some of the drawbacks of sparse matrices, the absence of an assembled matrix operator can make preconditioning difficult; for example, algebraic multigrid methods normally construct the preconditioner from an assembled operator matrix. One approach, which we explore here, is to use p-multigrid with the finite element method [11], where coarsening is achieved by reducing the element polynomial degree. On the coarsest level, linear elements can be used, for which a matrix can be assembled and for which algebraic multigrid performs well. Matrix-free operators can be used on the finer levels.

Kronbichler and Ljungkvist [8] developed a parallel GPU accelerated geometric multigrid solver for the Poisson equation. Their approach employs matrix-free operators, tensor-product cells, and sum factorisation. They compare the performance of their GPU implementation, running on NVIDIA P100 GPUs, with an optimised CPU implementation on a comparable CPU. The GPU implementation is approximately 1.5–2 times faster. Brown et al. [2] demonstrate that even for real engineering problems, where solutions often exhibit singularities due to boundary conditions/complex geometry, etc., high-order p-multigrid schemes on GPUs can still significantly outperform industry-standard low-order alternatives. They focus on hyperelastic simulations of multiscale structures, and present performance results on the AMD MI250x, NVIDIA A100 and NVIDIA P100 GPUs for problems with billions of DoFs.

We develop and test a p-multigrid finite element solver for the Poisson problem on GPUs, targeting the GPU partition of the LUMI supercomputer, which is equipped with AMD MI250x GPUs. The method uses unstruc-

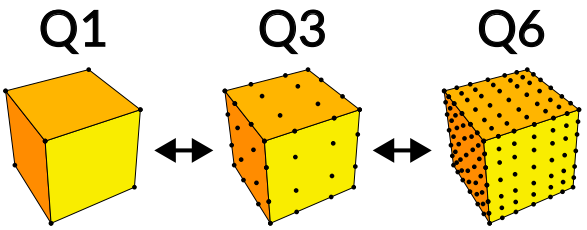


Fig. 2: A p-multigrid hierarchy on hexahedral cells.

tured meshes of hexahedral cells, and uses sum-factorisation techniques for fast integration and interpolation. The source code used in this paper is available under an open-source license at `https://github.com/Wells-Group/pmg-dolfinx-lumi`. The code can also be compiled for CUDA and has been tested on NVIDIA A100 and GH200 GPUs.

2. p-multigrid algorithm

We consider the homogeneous Poisson problem, given by

$$-\nabla^2 u = f \quad \text{in } \Omega \subset \mathbb{R}^3, \tag{1}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{2}$$

where u is the unknown field and f is prescribed. We discretise eq. (1) using a standard conforming Lagrange finite element method using hexahedral cells.

The discrete systems generated by the finite element method for eq. (1) can be solved efficiently by a multigrid method. In the p-multigrid method, the polynomial degree (and therefore the problem size) is reduced when moving from the 'fine grid' (the solution level grid), through intermediate grids, to the 'coarse grid'. The mesh (triangulation) at each level is the same; we use the multigrid jargon 'grid' to indicate resolution and problem size level, rather than a grid size. Figure 2 illustrates a three-level p-hierarchy of finite elements. The coarse, intermediate, and fine levels use degree one ('Q1'), three ('Q3'), and six ('Q6') Lagrange elements, respectively. In this approach, the problem size on the coarse level grid (level 0) may still be a very large problem, and too large for the efficient use of a direct solver. We therefore use an algebraic multigrid solver for the coarse grid. We note that algebraic multigrid methods are observed to perform well for the lowest-degree elements, and their performance degrades with increasing element degree. We use a Q1 geometric map for cells on all levels.

The choice of the number of levels and the polynomial degree for each level is somewhat heuristic. Our choice of levels Q6 – Q3 – Q1 (three levels, see fig. 2) was guided by some simple numerical tests which showed that this choice gave the lowest computational cost on a simple geometry. Theoretical approaches are possible [12], but in practice the above choice works well for the problems we have considered.

A three-level p-multigrid procedure is presented in algorithm 1. The prolongation operator $\mathcal{P}_i$ takes a field on the ith level and interpolates it on the finer $i+1$ level. This is straightforward with Lagrange elements, since the degrees-of-freedom are defined via pointwise evaluations of the field being interpolated. The finite element space on the $i+1$ level contains the space on level i, hence the interpolation is exact. The restriction operator $\mathcal{R}_i$, which restricts to the ith level a field defined on the finer level $i+1$, is taken to be the transpose of the prolongation, $\mathcal{P}_i^T$. On the coarse level (level 0), lowest-degree continuous Lagrange elements are used, and the matrix operator A_0 is formed and is solved using an algebraic multigrid method. On all other levels, we follow a matrix-free approach where the action of a matrix operator A on a vector is computed — the (sparse) matrix A is never formed.

Algorithm 1 Three level p-multigrid V-cycle for solving the fine-level problem $A_2 u_2 = b_2$. The initial estimate of the solution u_2 is improved on each application of this algorithm.

$u_2 \leftarrow \text{Smooth}(A_2, u_2, b_2)$ ▷ Smooth solution, see algorithm 3
$r_1 \leftarrow \mathcal{R}_1(b_2 - A_2 u_2)$ ▷ Restrict residual → level 1
$u_1 \leftarrow \text{Smooth}(A_1, u_1, r_1)$
$r_0 \leftarrow \mathcal{R}_0(r_1 - A_1 u_1)$ ▷ Restrict level 1 residual → level 0
$u_0 \leftarrow \text{AMG}(\text{A}_0, \text{r}_0)$ ▷ Algebraic multigrid solver on coarse level
$u_1 \leftarrow u_1 + \mathcal{P}_0(u_0)$ ▷ Prolongate level 0 correction → level 1
$u_1 \leftarrow \text{Smooth}(A_1, u_1, r_1)$
$u_2 \leftarrow u_2 + \mathcal{P}_1(u_1)$ ▷ Prolongate level 1 correction → level 2
$u_2 \leftarrow \text{Smooth}(A_2, u_2, b_2)$

In the remainder of this section, we consider in more detail (i) the restriction and prolongation operations, (ii) the smoothing (relaxation) step and (iii) the computation of the operator (matrix) action.

2.1. Prolongation and restriction

Prolongation and restriction operations applied to residual vectors could be implemented using assembled sparse matrices. However, we aim here to avoid forming sparse matrices where possible. The bandwidth of the matrices operating between the Q6 and Q3 levels is substantial. Moreover, computation with the transpose of the prolongation matrix (to get the restriction matrix) is not straightforward in a distributed memory (multi-GPU) setting. We therefore implemented 'matrix-free' interpolation (restriction) and prolongation (algorithm 2). Prolongation (the coarse-to-fine operator, $\mathcal{P}$) is straightforward to implement cell-wise, since the fine DoFs only depend on coarse values from the same cell. However, the transpose operation (the fine-to-coarse restriction operator, $\mathcal{R}$) requires summation at the coarse degrees-of-freedom and division by a 'multiplicity' factor, which is the number of cells that a DoF is shared with.

Algorithm 2 Cell-wise restriction/prolongation operator from the degree p_0 input space to the degree p_1 output space using sum factorisation. ϕ is the (rectangular) 1D interpolation matrix from the input to the output space.

Copy input cell DoFs to u_{ijk} ▷ Input data is conceptually stored an indexed 3D block of size $(p_0 + 1)^3$.
if restriction **then**
    $u_{ijk} \leftarrow u_{ijk} / M_{dof}$ ▷ Input DoFs are divided by the "multiplicity"
$\hat{u}^0_{ijk} \leftarrow \sum_\alpha \phi_{i\alpha} u_{\alpha jk}$ ▷ Interpolate in x
$\hat{u}^1_{ijk} \leftarrow \sum_\alpha \phi_{j\alpha} \hat{u}^0_{i\alpha k}$ ▷ Interpolate in y
$v_{ijk} \leftarrow \sum_\alpha \phi_{k\alpha} \hat{u}^1_{ij\alpha}$ ▷ Interpolate in z
if restriction **then**
    atomicAdd(v_{ijk}) to output DoFs ▷ Output data is stored in a 3D block of size $(p_1 + 1)^3$.
else
    Copy v_{ijk} to output DoFs

Figure 3 shows the throughput (in DoFs processed per second) of the restriction and prolongation operators against the total number of DoFs on a single Graphics Compute Die (GCD) of the AMD MI250x GPU. Overall, restriction and prolongation operations make up only a very small fraction (a few percent) of the total time taken for the multigrid cycle. This is quantified in section 4.

2.2. Relaxation (smoothing)

On each level, the solution is smoothed using matrix-free Chebyshev iteration [10] with polynomials of the fourth kind, see algorithm 3. The required operator eigenvalue upper bound estimates are computed using a few Conjugate

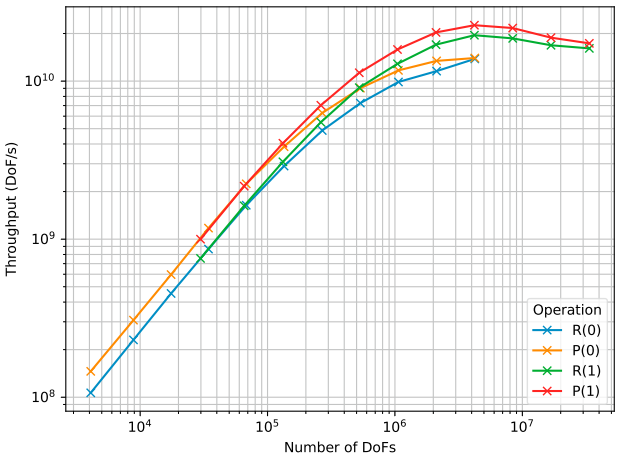


Fig. 3: Restriction and prolongation throughput for R(0)[Q3→Q1], R(1)[Q6→Q3], P(0)[Q1→Q3], and P(1)[Q3→Q6], against input vector size on one GCD of an MI250X on a cube mesh.

Gradient iterations on each level at setup time. To accelerate convergence, the Chebyshev iterations use a Jacobi preconditioning step. Jacobi preconditioning involves division by the matrix diagonal of the operator matrix. Since we do not wish to assemble the operator matrix, we have implemented a GPU kernel that precomputes the matrix diagonal entries for each level. The number of Chebyshev iterations used (n in the algorithm) can be increased for stronger smoothing.

Algorithm 3 Chebyshev iterations of the fourth kind [10].

$r \leftarrow b - Au$ ▷ Initial residual

$z \leftarrow \frac{4}{3\lambda_{\max}} M^{-1} r$ ▷ λ_{max} is the maximum eigenvalue estimate of A

for k in range$(1, n+1)$ **do**

  $u \leftarrow u + z$

  $r \leftarrow r - Az$

  $z \leftarrow \frac{2k-1}{2k+3} z + \frac{8k+4}{(2k+3)\lambda_{\max}} M^{-1} r$ ▷ M^{-1} is a simple preconditioner of A

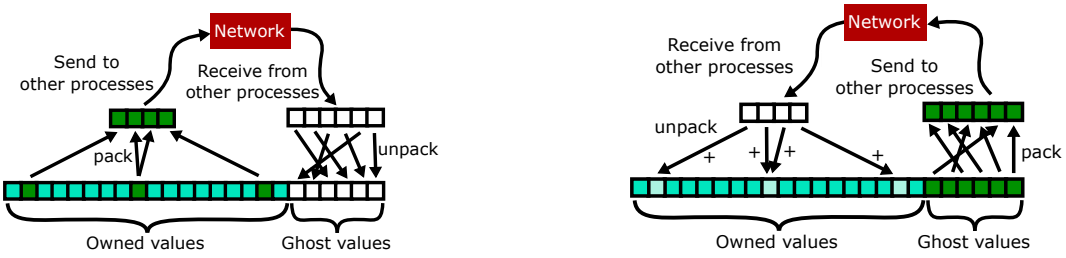
2.3. Operator application

The most substantial computation in the p-multigrid solver is computation of the operator action, $y = Ax$. We compute the action using the sum-factorisation technique on hexahedral cells and using full Gauss quadrature integration ($p + 2$ quadrature points in each direction). Implementation and analysis of the action kernel is technical, and we report this in detail elsewhere [4]. The performance of the operator is limited by shared memory bandwidth, and the Q6 kernel achieves approximately 45% of DRAM bandwidth.

3. Implementation

The p-multigrid solver is built on the DOLFINx finite element library [1]. DOLFINx supports distributed memory unstructured grids, parallel data structures and DoF map construction. The coarse level problem uses Q1 finite elements and is solved using algebraic multigrid. We use the BoomerAMG library from HYPRE [6] for this. GPU kernels are implemented in HIP.

When running on multiple GPUs, we use MPI to communicate across devices. The mesh is partitioned across the GPUs with approximately equal numbers of cells on each process. The mesh partition is computed by ParMETIS [7].



(a) Forward scatter

(b) Reverse scatter

Fig. 4: When a vector is shared across processes, some values are 'ghosted' on other processes. (a) The forward scatter takes local values which are required elsewhere, and packs them into an appropriate send buffer for network communication. When data is received from other processes, it is unpacked into the ghost region. (b) The reverse scatter takes values from the ghost region, and sends them to the owning process. In this case, it is possible to receive multiple values from different processes, and they must be added together in the receiving vector.

With a continuous finite element function space, DoFs on a partition boundary are shared between processes. For each shared DoF we assign an 'owning' MPI rank. On other ranks that require the DoF, it is a 'ghost' DoF. Collecting DoFs into a vector x, computation of the action of the discrete finite element operator, Ax, requires that shared entries in x are available on all processes that share the entry. We support this by packing ghost DoFs at the end of the x array on each process, as illustrated in fig. 4(a). Updating ghost values with the value from the owner is a *forward* scatter. When computing $y = Ax$, each process will compute contributions to the shared DoFs in y. This normally requires a *reverse* scatter to accumulate the contributions on the owning MPI rank, and is illustrated in fig. 4(b).

The need for the reverse scatter can be removed by adding a mesh ghost (halo) cell region (which will also increase the number of ghost entries in x). In this case, when computing $y = Ax$ (matrix-free), each process has enough data to compute its owned entries in y without communication.

Overlapping the communication and computation is straightforward for matrix-free evaluation of operators. Communication of values in x from owners to ghosting ranks can be started, followed by the evaluation of the action on the local (owned) part of x. Once computation using the part of x is complete and ghost values in x have been received, the computation of the action requiring ghost values is executed.

4. Results

To solve the Poisson problem at large scale, we used a multigrid scheme with a hierarchy of levels: Q6–Q3–Q1 on each element (fig. 2), followed by an AMG coarse grid solver based on the Q1 DoFs. All examples use 64-bit floats. We use one MPI rank per GCD. All computations are performed on the GPU partition of the LUMI supercomputer (LUMI-G). LUMI-G has four AMD MI250x GPUs per node, 512 GB of RAM per node and a HPE Cray Slingshot-11 interconnect.

4.1. Smoother and coarse grid solver performance

Figure 5 shows the computed residual against time for different smoother and coarse grid (AMG) solver settings. Each dot in fig. 5 is for one V-cycle. The results indicate that two applications of the Chebyshev smoother perform considerably better than one application, and that 5–10 cycles of the coarse grid AMG solver is appropriate.

4.2. Operator evaluation

Before considering the full multigrid solver, we examine briefly the performance of the operator evaluation, which is the most costly step in algorithm. We assess the operator evaluation using the CEED BP3 problem [3], which involves computing the action of the Laplacian operator within a conjugate gradient loop using $p+2$ Gauss–Legendre quadrature points in each direction, i.e. for Q6, we have $8 \times 8 \times 8$ quadrature points.

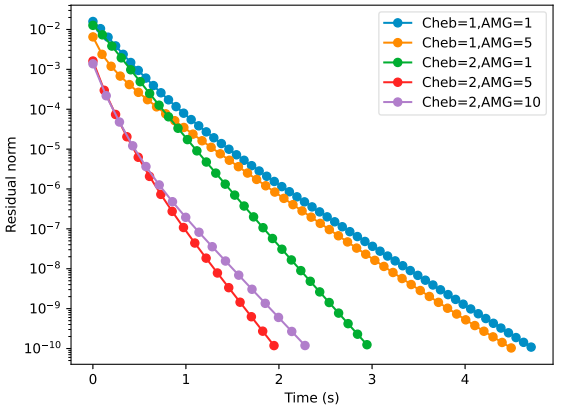


Fig. 5: Convergence of the multigrid scheme on a 160M DoF problem on a cube, using 8 GCDs, showing the effect of different choices of the number of coarse level AMG cycles and the number of Chebyshev smoothing iterations (algorithm 3). The residual norm for fine level problem is shown against elapsed time, and each multigrid cycle is highlighted with a dot. At least two Chebyshev passes are needed in the smoother to obtain fast convergence; it can be seen that 5 AMG iterations lead to fewer iterations, and faster convergence, compared to just using 1, but using 10 AMG iterations is slower overall.

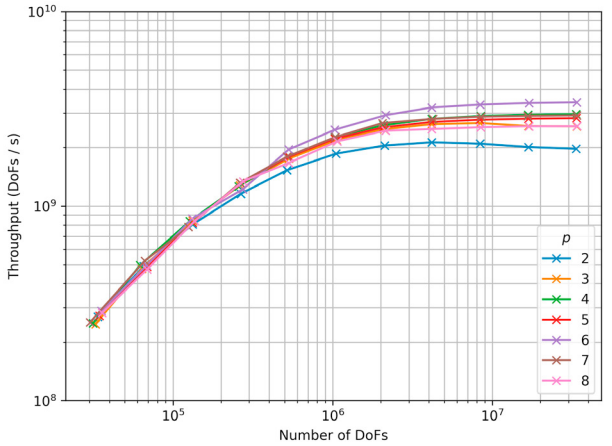


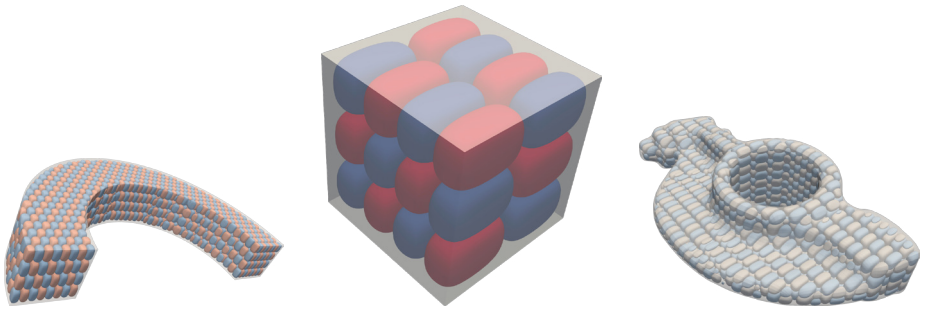
Fig. 6: Throughput for the CEED BP3 problem on MI250x on one GCD using a cube mesh: the maximum efficiency is reached above 10^6 DoFs.

Figure 6 shows the throughput in DoFs per second for different polynomial degrees (Q2–Q8) on a single MI250x GCD as a function of the problem size. Note that a single MI250x GPU device has two GCDs. For problems below approximately 10^6 DoFs, the throughput drops due to kernel launch latency. This sets a lower bound on the amount of work per device required for efficient computation. Note from fig. 6 that the Q6 element yields the highest throughout, peaking at 3.4 GDoF/s. The throughout at lower degrees is lower, but this has minimal effect on overall runtime because the lower degree grids are considerably smaller.

4.3. Full Poisson solver

We examine performance by solving a homogeneous Poisson problem eq. (1) with the right-hand side:

$$f = \sin(k_x x)\sin(k_y y)\sin(k_z z) \tag{3}$$



(a) B16 (b) cube (c) rocker

Fig. 7: Solutions to eq. (1) on hexahedral meshes: Meshes (a) and (c) are taken from Evocube [5]; contour surfaces of the solution are shown at values of ±0.1.

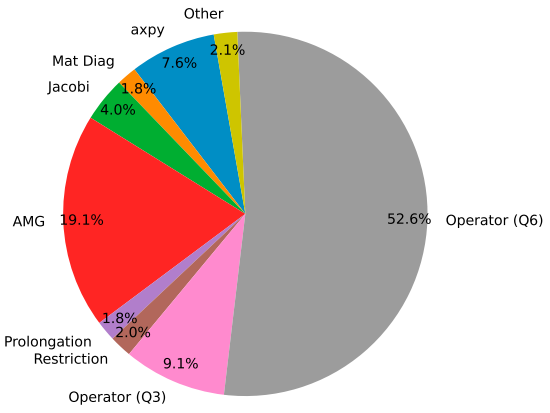


Fig. 8: Proportion of time spent in different parts of the p-multigrid iterations, run on a single node (8 GCDs) with 20M DoFs per GCD: 160M DoFs in total. Two Chebyshev passes are used per smoothing operation, and five AMG cycles on the coarse level. The timings are averaged over 60 V-cycles.

with the wavenumbers $k_x = 2$, $k_y = 3$ and $k_z = 4$ in the x, y and z directions. For comparison purposes, we used a fixed number of multigrid cycles rather than terminating once a convergence criterion is met.

Weak scaling performance is shown in fig. 9 on unit cube meshes of increasing size, providing a weak scaling from one GCD to a maximum of 1024 – an example solution is shown in Figure 7(b). A consideration for weak scaling for this problem is that at low numbers of MPI ranks the domain portion on each rank will likely include the external boundary, and will therefore involve relatively less parallel communication than for larger problems on a greater number of ranks. This effect diminishes for large problems.

To investigate strong scaling, we refined the rocker mesh (fig. 7(c)) to create problems with (i) 105 million and (ii) 840 million DoFs. The strong scaling parallel efficiency for each case is shown in fig. 10. The efficiency is reasonable, but highlights a typical problem that can limit the strong scaling of GPU solvers: kernel launch latency. The latency issue is evident in fig. 6, which shows that there is a lower bound on the amount of 'work' per GPU for hiding kernel launch latency.

Figure 8 shows the fraction of time spent in each kernel on each V-cycle. It is clear that evaluation of the operator action on the fine grid is the dominant cost, followed by the AMG solver on the coarse grid.

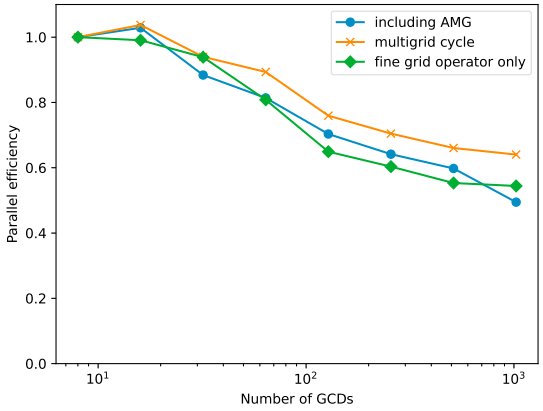
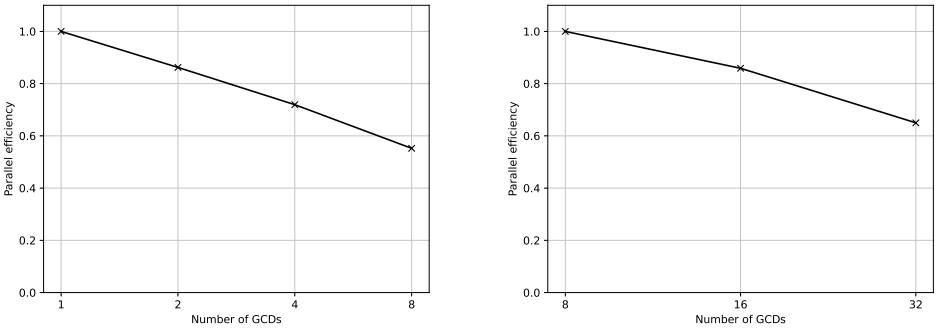


Fig. 9: Weak scaling on a unit cube (10M DoFs/GCD) from 1 to 1024 GCDs. Parallel efficiency is shown for 40 iterations of P-multigrid; timings are shown for the fine-grid operator alone, then with the multigrid cycle added, and finally also with algebraic multigrid (BoomerAMG) on the coarse level.



(a) 105M DoFs.

(b) 840M DoFs.

Fig. 10: Strong scaling parallel efficiency on the rocker mesh with (a) 105M DoFs and (b) 840M DoFs. Case (b) is too large for a single GCD; the smallest number of GCDs that can be used for this case is 8.

5. Conclusions

We have described a *p*-multigrid method and implementation for solving finite element problems efficiently on multi-GPU systems. There are numerous ingredients (kernels) in the solver, including kernels for geometry data pre-computation, operator evaluation, interpolation and restriction between levels, diagonal preconditioning, interfacing to existing algebraic multigrid packages, and distributed memory ghost updates. The performance of the method and implementation has been tested on the LUMI supercomputer and demonstrates good parallel performance. Further work is needed to generalise to other equation operators, and to extend to other cell shapes. Tetrahedral cells give greater geometric flexibility but efficient algorithms at higher orders are more complex as tetrahedral cells do not have a natural tensor-product structure that can be exploited by sum factorisation.

Acknowledgements

We acknowledge EuroHPC benchmarking and development access (EHPC-BEN-2022B11-044, EHPC-DEV-2023D06-031 and EHPC-DEV-2024D05-052). CNR, IAB, AJ and GNW acknowledge the support of EPSRC through EP/S005072/1, and CNR, JPD and GNW acknowledge the support of UKRI/EPSRC through EP/W026635/1.

References

[1] Baratta, I.A., Dean, J.P., Dokken, J.S., Habera, M., Hale, J., Richardson, C.N., Rognes, M.E., Scroggs, M.W., Sime, N., Wells, G.N., 2023. DOLFINx: the next generation FEniCS problem solving environment URL: https://doi.org/10.5281/zenodo.10447666.

[2] Brown, J., Barra, V., Beams, N., Ghaffari, L., Knepley, M., Moses, W., Shakeri, R., Stengel, K., Thompson, J.L., Zhang, J., 2022. Performance portable solid mechanics via matrix-free *p*-multigrid. arXiv preprint arXiv:2204.01722 .

[3] CEED Team, 2025. CEED bake-off problems. https://ceed.exascaleproject.org/bps/. Accessed: April 8, 2025.

[4] Dean, J.P., 2025. Algorithms and performance analysis for finite element operators on GPUs.

[5] Dumery, C., Protais, F., Mestrallet, S., Bourcier, C., Ledoux, F., 2022. Evocube: A genetic labelling framework for polycube-maps, in: Computer Graphics Forum, Wiley Online Library. pp. 467–479.

[6] Falgout, R.D., Yang, U.M., 2002. hypre: A library of high performance preconditioners, in: International Conference on computational science, Springer. pp. 632–641.

[7] Karypis, G., Schloegel, K., Kumar, V., 1997. Parmetis: Parallel graph partitioning and sparse matrix ordering library .

[8] Kronbichler, M., Ljungkvist, K., 2019. Multigrid for matrix-free high-order finite element computations on graphics processors. ACM Transactions on Parallel Computing (TOPC) 6, 1–32.

[9] Naumov, M., Arsaev, M., Castonguay, P., Cohen, J., Demouth, J., Eaton, J., Layton, S., Markovskiy, N., Reguly, I., Sakharnykh, N., et al., 2015. AmgX: A library for GPU accelerated algebraic multigrid and preconditioned iterative methods. SIAM Journal on Scientific Computing 37, S602–S626.

[10] Phillips, M., Fischer, P., 2022. Optimal Chebyshev smoothers and one-sided V-cycles. arXiv preprint arXiv:2210.03179 .

[11] Rønquist, E.M., Patera, A.T., 1987. Spectral element multigrid. I. formulation and numerical results. Journal of Scientific Computing 2, 389–406.

[12] Thompson, J.L., Brown, J., He, Y., 2023. Local Fourier analysis of *p*-multigrid for high-order finite element operators. SIAM Journal on Scientific Computing 45, S351–S370.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 92–101

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Large-scale simulations of lattice QCD for nucleon structure using N_f=2+1+1 flavors of twisted mass fermions

C. Alexandrou[a,b], S. Bacchio[b], L. Chacon[b], J. Finkenrath[c], M. Garofalo[d], C. Iona[a], B. Kostrzewa[d], G. Koutsou[b,*], Y. Li[a], F. Pittler[b], B. Prasad[b], A. Sen[d], C. Urbach[d]

[a]*Department of Physics, University of Cyprus, 20536 Nicosia, Cyprus*
[b]*Computation-based Science and Technology Research Center, The Cyprus Institute, 2121 Nicosia, Cyprus*
[c]*Theoretical Physics Department, CERN 1211 Geneva 23, Switzerland*
[d]*Helmholtz-Institut für Strahlen und Kernphysik (Theory), Rheinische Friedrich-Wilhelms-Universität Bonn, Nussallee 14-16, 53115 Bonn, Germany*

Abstract

Understanding the internal structure of protons and neutrons is a fundamental challenge in nuclear physics that requires both theoretical and computational advances. In this work, we present results from large-scale lattice Quantum Chromodynamics (QCD) simulations performed on European supercomputing facilities to calculate key nucleon structure quantities. Our calculations use gauge field configurations with N_f=2+1+1 twisted mass Wilson-clover fermions at physical quark masses and multiple lattice spacings, allowing for controlled continuum extrapolations. The highly optimized tmLQCD software, combined with the QUDA library for GPU acceleration, enables efficient execution on current heterogeneous computing architectures. We provide details of this computational approach and present our latest results on nucleon structure observables, including nucleon charges, electromagnetic form factors, and momentum fraction. These results include the first continuum extrapolation of nucleon structure quantities using only simulations at physical quark masses, eliminating systematic uncertainties associated with chiral extrapolations.

Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day

Keywords: lattice QCD; nucleon structure; Hybrid Monte Carlo

1. Introduction

The strong interactions, described by Quantum Chromodynamics (QCD), bind quarks and gluons to form protons and neutrons, the building blocks of atomic nuclei. Understanding the internal structure of these particles is crucial for advancing nuclear and particle physics, however deriving their properties directly from QCD remains challenging due to its non-perturbative nature at low energies. Lattice QCD provides a first-principles computational framework to tackle this challenge by discretizing space-time on a four-dimensional lattice and numerically evaluating the path

* Corresponding author. Tel.: +357-22-208-633
E-mail address: g.koutsou@cyi.ac.cy

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day
10.1016/j.procs.2025.08.236

Ensemble	L/a	T/a	$\sim a$ [fm]	$\sim M_\pi$ [MeV]	β	$c_{\rm sw}$	κ	$a\mu_\ell$	$a\mu_\sigma$	$a\mu_\delta$	$N_{\rm traj}$	τ
cA211.12.48	48	96	0.091	174	1.726	1.7400	0.1400650	0.00120	0.14080	0.15210	2721	1.0
cA211.15.64	64	128	–	194	–	–	0.1400640	0.00150	–	–	3415	–
cA211.15.48	48	96	–	–	–	–	0.1400640	–	–	–	6147	–
cA211.30.32	30	64	–	272	–	–	0.1400645	0.00300	–	–	10234	–
cA211.40.24	24	48	–	315	–	–	0.1400645	0.00400	–	–	4882	–
cA211.53.24	24	48	–	360	–	–	0.1400645	0.00530	–	–	4489	–
cAp211.085.56	56	112	0.087	145	1.745	1.7112	0.1400083	0.00085	0.13839	0.14656	5494	2.0
cAp211.085.48	48	96	–	–	–	–	–	–	–	–	15003	1.0
cB211.072.96	96	192	0.080	140	1.778	1.6900	0.1394267	0.00072	0.1246864	0.1315052	3428	1.0
cB211.072.64	64	128	–	–	–	–	–	–	–	–	3165	–
cB211.072.48	48	96	–	–	–	–	–	–	–	–	3534	–
cB211.14.64	64	128	–	194	–	–	–	0.00140	–	–	4397	1.5
cB211.14.48	48	96	–	–	–	–	–	0.00140	–	–	2897	–
cB211.25.48	48	96	–	260	–	–	–	0.00250	–	–	5349	1.0
cB211.25.32	32	64	–	–	–	–	–	–	–	–	3959	–
cB211.25.24	24	48	–	–	–	–	–	–	–	–	4585	–
cC211.06.112	112	224	0.068	137	1.836	1.6452	0.13875285	0.00060	0.106586	0.107146	1303	1.0
cC211.06.80	80	160	–	–	–	–		–	–	–	3147	–
cC211.20.48	48	96	–	250	–	–		0.00200	–	–	2727	–
cD211.054.128	128	256	0.057	141	1.900	1.6112	0.137972174	0.00054	0.087911	0.086224	838	–
cD211.054.96	96	192	–	–	–	–		–	–	–	3386	–
cE211.044.112	112	224	0.049	136	1.960	1.5792	0.137412880	0.00044	0.077707	0.074647	4079	–

Table 1. Algorithmic and bare action parameters of the ETMC N_f=2+1+1 Wilson clover twisted mass ensembles. Lattice spacing (a) and pion mass M_π values are approximate. The trajectory length in molecular dynamics units is given by τ. The trajectory counts $N_{\rm traj}$ correspond to the state at the time of writing and might have increased since. Dashes (–) indicate that the value from the row above is implied.

integral of QCD through Monte Carlo methods [1]. Lattice QCD calculations involve two computationally intensive stages; first, the generation of gauge field configurations via Markov chain Monte Carlo sampling of the QCD path integral, and second the analysis of these configurations to extract physical observables. Both stages require significant high performance computing resources and specialized algorithms. In this proceeding, we present results from large-scale computational efforts of our Extended Twisted Mass collaboration (ETMC) in both the simulation component and the extraction of nucleon structure quantities using European supercomputing facilities.

For gauge field generation, our simulations use the tmLQCD software package [2, 3, 4] with significant portions of the computation offloaded to GPUs through the QUDA library [5, 6], enabling efficient use of heterogeneous supercomputing resources. For the analysis phase, recent algorithmic advances, including highly optimized multigrid solvers for the inversion of the Dirac operator [7, 8, 9] and effective noise reduction techniques for disconnected quark diagrams [10, 11], have made it possible to perform these calculations directly at physical quark masses, eliminating the need for extrapolations from heavier-than-physical quark masses that introduce uncontrolled systematic errors.

In the remainder of this proceeding, we outline our simulation program and present results for key nucleon structure observables using N_f=2+1+1 twisted mass fermions using exclusively physical quark mass simulations (physical point) with different lattice spacings, providing a first continuum extrapolation of these quantities [12, 13].

2. Overview of Current Ensembles

An overview of the current ETMC ensembles, using the N_f=2+1+1 twisted-mass Wilson clover formulation [14, 15], is listed in Table 1, including the bare parameters of each ensemble. In Figure 1, we plot the available lattice spacings (a) and lattice volumes in terms of $M_\pi L$, where L is the spatial extent and M_π the pion mass. Gradient flow observables are shown in the left panel of Figure 2, whereas the right panel illustrates the evolution of the topological charge, demonstrating that no noticeable freezing is present even at the smallest lattice spacing of $a\simeq$0.049 fm.

3. Computational Setup

A main component of our computations is the repeated inversion of the twisted-mass Dirac matrix, for which we employ a highly optimized iterative multigrid solver [6, 7], allowing us to efficiently invert the matrix for multiple

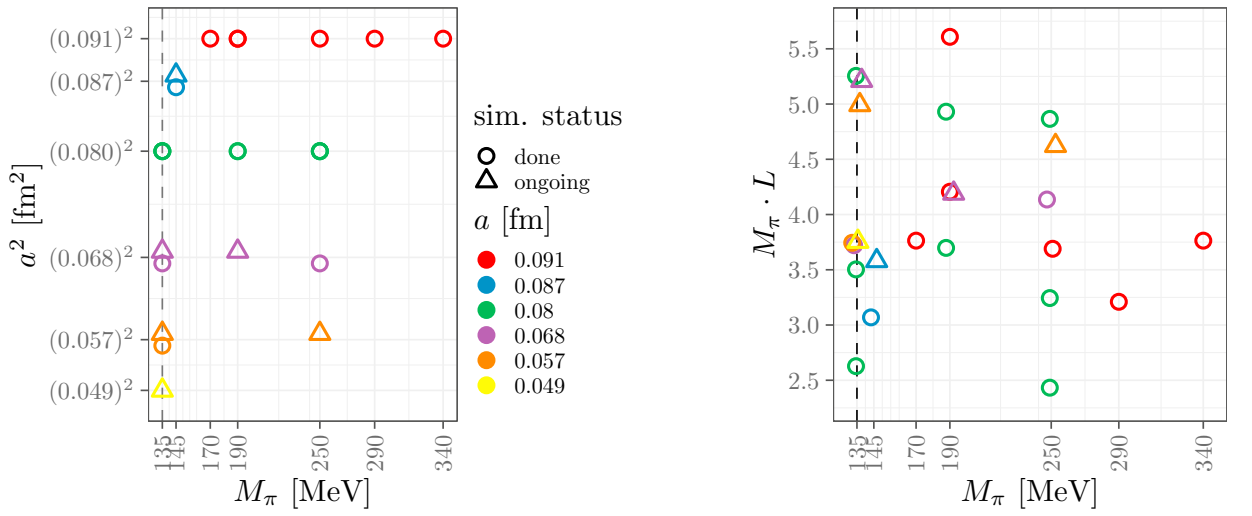


Fig. 1. Overview of all N_f=2+1+1 ETMC Wilson clover twisted mass ensembles as a function of the pion mass, the lattice spacing and $M_\pi L$. Points are slightly displaced either horizontally or vertically to improve visibility. Lattice spacing and pion mass values are approximate.

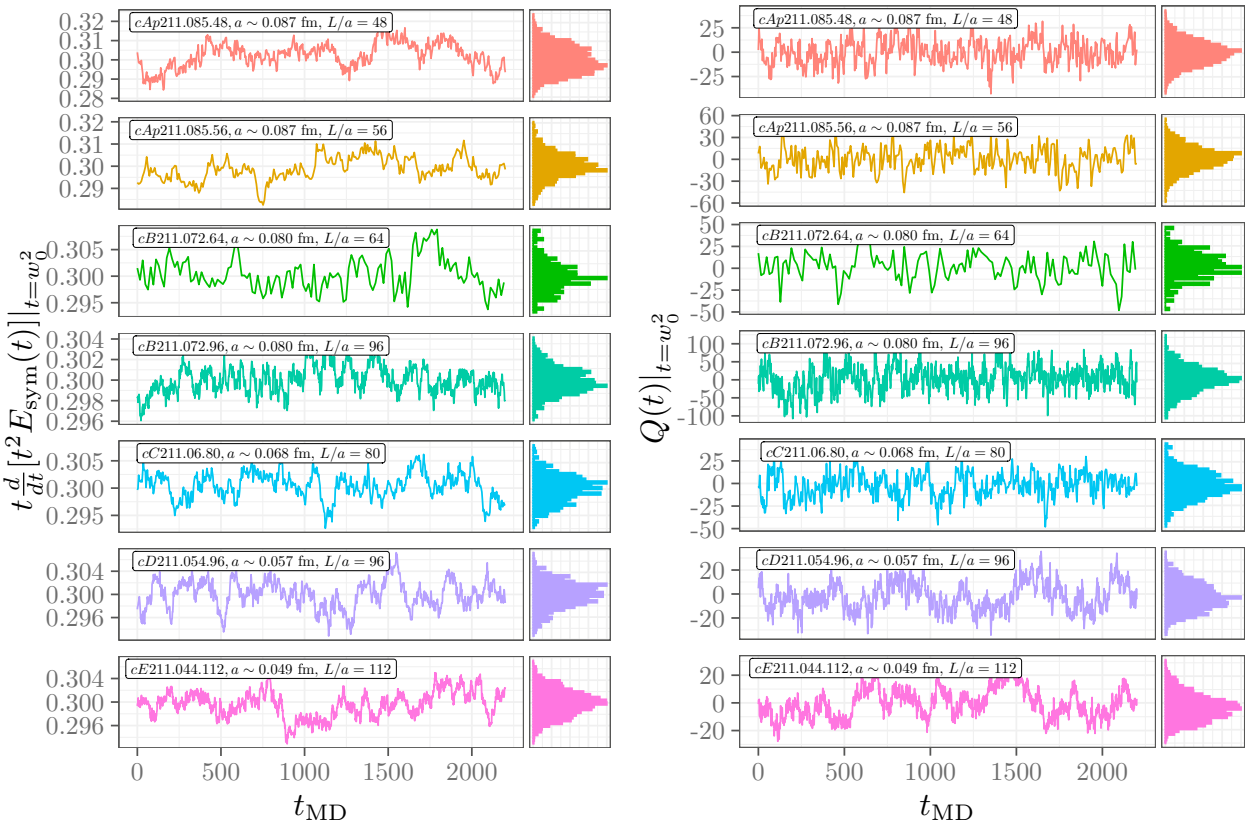


Fig. 2. Left: Molecular dynamics histories for different ensembles close to or at the physical point of the flow-time derivative of the gradient-flowed energy density, interpolated to the flow time where the ensemble average $t\frac{t}{dt}\{t^2\langle E(t)\rangle\} = W(t)|_{t=w_0^2} = 0.3$. Only the first 2,200 unit-length molecular dynamics units are shown in the histories whereas the histograms in the right sub-panels represent the number of trajectories listed in Table 1. Right: Similar histories for the gradient-flowed topological charge in the clover definition evaluated at a flow time $t = w_0^2$.

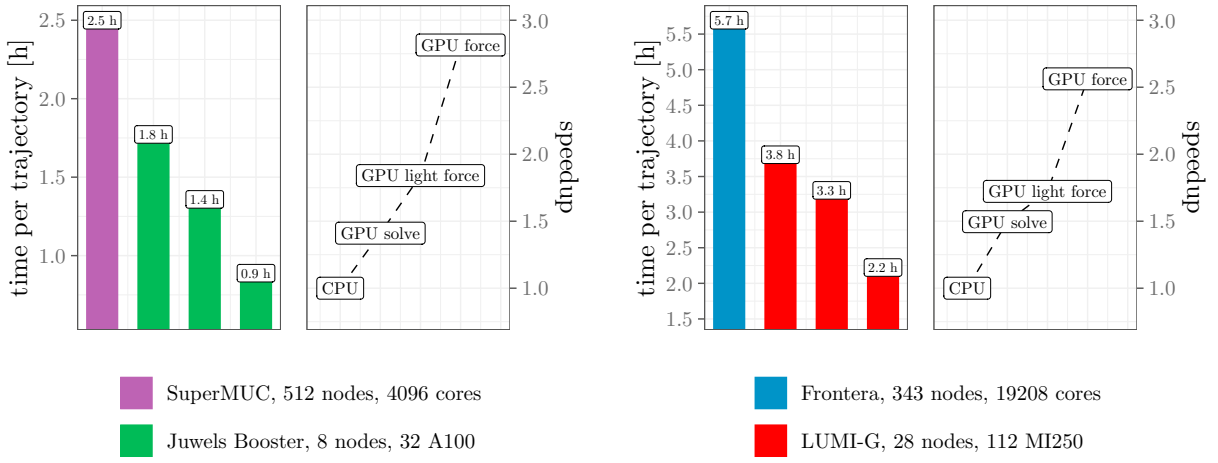


Fig. 3. Left: Time per unit length trajectory of tmLQCD+QUDA (green) for a $64^3 \times 128$ ensemble at the physical point compared to the CPU machine it was originally generated on (purple) using tmLQCD+DD-αAMG [7]. The different speedups correspond to increasing levels of QUDA offloading. Right: The same comparison, but between tmLQCD+QUDA and tmLQCD+DD-αAMG+QPhiX [16] running a $112^3 \times 224$ ensemble at the physical point on LUMI-G and Frontera, respectively.

right-hand sides at a reduced cost. Multigrid proceeds by solving the system on coarser grids, using the solution as a preconditioner for the full system and yielding orders of magnitude improvement compared to standard solvers, such as Conjugate Gradient (CG). Within our group, we have developed a multigrid specifically for twisted mass fermions, which allows tuning the twisted mass parameter at each level of the multigrid algorithm [6, 7]. The solver, as implemented in the QUDA community library optimized for multi-GPU systems [5, 17], results in a highly efficient implementation that has allowed us to efficiently carry our these calculations on several flagship European supercomputers. As a first step in interfacing our main code, tmLQCD, to QUDA [4] we offloaded the inversion of the various Dirac operators as well as the gauge force computation. To ensure optimal performance on different GPU architectures we utilize an auto-tuner for the solver parameters [18]. Recently, we further offloaded all computations of the fermionic force. In this way, tmLQCD can reach GPU utilizations over 70% and even up to 90% depending on the number of available CPU cores per GPU. The evolution of the performance of this tmLQCD+QUDA setup is shown in the left panel of Figure 3.

4. Nucleon Structure

Understanding nucleon structure is essential for gaining insights into the fundamental properties of QCD. In this section, we briefly introduce our methodology and present representative results of the continuum limit for key nucleon structure observables, using three ensembles at physical values of the pion mass, namely the ensembles `cB211.072.64`, `cC211.060.80`, and `cD211.054.96` from Table 1. We focus on the isovector axial charge, proton and neutron electromagnetic form factors, and the momentum fraction. i) The nucleon axial charge governs the rate of neutron beta decay and is crucial for tests of the Standard Model's CKM matrix unitarity [12, 19]. Its precise lattice determination serves as a key benchmark due to its well-established experimental value [20]. ii) The electromagnetic form factors of the proton and neutron, which parameterize the distribution of charge and magnetization within these particles, are essential for interpreting electron scattering experiments [21] and resolving puzzles such as the proton radius discrepancy between muonic hydrogen spectroscopy and electron scattering measurements [22, 23]. iii) Quark momentum fractions determine how the nucleon's momentum is distributed among its constituents, providing crucial constraints for parton distribution functions used to interpret high-energy collider experiments [24, 25].

4.1. Methodology

Nucleon structure quantities are extracted from matrix elements of local operators between nucleon states. On the lattice, these matrix elements are extracted from correlation functions computed using appropriately defined interpolating operators with the quantum numbers of the nucleon. The standard nucleon interpolating field used in our calculations is $\mathcal{J}_N(x) = \epsilon^{abc} u^a(x) \left[u^{bT}(x) C\gamma_5 d^c(x)\right]$, where $u(x)$ and $d(x)$ are up and down quark fields, $C = \gamma_0\gamma_2$ is the charge conjugation matrix, and a, b, c are color indices. For nucleon structure, we compute two- and three-point functions given by

$$C(\Gamma_0, \vec{p}; t_s, t_0) = \sum_{\vec{x}_s} e^{-i(\vec{x}_s - \vec{x}_0)\cdot\vec{p}} \mathrm{Tr}\left[\Gamma_0 \langle \mathcal{J}_N(t_s, \vec{x}_s) \bar{\mathcal{J}}_N(t_0, \vec{x}_0)\rangle\right], \text{ and} \tag{1}$$

$$C^\mu(\Gamma_\nu, \vec{q}, \vec{p}\,'; t_s, t_{\rm ins}, t_0) = \sum_{\vec{x}_{\rm ins}, \vec{x}_s} e^{i(\vec{x}_{\rm ins} - \vec{x}_0)\cdot\vec{q}} e^{-i(\vec{x}_s - \vec{x}_0)\cdot\vec{p}\,'} \mathrm{Tr}\left[\Gamma_\nu \langle \mathcal{J}_N(t_s, \vec{x}_s) \mathcal{O}^\mu(t_{\rm ins}, \vec{x}_{\rm ins}) \bar{\mathcal{J}}_N(t_0, \vec{x}_0)\rangle\right], \tag{2}$$

where t_0 is the source time, t_s is the sink time, $\Gamma_0 = \frac{1}{2}(1 + \gamma_0)$ is the unpolarized projector and $\Gamma_k = i\Gamma_0\gamma_5\gamma_k$ with $k = 1, 2, 3$ are polarized projectors, $\mathcal{O}^\mu$ is the operator of interest inserted at time $t_{\rm ins}$, $\vec{q}$ is the momentum transfer, $\vec{p}'$ is the sink momentum, and the traces are over spinor indices.

Three-point functions receive contributions from both so-called connected and disconnected diagrams. The former correspond to the operator being inserted on one of the valence quarks, while the latter involve operator insertions on sea quarks, resulting in quark loops. The evaluation of disconnected diagrams is computationally very demanding, requiring advanced noise reduction techniques [26]. For the connected contributions, we use standard techniques involving the computation of the sequential propagator through the sink. In this approach, the sink-source time separation, the projector and the momentum at the sink $\vec{p}$,$'$ are kept fixed. For the disconnected contributions, we employ stochastic techniques combined with dilution schemes, hierarchical probing [11], and deflation of low modes [27]. To illustrate the scale of computations required, Table 2 shows the statistics used for calculating the connected three-point functions for our three ensembles. As the sink-source separation t_s increases, we increase the number of source positions to maintain similar statistical precision. Indicatively, for the statistics given in Table 2 we find a relative statistical error between 0.8% to 2% depending on the value of the time (t_s) for the nucleon isovector axial three-point function. The computational resources required are approximately 250,000 GPU-hours for `B64`, 550,000 GPU-hours for `C80`, and 600,000 GPU-hours for `D96`, in units of NVIDIA A100 GPU-hours. This means that the analysis of the connected contributions alone for a single ensemble, requires resources comparable to an annual extreme scale allocation for this statistical accuracy.

Ens. ID (short name)	Vol.	a [fm]	$N_{\rm conf}$	$(t_s/a)_{N_{\rm src}}$	$N^{\rm 2p}_{\rm src}$
`cB211.072.64` (`B64`)	64×128	0.080	750	8_1, 10_2, 12_5, 14_{10}, 16_{32}, 18_{112}, 20_{128}	477
`cC211.060.80` (`C80`)	80×160	0.068	400	6_1, 8_2, 10_4, 12_{10}, 14_{22}, 16_{48}, 18_{45}, 20_{116}, 22_{246}	650
`cD211.054.96` (`D96`)	96×192	0.057	500	8_1, 10_2, 12_4, 14_8, 16_{16}, 18_{32}, 20_{64}, 22_{16}, 24_{32}, 26_{64}	480

Table 2. We show details of the ensembles analyzed, including their short name (first column), lattice volume (second column), lattice spacing in fm (third column), number of configurations analyzed (fourth column), sink-source separations (t_s/a) analyzed for the connected three-point functions with number of source positions per configuration ($N_{\rm src}$) indicated as a subscript (fifth column), and number of source positions per configuration analyzed for the two-point function ($N^{\rm 2p}_{\rm src}$).

For the disconnected contributions, we use the techniques and parameters shown in Table 3. The calculation of quark loops is particularly intensive, requiring specialized algorithms and significant computational resources.

As in the case of the simulations, the most demanding component of our nucleon structure program is the inversion of the twisted-mass Dirac matrix to obtain the quark propagators which are in turn contracted appropriately to produce

	cB211.072.64				cC211.060.80				cD211.054.96			
Flavor	N_{defl}	N_r	N_{Had}	N_{vect}	N_{defl}	N_r	N_{Had}	N_{vect}	N_{defl}	N_r	N_{Had}	N_{vect}
Light	200	1	512	6144	450	1	512	6144	0	8	512	49152
Strange	0	2	512	12288	0	4	512	24576	0	4	512	24576
Charm	0	12	32	4608	0	1	512	6144	0	1	512	6144

Table 3. Parameters and statistics used for the evaluation of the disconnected three-point functions, for the same N_{conf} as Table 2. For each ensemble, in the columns from left to right we give: i) the number of deflated eigenvectors N_{def}, ii) the number of stochastic sources N_r, iii) the number of Hadamard vectors N_{Had}, and iv) the total number of computed vectors, N_{vect}, which after color and spin dilution are obtained via $12 \times N_r \times N_{\text{Had}}$.

the nucleon two- and three-point functions. For each configuration and source position listed in Tables 2 and 3, we require $O(10)$ inversions of the Dirac matrix, meaning for any given ensemble we require $O(10^6)$ inversions. The multigrid solver via QUDA is crucial in this step, as it allows us to efficiently solve for the required number right hand sides within a typical annual allocation on a range of supercomputing systems, as shown in the left panel of Figure 4.

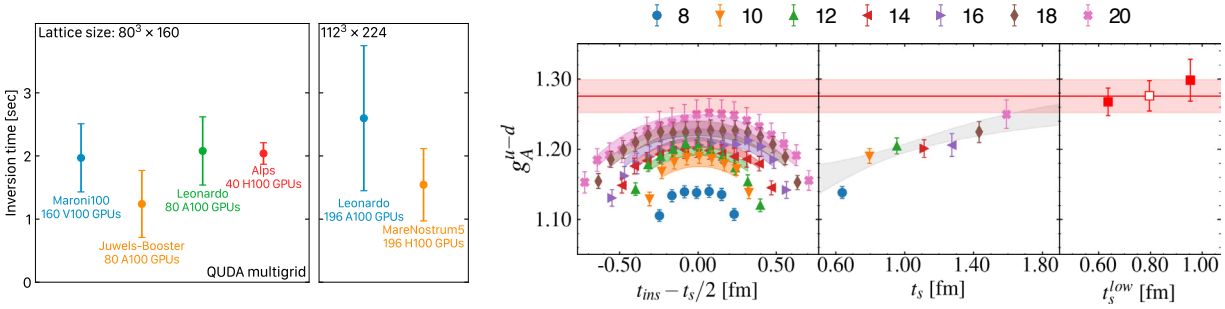


Fig. 4. Left: Time per inversion of the QUDA multigrid solver when using the C80 and E112 ensembles on European supercomputers. The bars indicate the standard deviation over a number of inversions spanning between hundreds of thousands to millions, depending on the system. Right: Excited state analysis for determining the isovector axial charge g_A^{u-d} on the cB211.072.64 ensemble. Left panel: Ratio data for different source-sink separations t_s (different symbols) as a function of the insertion time shifted by $t_s/2$. The bands show two-state fits. Middle panel: Ratio values for $t_{\text{ins}} = t_s/2$, with the grey band showing the predicted time-dependence from the two-state fit. Right panel: Extracted values of g_A^{u-d} using different two-state fit ranges. The open symbol indicates the selected value, with the red band showing its statistical error.

The two- and three-point functions contain contributions from the ground state as well as excited states. Their spectral decomposition takes the form, $C(\vec{p}, t_s) = \sum_i c_i(\vec{p}) e^{-E_i(\vec{p})t_s} + \cdots$ and $C^\mu(\Gamma_\nu, \vec{p}\,', \vec{p}, t_s, t_{\text{ins}}) = \sum_{i,j} \mathcal{A}_\mu^{i,j}(\Gamma_\nu, \vec{p}\,', \vec{p}) \times e^{-E_i(\vec{p}\,')(t_s - t_{\text{ins}}) - E_j(\vec{p})t_{\text{ins}}} + \cdots$, respectively, where $E_i(\vec{p})$ is the energy of the i-th state with momentum $\vec{p}$, and $\mathcal{A}_\mu^{i,j}$ are the transition matrix elements. To isolate the ground state contribution, we form the ratio:

$$R^\mu(\Gamma_\nu, \vec{p}\,', \vec{p}; t_s, t_{\text{ins}}) = \frac{C^\mu(\Gamma_\nu, \vec{p}\,', \vec{p}; t_s, t_{\text{ins}})}{C(\Gamma_0, \vec{p}\,'; t_s)} \sqrt{\frac{C(\Gamma_0, \vec{p}; t_s - t_{\text{ins}}) C(\Gamma_0, \vec{p}\,'; t_{\text{ins}}) C(\Gamma_0, \vec{p}\,'; t_s)}{C(\Gamma_0, \vec{p}\,'; t_s - t_{\text{ins}}) C(\Gamma_0, \vec{p}; t_{\text{ins}}) C(\Gamma_0, \vec{p}; t_s)}}. \tag{3}$$

In the limit of large Euclidean time separations, this ratio yields the desired ground state matrix element, $\Pi^\mu(\Gamma_\nu; \vec{p}\,', \vec{p})$, i.e. $R^\mu(\Gamma_\nu, \vec{p}\,', \vec{p}; t_s, t_{\text{ins}}) \xrightarrow[t_{\text{ins}} \gg a]{t_s - t_{\text{ins}} \gg a} \Pi^\mu(\Gamma_\nu; \vec{p}\,', \vec{p})$. In practice, we cannot take the time separations arbitrarily large due to the exponential increase of the relative statistical errors. Therefore, to ensure ground state dominance, we explicitly include contributions from excited states, fitting the two- and three-point functions to forms including terms for both ground and excited states [12]. In many cases, we find the excited state energies extracted from fits to two-point functions cannot fully describe the excited-states in three-point functions, as also predicted from chiral perturbation theory [28]. Therefore, we allow different excited state energies in the two- and three-point functions, which is critical for reliably extracting ground state matrix elements. The right panel of Figure 4 shows an example of our excited state analysis for the isovector axial charge g_A^{u-d} on the cB211.072.64 ensemble. The right panel, which shows

the extracted g_A^{u-d} values from the two-state fit as we increase the smallest sink-source separation, demonstrates the convergence of our analysis. For our final result, we use a model averaging procedure based on the Akaike Information Criterion (AIC) [29] to combine results from different fit ranges and ansätze, which allows us to reliably estimate both statistical and systematic uncertainties.

4.2. Isovector nucleon matrix elements in the continuum limit

As mentioned, in the results that follow we take the continuum limit using three ensembles with different lattice spacings and approximately the same physical volume of about (5 fm)^3 [30, 12]. The nucleon axial charge g_A^{u-d}

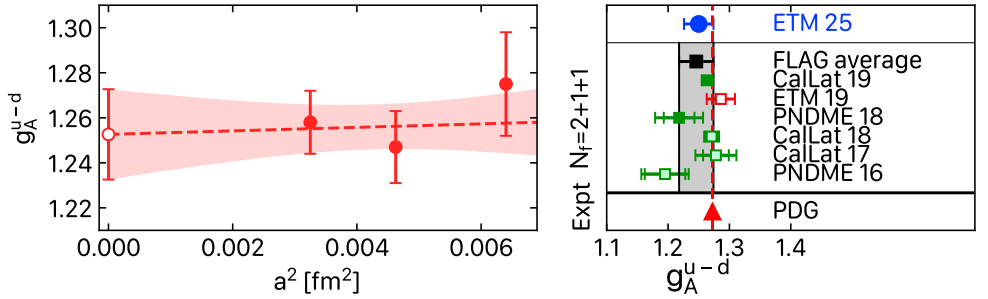


Fig. 5. Left: Continuum extrapolation of the isovector axial charge g_A^{u-d}. The filled symbols represent results from individual ensembles, while the open symbol shows the extrapolated value at the continuum limit. Right: Comparison of our result for g_A^{u-d} with other lattice QCD results and with their average as done in the recent FLAG review [31].

is defined as the nucleon matrix element of the isovector axial current at zero momentum transfer, $\langle N|\bar{u}\gamma_\mu\gamma_5 u - \bar{d}\gamma_\mu\gamma_5 d|N\rangle = g_A^{u-d}\bar{u}_N\gamma_\mu\gamma_5 u_N$, where u_N is a nucleon spinor. This quantity determines the rate of neutron beta decay and is measured to high precision experimentally, making it an important benchmark for lattice QCD calculations. In the left panel of Figure 5 we show our continuum extrapolation of the isovector axial charge. Our final result in the continuum limit is $g_A^{u-d} = 1.250(24)$ [13], which is in excellent agreement with the experimental value of $g_A^{u-d} = 1.2754(13)$ [20]. We also compare our result to lattice QCD results as aggregated by the Flavor Lattice Averaging Group (FLAG) in their most recent review [31], published prior to our result.

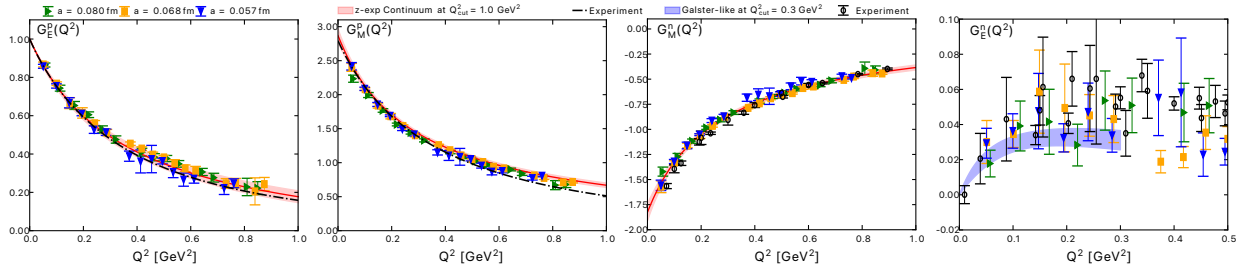


Fig. 6. From left to right, the proton electric, magnetic, and neutron magnetic and electric form factors as functions of Q^2, for the three ensembles (green, yellow, and blue points with decreasing a) and at the continuum limit (red and blue bands). Black dashed lines for the proton and black circles for the neutron show a collection of recent experimental results.

The electromagnetic form factors parameterize the nucleon matrix element of the electromagnetic current, yielding the Dirac and Pauli form factors, $F_1(Q^2)$ and $F_2(Q^2)$, respectively, where Q^2 is the momentum transfer squared. These can be rewritten in terms of the electric and magnetic Sachs form factors, $G_E(Q^2) = F_1(Q^2) + \frac{Q^2}{4m_N^2}F_2(Q^2)$ and $G_M(Q^2) = F_1(Q^2) + F_2(Q^2)$. We calculate both isovector and isoscalar form factors, accounting for disconnected contributions in the latter, and combine them to obtain proton and neutron form factors [21, 32]. Figure 6 shows the Q^2 dependence of the proton and neutron electromagnetic form factors and their continuum limit. We extract the electric and magnetic radii of the proton (p) and neutron (n), i.e. $\langle r_E^2\rangle^{p/n}$ and $\langle r_M^2\rangle^{p/n}$, as well as the magnetic moments (μ_p and

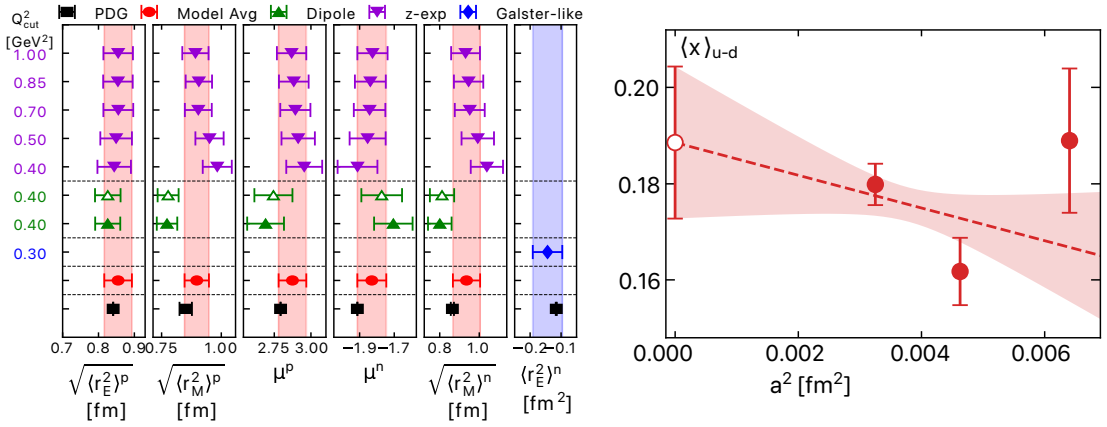


Fig. 7. Left: Electric and magnetic radii and magnetic moments of the proton and neutron for all $Q^2_{\rm cut}$ for the z-expansion (purple downward-pointing triangles) and, in the dipole case, using both one-step (filled green triangle) or two-step (open green triangle) approaches. The blue diamond and blue band corresponds to Galster-like fit to G^n_E. The red point and band denoted "Model average" is obtained by weighting according to the AIC. Right: Continuum extrapolation of the isovector momentum fraction $\langle x\rangle_{u-d}$. The filled symbols represent results from individual ensembles, while the open symbol shows the extrapolated value at the continuum limit.

μ_n), using a range of $Q^2 \to 0$ extrapolations, including varying the maximum Q^2 value used, varying the fit function between the so-called z-expansion and dipole forms, and via either a one-step extrapolation, where the dependence in Q^2 and a^2 are fitted simultaneously, or a two-step extrapolation, where the Q^2-dependence is fitted first for the three ensembles and then the continuum extrapolation is taken in a second step. More details are provided in Ref. [32]. The results are shown in the left panel of Figure 7 and are compared to the experimentally determined values.

The average momentum fraction carried by quarks in the nucleon, $\langle x\rangle^q$, is obtained from the forward matrix element of the traceless part of the energy-momentum tensor, $\langle N(p)|\bar{q}\gamma_{\{\mu}iD_{\nu\}}q|N(p)\rangle = \langle x\rangle^q \bar{u}_N(p)\gamma_{\{\mu}p_{\nu\}}u_N(p)$, where $\{\cdots\}$ denotes symmetrization and subtraction of the trace [33, 30]. Figure 7 shows our continuum extrapolation of the isovector combination of the momentum fraction, $\langle x\rangle^{u-d}$. Our final result in the continuum limit is $\langle x\rangle^{u-d} = 0.171(18)$ [30], which is in good agreement with recent phenomenological determinations from NNPDF [34], JAM [35], and CTEQ-TEA using data from LHC [36].

5. Conclusions

We have presented an overview of our large-scale lattice QCD calculations for both simulations and analysis using European supercomputing facilities. Our gauge field generation program has successfully produced multiple ensembles with N_f=2+1+1 twisted mass Wilson-clover fermions at physical quark masses and with varying lattice spacings, enabled by the highly optimized tmLQCD software package with GPU acceleration through QUDA. Significant computational improvements have resulted in GPU utilization of up to 90%, depending on the target machine.

The analysis of these gauge configurations has yielded precise determinations of key nucleon structure observables, including the axial charge, electromagnetic form factors, and momentum fraction. These results, enabled by large scale computational resources such as those provided by EuroHPC, are the first to include continuum limit extrapolations exclusively using simulations with physical quark masses. This represents a significant advancement, since so far such results included simulations with heavier-than-physical quark masses, thus requiring a modeling of the quark mass dependence of the nucleon structure quantities, which in turn introduced an unknown systematic error.

Current computational efforts are focused on incorporating results with finer lattice spacing, namely the `cE211.044.112` ensemble in our analysis, with $a \simeq 0.049$ fm, which will further improve the robustness of our continuum extrapolations and reduce overall systematic uncertainties. As an indicative example, preliminary results for the isovector tensor charge, $\langle N|\bar{u}\sigma_{\mu\nu}u - \bar{d}\sigma_{\mu\nu}\gamma_5 d|N\rangle = g_T^{u-d}\bar{u}_N\sigma_{\mu\nu}u_N$, available at low statistics are shown in Figure 8. This continued progress in both simulation methodology and analysis techniques promises to deliver increasingly pre-

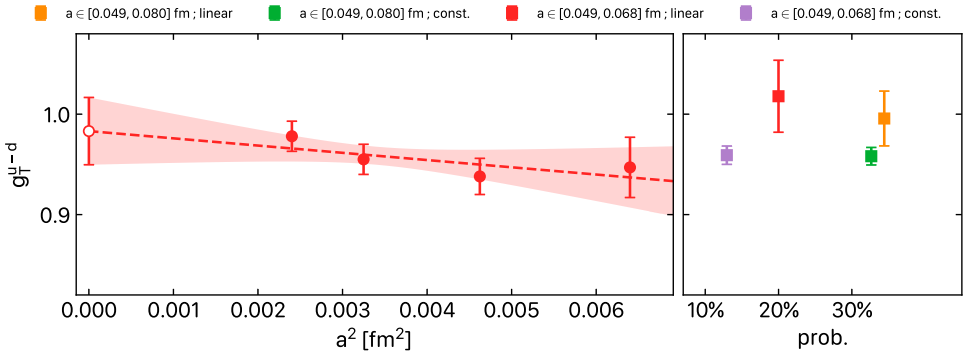


Fig. 8. Continuum limit of the isovector tensor charge g_T^{u-d} with preliminary results for the `cE211.044.112` ensemble included. We vary between linear and constant fits as described in the figure title, and show the value obtained from each fit and its probability in the right panel.

cise determinations of hadron properties directly from QCD, contributing to our fundamental understanding of strong interaction physics.

Acknowledgements

We acknowledge EuroHPC Joint Undertaking for awarding the projects with ID EHPC-EXT-2023E02-052 with access to MareNostrum5, EHPC-REG-2022R02-094 and EHPC-REG-2023R03-187 with access to Meluxina, EHPC-EXT-2022E01-033 access to Leonardo and EHPC-REG-2021R0095 access to LUMI. The authors also gratefully acknowledge access to the Marvin cluster of the University of Bonn and support by the HRZ-HPC team and acknowledge the Texas Advanced Computing Center (TACC) at the University of Texas at Austin for proving HPC resources on Frontera (Project ID PHY21001). C.A. and G.K. acknowledge partial support from the European Joint Doctorate AQTIVATE (Grant Agreement No. 101072344). C. A., S. B., G. K., and F. P. acknowledge support by the projects "IMAGE-N" (EXCELLENCE/0524/0459), "Baryon8" (POSTDOC/0524/0001), "HyperON" (VISION ERC-PATH 2/0524/0001), "MuonHVP" (EXCELLENCE/0524/0017), "PulseQCD" (EXCELLENCE/0524/0269), and "DeNuTra" (EXCELLENCE/0524/0455), co-financed by the European Regional Development Fund and the Republic of Cyprus through the Research and Innovation Foundation. L. C. and B. P. are supported by ENGAGE, which received funding from the EU's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie GA No.101034267. J. F. acknowledges financial support from the Next Generation Triggers project. B. K. and A. S. acknowledge financial support by CRC 1639 NuMeriQS - project no. 511713970.

References

[1] J. Finkenrath, Review on Algorithms for dynamical fermions, PoS LATTICE2022 (2023) 227. arXiv:2402.11704, doi:10.22323/1.430.0227.

[2] K. Jansen, C. Urbach, tmLQCD: A Program suite to simulate Wilson Twisted mass Lattice QCD, Comput. Phys. Commun. 180 (2009) 2717–2738. arXiv:0905.3331, doi:10.1016/j.cpc.2009.05.016.

[3] A. Abdel-Rehim, F. Burger, A. Deuzeman, K. Jansen, B. Kostrzewa, L. Scorzato, C. Urbach, Recent developments in the tmLQCD software suite, PoS LATTICE2013 (2014) 414. arXiv:1311.5495, doi:10.22323/1.187.0414.

[4] B. Kostrzewa, S. Bacchio, J. Finkenrath, M. Garofalo, F. Pittler, S. Romiti, C. Urbach, Twisted mass ensemble generation on GPU machines, PoS LATTICE2022 (2023) 340. arXiv:2212.06635, doi:10.22323/1.430.0340.

[5] M. A. Clark, R. Babich, K. Barros, R. C. Brower, C. Rebbi, Solving Lattice QCD systems of equations using mixed precision solvers on GPUs, Comput. Phys. Commun. 181 (2010) 1517–1528. arXiv:0911.3191, doi:10.1016/j.cpc.2010.05.002.

[6] M. A. Clark, B. Joó, A. Strelchenko, M. Cheng, A. Gambhir, R. Brower, Accelerating Lattice QCD Multigrid on GPUs Using Fine-Grained Parallelization (12 2016). arXiv:1612.07873.

[7] C. Alexandrou, S. Bacchio, J. Finkenrath, A. Frommer, K. Kahl, M. Rottmann, Adaptive Aggregation-based Domain Decomposition Multigrid for Twisted Mass Fermions, Phys. Rev. D 94 (11) (2016) 114509. arXiv:1610.02370, doi:10.1103/PhysRevD.94.114509.

[8] S. Bacchio, C. Alexandrou, J. Finkerath, Multigrid accelerated simulations for Twisted Mass fermions, EPJ Web Conf. 175 (2018) 02002. arXiv:1710.06198, doi:10.1051/epjconf/201817502002.

[9] C. Alexandrou, S. Bacchio, J. Finkenrath, Multigrid approach in shifted linear systems for the non-degenerated twisted mass operator, Comput. Phys. Commun. 236 (2019) 51–64. arXiv:1805.09584, doi:10.1016/j.cpc.2018.10.013.

[10] C. McNeile, C. Michael, Decay width of light quark hybrid meson from the lattice, Phys. Rev. D 73 (2006) 074506. arXiv:hep-lat/0603007, doi:10.1103/PhysRevD.73.074506.

[11] A. Stathopoulos, J. Laeuchli, K. Orginos, Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices (2013). arXiv:1302.4018.

[12] C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, R. Frezzotti, B. Kostrzewa, G. Koutsou, G. Spanoudes, C. Urbach, Nucleon axial and pseudoscalar form factors using twisted-mass fermion ensembles at the physical point, Phys. Rev. D 109 (3) (2024) 034503. arXiv:2309.05774, doi:10.1103/PhysRevD.109.034503.

[13] C. Alexandrou, S. Bacchio, J. Finkenrath, C. Iona, G. Koutsou, Y. Li, G. Spanoudes, Nucleon charges and σ-terms in lattice QCD, Phys. Rev. D 111 (5) (2025) 054505. arXiv:2412.01535, doi:10.1103/PhysRevD.111.054505.

[14] C. Alexandrou, et al., Simulating twisted mass fermions at physical light, strange and charm quark masses, Phys. Rev. D 98 (5) (2018) 054518. arXiv:1807.00495, doi:10.1103/PhysRevD.98.054518.

[15] J. Finkenrath, et al., Twisted mass gauge ensembles at physical values of the light, strange and charm quark masses, PoS LATTICE2021 (2022) 284. arXiv:2201.02551, doi:10.22323/1.396.0284.

[16] M. Schröck, S. Simula, A. Strelchenko, Accelerating Twisted Mass LQCD with QPhiX, PoS LATTICE2015 (2016) 030. arXiv:1510.08879, doi:10.22323/1.251.0030.

[17] R. Babich, M. A. Clark, B. Joo, G. Shi, R. C. Brower, S. Gottlieb, Scaling lattice QCD beyond 100 GPUs, in: International Conference for High Performance Computing, Networking, Storage and Analysis, 2011. arXiv:1109.2935, doi:10.1145/2063384.2063478.

[18] M. Garofalo, B. Kostrzewa, S. Romiti, A. Sen, Autotuning multigrid parameters in the HMC on different architectures, PoS LATTICE2024 (2025) 276. doi:10.22323/1.466.0276.

[19] C. Alexandrou, M. Constantinou, K. Hadjiyiannakou, K. Jansen, C. Kallidonis, G. Koutsou, A. Vaquero Aviles-Casco, Nucleon axial form factors using $N_f = 2$ twisted mass fermions with a physical value of the pion mass, Phys. Rev. D96 (5) (2017) 054507. arXiv:1705.03399, doi:10.1103/PhysRevD.96.054507.

[20] S. Navas, et al., Review of particle physics, Phys. Rev. D 110 (3) (2024) 030001. doi:10.1103/PhysRevD.110.030001.

[21] C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, K. Hadjiyiannakou, K. Jansen, G. Koutsou, A. V. A. Casco, Proton and neutron electromagnetic form factors from lattice QCD (2018). arXiv:1812.10311.

[22] R. Pohl, et al., The size of the proton, Nature 466 (2010) 213–216. doi:10.1038/nature09250.

[23] V. Punjabi, C. F. Perdrisat, M. K. Jones, E. J. Brash, C. E. Carlson, The Structure of the Nucleon: Elastic Electromagnetic Form Factors, Eur. Phys. J. A 51 (2015) 79. arXiv:1503.01452, doi:10.1140/epja/i2015-15079-x.

[24] C. Alexandrou, M. Constantinou, K. Hadjiyiannakou, K. Jansen, F. Manigrasso, Flavor decomposition for the proton helicity parton distribution functions, Phys. Rev. Lett. 126 (10) (2021) 102003. arXiv:2009.13061, doi:10.1103/PhysRevLett.126.102003.

[25] C. Alexandrou, et al., Moments of nucleon generalized parton distributions from lattice QCD simulations at physical pion mass, Phys. Rev. D 101 (3) (2020) 034519. arXiv:1908.10706, doi:10.1103/PhysRevD.101.034519.

[26] C. Alexandrou, M. Constantinou, V. Drach, K. Hadjiyiannakou, K. Jansen, G. Koutsou, A. Strelchenko, A. Vaquero, Evaluation of disconnected quark loops for hadron structure using GPUs, Comput. Phys. Commun. 185 (2014) 1370–1382. arXiv:1309.2256, doi:10.1016/j.cpc.2014.01.009.

[27] A. S. Gambhir, A. Stathopoulos, K. Orginos, Deflation as a Method of Variance Reduction for Estimating the Trace of a Matrix Inverse, SIAM J. Sci. Comput. 39 (2017) A532–A558. arXiv:1603.05988, doi:10.1137/16M1066361.

[28] O. Bär, $N\pi$-state contamination in lattice calculations of the nucleon axial form factors, Phys. Rev. D 99 (5) (2019) 054506. arXiv:1812.09191, doi:10.1103/PhysRevD.99.054506.

[29] W. I. Jay, E. T. Neil, Bayesian model averaging for analysis of lattice field theory results, Phys. Rev. D 103 (2021) 114502. arXiv:2008.01069, doi:10.1103/PhysRevD.103.114502.

[30] C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, K. Hadjiyiannakou, K. Jansen, G. Koutsou, A. Vaquero Aviles-Casco, Nucleon axial, tensor, and scalar charges and σ-terms in lattice QCD, Phys. Rev. D 102 (5) (2020) 054517. arXiv:1909.00485, doi:10.1103/PhysRevD.102.054517.

[31] Y. Aoki, et al., FLAG Review 2024 (11 2024). arXiv:2411.04268.

[32] C. Alexandrou, S. Bacchio, G. Koutsou, B. Prasad, G. Spanoudes, Proton and neutron electromagnetic form factors using Nf=2+1+1 twisted-mass fermions with physical values of the quark masses, PoS LATTICE2024 (2025) 332. arXiv:2502.11301, doi:10.22323/1.466.0332.

[33] C. Alexandrou, M. Constantinou, K. Hadjiyiannakou, K. Jansen, C. Kallidonis, G. Koutsou, A. Vaquero Avilés-Casco, C. Wiese, Nucleon Spin and Momentum Decomposition Using Lattice QCD Simulations, Phys. Rev. Lett. 119 (14) (2017) 142002. arXiv:1706.02973, doi:10.1103/PhysRevLett.119.142002.

[34] R. D. Ball, et al., Parton distributions from high-precision collider data, Eur. Phys. J. C 77 (10) (2017) 663. arXiv:1706.00428, doi:10.1140/epjc/s10052-017-5199-5.

[35] N. Sato, C. Andres, J. J. Ethier, W. Melnitchouk, Strange quark suppression from a simultaneous Monte Carlo analysis of parton distributions and fragmentation functions, Phys. Rev. D 101 (7) (2020) 074020. arXiv:1905.03788, doi:10.1103/PhysRevD.101.074020.

[36] T.-J. Hou, et al., New CTEQ global analysis of quantum chromodynamics with high-precision data from the LHC, Phys. Rev. D 103 (1) (2021) 014013. arXiv:1912.10053, doi:10.1103/PhysRevD.103.014013.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 102–111

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Production-scale Performance Analysis and Optimization of Vlasiator

Valentin Seitz[a,*], Markus Battarbee[b], Urs Ganse[b], Vertti Tarvus[b], Minna Palmroth[b], Marta Garcia-Gasulla[a]

[a]*Barcelona Supercomputing Center, Plaça Eusebi Güell, 1-3, 08034 Barcelona, Spain*
[b]*University of Helsinki, PL 68 (Pietari Kalmin katu 5), 00560 Helsinki, Finland*

Abstract

As HPC systems scale toward the exascale era, the ability to efficiently utilize the growing number of parallelism available becomes critical. Traditional performance validation methods - such as scalability studies, and aggregated performance metrics obtained by profilers and mini-apps - often fall short when applied to complex, full-scale production runs. In this paper, we present a novel methodology for in-depth performance analysis of large-scale simulations, demonstrated on Vlasiator, a space plasma simulation code executed on up to 500 nodes of a European pre-exascale supercomputer. Our approach combines targeted instrumentation, programming model-based performance metrics, and iterative investigation to identify and address performance bottlenecks that are otherwise hidden in conventional analyses. The methodology led to optimizations, resulting in improved parallel efficiency and measurable speedups. This work highlights the importance of production-level analysis for HPC applications and provides a practical framework for similar efforts across other scientific domains.

Keywords: HPC; Performance Analysis; Vlasiator; Optimization

1. Introduction

High-Performance Computing (HPC) clusters continue to grow in size, providing access to increasing numbers of CPU cores and GPUs. At the same time, HPC applications are evolving to tackle increasingly complex simulations with higher resolution and greater physical detail. Amid this progress, it is critical to ensure the efficient use of computational resources.

HPC developers often rely on scalability curves to validate and showcase the performance of their codes. When scalability results are insufficient, profiling tools are often the next step. These tools provide information on how time is spent across various parts of the code, such as specific subroutines or key regions like I/O and MPI communication. Additional approaches include the use of mini-apps or per-kernel benchmarks.

* Corresponding author
E-mail address: valentin.seitz@bsc.es

1877-0509
10.1016/j.procs.2025.08.237

Yet, these methods often fall short of capturing the true efficiency of full-scale production runs. For example, scalability curves can obscure underlying inefficiencies if the base case itself contains issues that scale with the problem size — as we will demonstrate later in this paper. Similarly, profiling tools may not provide enough insight into the root causes of inefficiencies, and mini-apps or benchmarks might fail to reflect the complexity of a real-world simulation.

In this paper, we introduce a methodology and supporting tools to analyze the performance of a full-production plasma simulation running on up to 500 nodes of a European pre-exascale HPC system. We present the performance analysis conducted, the insights obtained, and how these led to optimizations that improved execution efficiency and resulted in meaningful speedups.

The remainder of the paper is organized as follows: Section 2 introduces Vlasiator, the HPC code, and the simulation use case analyzed. Section 3 presents the methodology and tools used for the performance study. Section 4 details the step-by-step analysis and its findings. In Section 5, we discuss the main challenges encountered and the key findings. Finally, Section 6 concludes the paper and outlines future work.

2. Vlasiator

Vlasiator [17] is a HPC plasma simulation code, designed for modeling the collisionless plasma environment surrounding Earth. It simulates the solar wind – magnetosphere – ionosphere interaction through ion-kinetic physics and response through electromagnetic fields. In Vlasiator, ions are modeled through the particle distribution function, discretized onto a Cartesian grid, with electrons contributing to the system as a charge-neutralizing fluid. Closure is achieved through the Hall MHD Ohm's law. Vlasiator is developed in C++ and parallelized using MPI and OpenMP.

Vlasiator simulates magnetospheric physics in 6 dimensions, with 3 spatial dimensions and 3 velocity dimensions (3D-3V). The spatial simulation domain for ions is discretized on a Cartesian grid utilizing the DCCRG library [13, 14] with regions of interest simulated at higher spatial resolution [8] evaluated through dynamic refinement criteria [15]. This cell-based octree refinement results in highly dynamic and nonuniform geometries contributing to the computational effort of solutions. The balancing of computational load across MPI domains is further influenced through the fact that Vlasiator utilizes a sparse velocity mesh [20] for each spatial cell, where only those regions of phase-space with non-negligible distribution function values are retained and simulated. The Vlasiator simulation grids are heterologous, with electromagnetic fields evaluated on a uniform mesh [18] and the ionospheric inner boundary solved on a further refined triangular mesh [9]. Global 3D-3V Vlasiator simulations model trillions of phase-space cells for hundreds of thousands of time steps (up to several thousands of seconds of real time), utilizing significant HPC allocations for these runs. Recent efforts in improving parallel performance of Vlasiator include developments to support heterogeneous architectures [5] and coalescing MPI transfers [4].

2.1. Configuration used in the analysis

Vlasiator simulations are highly dynamic in their computational complexity, with the spatial grid refined in a dynamic fashion with up to three levels of octree refinement of cells, and the velocity space content of cells varying by up to several orders of magnitude. To this end, parallel performance analysis was evaluated not on a simulation starting from initial conditions, but rather from a checkpoint (restart) file storing the full simulation state of a real-life production run, as showcased in Figure 1. At the analyzed time of the run, the mesh contained nearly 12 million spatial cells, of which 6.7 million were at the highest refinement level. The velocity space content of these spatial cells amounted to 1.2 trillion phase-space cells, with a maximum of ca. $5 \cdot 10^5$ phase-space cells and an average of $1 \cdot 10^5$ phase-space cells per spatial cell.

3. Methodology and Tools

3.1. Efficiency driven approach

The approach we follow in this work is driven by first obtaining programming-model-specific efficiency metrics for different resource configurations. Low values in these efficiencies are then used to guide a detailed trace-based analysis to identify the root causes of inefficiencies. This approach is similar to the one proposed by Wagner et al. [21].

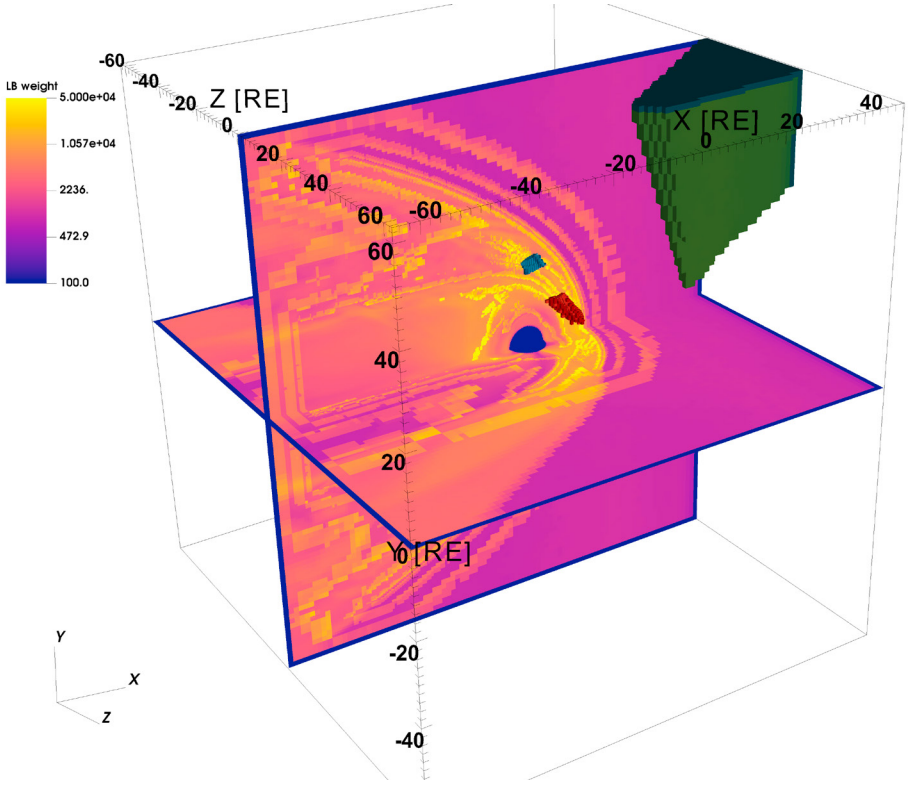


Fig. 1: Vlasiator simulation checkpoint state used for performance evaluation, with the colormap indicating the load balance weight, which is a function of spatial cell phase-space complexity and neighboring geometry. Highlighted are three MPI task domains in red, green, and blue, showcasing the heterogeneity of the computational domains. Axis labels are in units of Earth radii (RE).

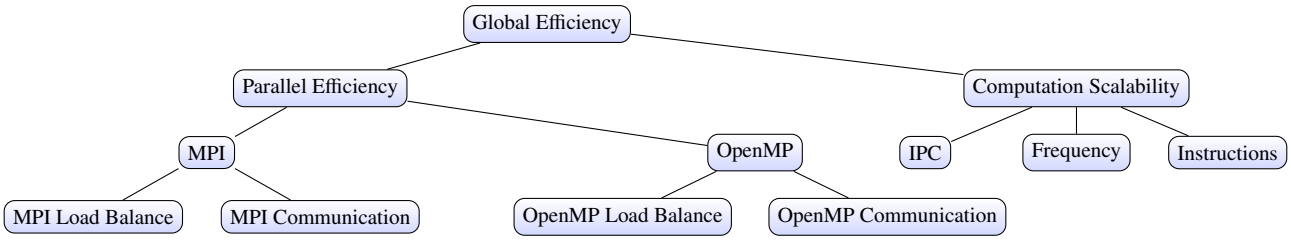


Fig. 2: Tree showing the hierarchical programming-model based performance efficiencies

Figure 2 gives an overview of the efficiency metrics that we will explore. These metrics are well-established [3] and hierarchical. Global efficiency is the product of computation scalability and the parallel efficiency of that execution. The Computation scalability term describes how the useful computation time of the workload scales when comparing different runs. This metric (and all its children) is computed relative to a base case. The computational scalability is divided into 3 factors instructions per cycle (IPC), frequency and number of executed instructions following the CPU performance equation described by Hennessy and Patterson [12]. For a strong scaling experiment, a perfect instruction scaling has the same number of instructions across all processes for all resource configurations. If work is replicated or additional work is required for more resources, the instruction scalability will quantify these effects. IPC and frequency scaling provide hints if any of those other two factors cause slowdowns or speedups relative to the base case.

Parallel efficiency is an absolute measure, it expresses how much of the time used by the execution is spent on doing useful computations vs time spent in parallel runtimes like Message Passing Interface (MPI) or Open Multi-Processing

(OpenMP). Load balance indicates how well the application distributes the required computation time across the resources. We compute it as the factor between the maximum and the average across the resources. Imbalance can arise for two different reasons. One is caused by distributing work unequally, and the other one is the work being a bit slower to perform in some CPUs. Some tools can additionally distinguish between the load balance in a node and the load balance between nodes. Communication efficiency quantifies how much time is spent in coordination or communication between resources. If there is a lot of time spent in transferring messages e.g. via MPI, this efficiency will quantify it.

3.2. Tools

An established way to get insight into a program execution and to compute these efficiency metrics is to trace the program execution and later perform a post-mortem analysis of the trace. For the generation of the execution trace, we rely on a modified version of Extrae [7]. For the visualization and analysis of the tracing data, we use Paraver [19].

The Tracking Application Live Performance (TALP) [16] component of the DLB [11] library offers a more lightweight approach to obtain the efficiency metrics. Instead of storing the traces and requiring additional post-processing, TALP computes the efficiency metrics on the fly and makes them available for the user after the execution. Using a region annotation API, which is part of TALP, it is able to compute these metrics for different regions of the application. In Appendix A we provide details on the tools and the respective versions used.

3.3. Integration with Phiprof

Vlasiator has code annotations using the Phiprof [1] library. This provides an easy entry hook for integration with the BSC performance tools set. Our approach is very similar to the already published integration with the TAU tools [6]. Compared to the TAU tools integration, we rely on another translation layer that provides a unified region-annotation interface for both Extrae, DLB, and other performance tools. This enables us to switch easily between online analysis using DLB and tracing with Extrae without the need to recompile or put any performance tool-specific code into Phiprof or Vlasiator.

3.4. Methodology

To analyze the application, we implemented some changes to the typical workflow described in [21]. Instead of selecting the focus of the analysis using an execution trace, we fix it a priori and select it to be a single timestep. As tracing at production scale becomes a major data processing challenge, we opt to first run an online analysis using TALP to obtain the efficiency metrics. Harnessing the integration with the Phiprof library, we add code regions through the TALP Application Programming Interface (API), which enables TALP to compute the efficiencies per region. We use the information obtained by TALP to drive further analysis. Regions with low efficiencies and high relative runtime are selected for an in-depth tracing analysis. For the tracing executions, we use the integration with Phiprof and store the region information in the trace using the translation layer. The translation layer also allows us to easily configure partial tracing. In partial tracing, we select some regions of interest and enable the tracing package only when in those regions. For all other regions, we only record the Phiprof annotations and region durations. The benefit of this is a significant reduction in trace size, while still keeping all information in the regions of interest. Essentially, partial tracing ensures that the tracer does not record programming-model specific calls e.g. MPI or OpenMP events while outside the region of interest.

3.5. Marenostrum 5

As HPC machines come in all architectures and sizes, we briefly introduce Marenostrum 5 [2], a EuroHPC petaflop machine that is hosted by the Barcelona Supercomputing Center. The General Purpose Partition, which was used in the analysis, consists of 3 islands with 30 racks each. Each rack is filled with 72 compute nodes. Each compute node consists of two Intel Xeon Platinum 8480+ 56-core CPUs with 224GB of usable 4800MHz DDR5 memory. Two nodes share one ConnectX-7 NDR200 InfiniBand network interface with an available bandwidth of 200Gb/s. This brings the available network bandwidth per node to 12.5 GB/s.

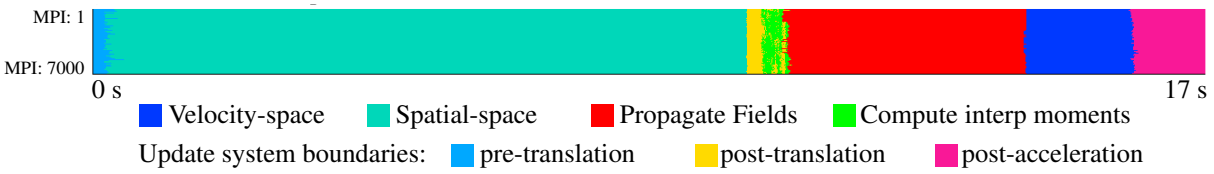


Fig. 3: Paraver timeline showing the Phiprof code annotations of the analyzed timestep in the 500 node production run.

4. Performance Analysis and Optimizations

Following the methodology introduced, we first perform a strong-scaling experiment and obtain the efficiencies using TALP. This is followed by an in-depth analysis of the current performance problems in the spatial-space updates as well as showcasing a performance improvement of the field solvers that resulted from the insights revealed in the analysis.

For all experiments included in this study, we launch 7 MPI processes per socket, that is 14 per node. We launch 8 OpenMP threads per process distribute them in a close fashion and bind them to cores while Simultaneous Multi-threading (SMT) is disabled.

If not stated otherwise, the Vlasiator version used corresponds to the `FIC` branch, which was used in the original production run. For details consult Appendix A.

4.1. Strong scaling experiment

As an initial assessment of the parallel performance, we conduct a strong-scaling experiment from 250 nodes in Marenostrum 5 up to 500 nodes. For the lower node count, we chose 250 nodes, as it was the minimum amount of nodes needed in terms of total memory to fit the simulation with some additional room for the tracing buffers. Figure 3 shows a timeline overview of the most significant Phiprof regions making up one timestep. We can see, that the `Spatial-space` updates take up roughly half of the runtime required for one timestep. The second biggest region is the field solver in the `Propagate Fields` region. This is followed by the `Velocity Space` updates, which only accounts for about 15% of runtime.

Table 1: Strong scaling efficiency and performance metrics obtained with TALP for different Phiprof regions. Efficiencies are measured without hardware counters; computation scalability is from runs with hardware counters; run times for the timestep are averages over two runs without hardware counters.

	Propagate		Spatial-space		Propagate Fields		Velocity Space	
Number of processors	28000	56000	28000	56000	28000	56000	28000	56000
MPI Parallel Efficiency	0.23	0.21	0.19	0.17	0.18	0.16	0.52	0.43
MPI Load Balance	0.51	0.53	0.35	0.39	0.31	0.27	0.54	0.51
MPI Communication Efficiency	0.46	0.40	0.55	0.43	0.58	0.63	0.97	0.84
Computation Scalability	1.00	1.07	1.00	1.08	1.00	1.05	1.00	1.01
Average IPC	1.52	1.59	0.99	1.08	2.38	2.40	1.68	1.68
Average Frequency (GHz)	2.89	2.91	2.95	2.95	2.84	2.87	2.91	2.92
Average Runtime (s)	36.8	17.3	21.9	9.40	8.42	4.71	4.82	2.64
Speedup	–	2.13	–	2.32	–	1.79	–	1.83

Table 1 shows the results from the strong-scaling experiment. We see that for the region `Propagate`, which encodes the timestep, we observe a speedup of 2.1 when doubling the available resources. While this indicates that Vlasiator benefits from additional resources slightly super-linearly, it falls short of giving the full performance picture. Focusing on the Parallel efficiency, we observe that the Parallel efficiency is not optimal sitting around 20%. This means, that

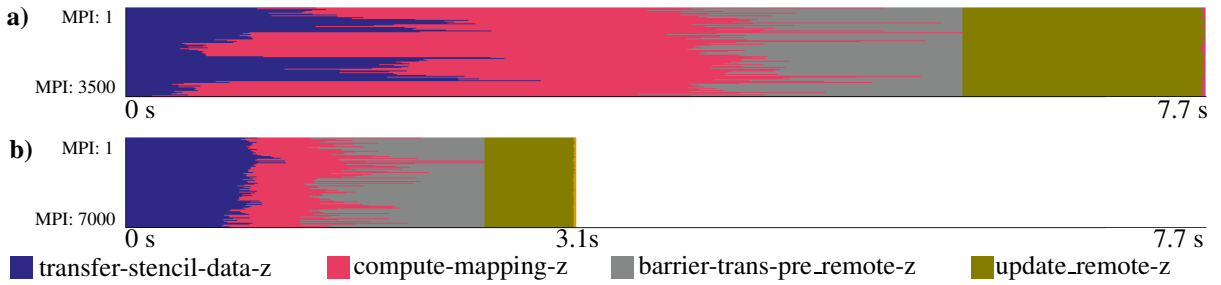


Fig. 4: Paraver timeline showing the Phiprof regions for the spatial space updates in z direction. **a)** shows the 250-node execution. **b)** the 500-node execution.

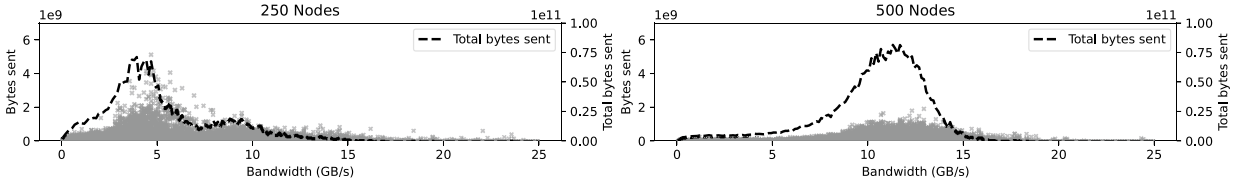


Fig. 5: Effective send-bandwidth achieved by the different nodes in the two execution configurations. Each point corresponds to a node sending a certain amount of data at a certain effective bandwidth. The dashed line shows the total data sent by all nodes at that bandwidth. We only consider the initial transfer of the spatial space update in z direction. Both plots include sends that did not go through the network fabric but remained in-node.

of the overall available CPU time only roughly 20% is spent executing application code, the rest is spent in MPI runtime code. As the GNU-OpenMP runtime currently does not support the OMPT interface, but TALP requires that for intercepting the runtime calls, the Table only contains the MPI related efficiencies. We can observe that the `Velocity Space` region exhibits a significantly higher parallel efficiency compared to the other 2 regions. We will therefore focus the following in-depth analysis only on the spatial space updates and the field solver implementation in the `Propagate Fields` region.

4.2. Spatial space

The `Spatial space` computations are split into three very similar regions, each concerned with updating the z, y and x direction of real space advection. As they exhibit very similar performance characteristics, we will focus on just the z-dimension, but our findings generalize to the y and x updates as well.

Computing the real space direction advection consists of three steps: first copying over stencil data from other MPI processes to form the ghost regions, followed by the actual stencil computations. The semi-Lagrangian solver results in integrated plasma contributions for both local and ghost cells, and as such, in the final step, these contributions spilling into remote neighbors must be communicated as *remote neighbor contributions*. Communication is implemented by instantiating the non-blocking send and receive pairs for each communication partner. This is followed by the `MPI_Waitall` for every communication receive request. After that, the `MPI_Waitall` calls for the send requests are issued. This communication structure is used for both the initial and later update transfers.

Considering the `Spatial-Space` column in Table 1 it becomes obvious that both load balance between the MPI processes and MPI communications seem to be major sources of inefficiency in this region. Load balance seems to improve slightly at 500 nodes, while communication efficiency drops. We also observe a super-linear speedup when moving to 500 nodes. We first focus on the communication part, followed by an analysis of the stencil computations.

4.2.1. Communication

Looking at the 250-node timeline in Figure 4 we see that region `transfer-stencil-data-z` is significantly imbalanced. The `compute-mapping-z` region is therefore delayed and reaches the synchronization in

the `barrier-trans-pre-update_remote-z` later. The fastest process finishes the MPI communication in 0.19s whereas the slowest takes 3.2s and the average time spent in transferring data is around 2.3s for the 250 nodes execution. Looking at Figure 5 we can see, that in the 250-node execution most nodes communicate data at an effective bandwidth between 3 and 5 GB/s with a few at around 9-10 GB/s. For the 500-node execution, most data is transferred at around 12 GB/s, and no bi-modality of the distribution is observed. The reason for the load balance problem in the initial transfers of the 250-node execution originates from the inability of some nodes to use the available bandwidth.

Based on the communication pattern and the available metrics the most likely reason for the contention, which is observable at 250 nodes, is either related to maximum message size or incoming message ordering in the MPI implementation or the underlying transport layer. In a more current version of Vlasiator the `MPI_Waitall` is called only once for all receives and later once for all the send requests. This removes the necessity for the MPI runtime to retire the messages in issue order. Unfortunately, these changes did not show any improvements in bandwidth utilization for the 250-node execution. This hints strongly towards contention in the MPI or networking stack as the reason for the low bandwidth utilization.

4.2.2. Load imbalance in the stencil computations

Looking again at Figure 4 focusing on the 500 node execution timeline, we can identify a significant imbalance in the `compute-mapping-z` region. This is, compared to the 250-node execution, easier to identify, as the transfers are much more balanced. TALP is able to distinguish the quality of intra-node and inter-node load balance separately. For the whole `Spatial Space` region TALP reports the inter-node load balance efficiency to be 0.82 and for the most-imbalanced node 0.48. So the load imbalance in the nodes seems to be dominating this inefficiency.

To summarize: The performance of the `Spatial-space` region is mainly dominated by network transfers and process-level load imbalance. Uneven bandwidth utilization can lead to additional unfavorable imbalance and contribute significantly to inefficiencies.

To improve communication efficiency, a scheme overlapping communication and computation, thus hiding communication overhead would be beneficial at production scale. In such a scheme, as soon as a portion of data becomes ready, the corresponding stencil computations are performed. As the load imbalance seems to not be dominated by inter-node load balance, dynamic intra-node load balancing approaches like Lend When Idle (LeWI) [10] should certainly be explored.

4.3. Field solver

After the analysis of the spatial-space updates, we now present the analysis of the field-solver implementation. The computational structure of the `Propagate Fields` region is to first compute the grid coupling. This links MPI processes to each other for the contribution of plasma moments required by the field solver, as the grids are heterologous [18]. After computing the coupling, the iterative part of the field solver, repeated potentially several times follows. After electric and magnetic fields have been solved for the new simulation time state, these are communicated back through coupling so the Vlasov solvers can use them in solving the Vlasov equation.

As the iterative part takes up most of the runtime, we will focus our analysis on this region. Computationally the iterative part is organized in OpenMP parallel worksharing loops with non-blocking communication between the parallel regions. This communication is however directly awaited, so no overlap occurs. Communication in the iterative part is dominantly between neighboring processes. At the end of each iteration global synchronization is enforced by a `MPI_Allreduce`.

Looking at Table 1 we find that load balance is the biggest contributor to inefficiency and gets worse when scaling to large node counts. Communication efficiency, however, seems to improve, but overall parallel efficiency does still suffer from scaling the application.

4.3.1. Analysis of the load imbalance

Selecting the second iteration and only focusing on the process that arrived the latest in the `MPI_Allreduce` and its neighbors we can see the structure of the computation depicted in Figure 6 a). Looking at the second block of the field solver in Figure 6 a), we clearly see that two threads have a high useful duration indicated by the blue lines starting in the second phase. This high computation time is identified to be an imbalanced OpenMP work distribution in the handling of the magnetic field boundary. As these computations need to be finished before communicating, we can

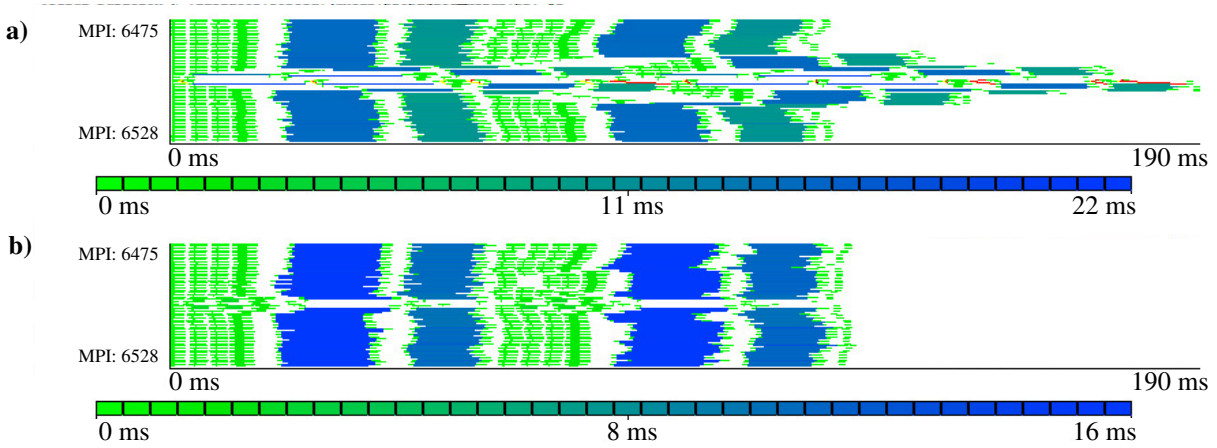


Fig. 6: Paraver timeline view of the granularity between runtime calls of one field solver iteration. Higher values mean longer computation between runtime calls. **a)** shows the original code version, **b)** is the same iteration using the optimized version. Both *x*-axis have the same length allowing for easy comparison.

see the effect of this initial delay propagating through the communication dependency chain and affecting neighboring processes. These initial delays in the magnetic field boundary handling de-synchronize the lockstep pattern and lead to low load balance values. To conclude, we can say that the field solver's performance currently is limited by load balance issues. The small disturbance due to improper worksharing combined with the fine-grained collective calls at the end of each iteration leads to low load balance efficiencies.

4.3.2. Performance improvement

Using this insight, we were able to provide a fix for this uneven work distribution. Details on the implantation are provided in Appendix A. Applying the changes and re-running resulted in Figure 6 b) which shows, that the de-synchronization is no longer visible and the execution of the iteration takes significantly less time.

Looking at the load balance value, we also see a significant improvement from 0.27 to 0.4 reported by TALP while execution time for this iteration dropped from 0.19 to 0.14 seconds. The overall runtime improvement of the `Propagate Fields` region for that timestep is about 10%.

5. Discussion

Any performance improvement efforts should be driven by an in-depth performance analysis. Gathering utilizable performance data for massively parallel software at production scale using full tracing approaches is not feasible. The changes to the analysis methodology presented in this work did prove to be effective, as we were able to use the information provided by TALP to guide our partial tracing efforts. However, as already pointed out, TALP uses the OMPT interface, which is not implemented by the GNU-OpenMP runtime. This leads to the OpenMP runtime information being invisible to the tool, only reporting MPI efficiencies. This potentially leads to an overestimation of efficiencies. For our analysis, this however posed no problem, as most of the inefficiencies are observed at MPI level. Another downside of using profilers like TALP is that the aggregation of measurement data between the invocations of the same region leads to a smearing effect. Especially for non-production runs, where the number of timesteps is rather small and timestep performance varies a lot, it can lead to hard-to-interpret results.

Compared to online analysis approaches like TALP, tracing can resolve these transients and provide data at a higher resolution without aggregations. At scale, however, the growth of data to store and post-process quickly becomes a major bottleneck. The average trace size of an execution in our analysis was 500 GB, which is at the boundary of being manageable and analyzable. If we followed a more typical approach not relying on the online analysis to guide the partial tracing efforts, we expect to obtain multiple TB of data per execution, which is prohibitive to perform. Thus, this approach of guided partial tracing is validated as a useful approach when profiling at scale.

Tracing applications at scale not only comes at the cost of storage and post-processing times but as tracing tools intercept runtime calls, they often find themselves in the critical path of the application, so overhead needs to be tightly monitored to ensure reliable data. For all traces obtained, we observed an overhead of 6-8% for the timestep with programming model call interceptions enabled. For the rest of the execution where only Phiprof data was added, we did not observe any significant overhead. We always executed a reference run to compute the overhead in the same node set, to reduce the machine-specific influence on the performance differences between runs. For the TALP executions we observed an overhead of around 8%.

The findings in the `Spatial Space` translation were not unexpected as balancing the load in a highly sparse and dynamic mesh is a major challenge. The bandwidth utilization-induced load imbalance, however, was something that we observed now for the first time. Identifying a slowdown as being caused by actual network bandwidth was only possible due to the utilized tools - timing information alone would not be able to provide this information. As the network bandwidth available also depends on how busy the machine is and the underlying network topology, we studied if this has any influence. Our data currently indicates that under-utilization of bandwidth is independent of the networking topology, as well as machine load. Here a more detailed analysis using the e.g. the MPI tools interface or network hardware counters would allow us to analyze the reasons for contention in the networking stack. It also remains to be seen if this is a common feature of the MPI usage of Vlasiator, or if this phenomenon is restricted to just this machine and MPI implementation.

6. Conclusion and outlook

In this paper we showcased a detailed performance analysis, for only select portions of a production-scale simulation run, driven by parallel efficiency metrics obtained by TALP. Region-based partial tracing enabled us to study the root causes of the low-efficiency values and identify current bottlenecks in Vlasiator while keeping trace size and post-processing times at a manageable level. For the spatial space updates, we identify communication and inter-process load imbalance to be the main contributors to inefficiencies. In the field solver, we identified an unfavorable work distribution among threads that leads to high idle times due to frequent global synchronizations. We were able to mitigate this issue partially and improve the field solver performance for that case by roughly 10%.

For future work, we plan to extend the collaboration and evaluate the possibility of an TALP integration into Vlasiator. We also plan to tackle the in-node component of the load imbalance in the spatial space translation by investigating the effects of LeWI in this case, and the implications for performance at production scale.

Our approach highlights the importance of using online analysis tools as they provide enough insight to uncover hidden inefficiencies while keeping the postprocessing effort to a minimum. Our findings also indicate how performance analysis at scale is a crucial step that should not be ignored, as the mental model of the execution and the actual execution behavior can diverge, especially when utilizing a larger portion of a supercomputer in a realistic simulation scenario.

Acknowledgments

This work have received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931 (POP3 project) and No 101093261 (Plasma-PEPSC project). The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal, the Czech Republic, Sweden, Finland, Greece, and Slovenia. The Plasma-PEPSC project (PCI2022-135050-2) and the POP3 project (PCI2024-153419) are also co-funded by MICIU/AEI /10.13039/501100011033 and by the UE NextGenerationEU/PRTR. Access to the computing resources was provided by the EuroHPC Development Access Call EHPC-DEV-2025D05-038

Appendix A. Reproducibility

We provide a repository to ease the reproducibility of our findings under: https://gitlab.bsc.es/vseitz/performance-analysis-of-vlasiator-at-production-scale. For access to raw data like traces or profiler data, please contact the corresponding author.

References

[1] von Alfthan, S., 2023. Phiprof – parallel hierarchical profiler. https://github.com/fmihpc/phiprof.

[2] Banchelli, F., Garcia-Gasulla, M., Mantovani, F., Vinyals, J., Pocurull, J., Vicente, D., Eguzkitza, B., Galeazzo, F.C.C., Acosta, M.C., Girona, S., 2025. Introducing MareNostrum5: A European pre-exascale energy-efficient system designed to serve a broad spectrum of scientific workloads. URL: http://arxiv.org/abs/2503.09917, doi:10.48550/arXiv.2503.09917. arXiv:2503.09917 [cs].

[3] Banchelli, F., Peiro, K., Querol, A., Ramirez-Gargallo, G., Ramirez-Miranda, G., Vinyals, J., Vizcaino, P., Garcia-Gasulla, M., Mantovani, F., 2020. Performance study of hpc applications on an arm-based cluster using a generic efficiency model, in: 2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), IEEE. pp. 167–174. doi:10.1109/PDP50117.2020.00032.

[4] Battarbee, M., Ganse, U., Pfau-Kempf, Y., Alho, M., Papadakis, K., Palmroth, M., 2025. Coalescing MPI communication in 6D Vlasov simulations: solving ghost domains in Vlasiator. URL: http://arxiv.org/abs/2504.21450, doi:10.48550/arXiv.2504.21450. arXiv:2504.21450 [physics].

[5] Battarbee, M., Papadakis, K., Ganse, U., Hokkanen, J., Kotipalo, L., Pfau-Kempf, Y., Alho, M., Palmroth, M., 2024. Porting the grid-based 3D+3V hybrid-Vlasov kinetic plasma simulation Vlasiator to heterogeneous GPU architectures. URL: http://arxiv.org/abs/2406.02201, doi:10.48550/arXiv.2406.02201. arXiv:2406.02201 [physics].

[6] Coti, C., Pfau-Kempf, Y., Battarbee, M., Ganse, U., Shende, S., Huck, K., Rodriquez, J., Kotipalo, L., Faj, J., Williams, J.J., Peng, I., Malony, A.D., Markidis, S., Palmroth, M., 2024. Integration of Modern HPC Performance Tools in Vlasiator for Exascale Analysis and Optimization, in: 2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 996–1005. URL: https://ieeexplore.ieee.org/document/10596357, doi:10.1109/IPDPSW63119.2024.00170.

[7] Extrae team, . Extrae | BSC-Tools. URL: http://web.archive.org/web/20240830235900/https://tools.bsc.es/extrae/.

[8] Ganse, U., Koskela, T., Battarbee, M., Pfau-Kempf, Y., Papadakis, K., Alho, M., Bussov, M., Cozzani, G., Dubart, M., George, H., Gordeev, E., Grandin, M., Horaites, K., Suni, J., Tarvus, V., Kebede, F.T., Turc, L., Zhou, H., Palmroth, M., 2023. Enabling technology for global 3D + 3V hybrid-Vlasov simulations of near-Earth space. Physics of Plasmas 30, 042902. URL: https://pubs.aip.org/pop/article/30/4/042902/2878614/Enabling-technology-for-global-3D-3V-hybrid-Vlasov, doi:10.1063/5.0134387.

[9] Ganse, U., Pfau-Kempf, Y., Zhou, H., Juusola, L., Workayehu, A., Kebede, F., Papadakis, K., Grandin, M., Alho, M., Battarbee, M., Dubart, M., Kotipalo, L., Lalagüe, A., Suni, J., Horaites, K., Palmroth, M., 2025. The Vlasiator 5.2 ionosphere – coupling a magnetospheric hybrid-Vlasov simulation with a height-integrated ionosphere model. Geoscientific Model Development 18, 511–527. URL: https://gmd.copernicus.org/articles/18/511/2025/, doi:10.5194/gmd-18-511-2025.

[10] Garcia, M., Labarta, J., Corbalan, J., 2014. Hints to improve automatic load balancing with LeWI for hybrid applications. Journal of Parallel and Distributed Computing 74, 2781–2794. URL: https://www.sciencedirect.com/science/article/pii/S0743731514000926, doi:10.1016/j.jpdc.2014.05.004.

[11] Garcia-Gasulla, M., Lopez, V., 2025. Dynamic load balancing library. https://doi.org/10.5281/zenodo.15000657. URL: https://pm.bsc.es/dlb, doi:10.5281/ZENODO.15000657.

[12] Hennessy, J.L., Patterson, D.A., 2011. Computer Architecture: A Quantitative Approach. 5th ed., Morgan Kaufmann. URL: https://dl.acm.org/doi/book/10.5555/1999263.

[13] Honkonen, I., Vlasiator Team, . Dccrg – a distributed cartesian cell-refinable grid. URL: https://github.com/fmihpc/dccrg/tree/vlasiator-version.

[14] Honkonen, I., Von Alfthan, S., Sandroos, A., Janhunen, P., Palmroth, M., 2013. Parallel grid library for rapid and flexible simulation development. Computer Physics Communications 184, 1297–1309. URL: https://linkinghub.elsevier.com/retrieve/pii/S0010465512004237, doi:10.1016/j.cpc.2012.12.017.

[15] Kotipalo, L., Battarbee, M., Pfau-Kempf, Y., Palmroth, M., 2024. Physics-motivated cell-octree adaptive mesh refinement in the Vlasiator 5.3 global hybrid-Vlasov code. Geoscientific Model Development 17, 6401–6413. URL: https://gmd.copernicus.org/articles/17/6401/2024/, doi:10.5194/gmd-17-6401-2024.

[16] Lopez, V., Ramirez Miranda, G., Garcia-Gasulla, M., 2021. TALP: A Lightweight Tool to Unveil Parallel Efficiency of Large-scale Executions, in: Proceedings of the 2021 on Performance EngineeRing, Modelling, Analysis, and VisualizatiOn STrategy, ACM, Virtual Event Sweden. pp. 3–10. URL: https://dl.acm.org/doi/10.1145/3452412.3462753, doi:10.1145/3452412.3462753.

[17] Palmroth, M., Ganse, U., Pfau-Kempf, Y., Battarbee, M., Turc, L., Brito, T., Grandin, M., Hoilijoki, S., Sandroos, A., von Alfthan, S., 2018. Vlasov methods in space physics and astrophysics. Living Reviews in Computational Astrophysics 4, 1. URL: https://doi.org/10.1007/s41115-018-0003-2, doi:10.1007/s41115-018-0003-2.

[18] Papadakis, K., Pfau-Kempf, Y., Ganse, U., Battarbee, M., Alho, M., Grandin, M., Dubart, M., Turc, L., Zhou, H., Horaites, K., Zaitsev, I., Cozzani, G., Bussov, M., Gordeev, E., Tesema, F., George, H., Suni, J., Tarvus, V., Palmroth, M., 2022. Spatial filtering in a 6D hybrid-Vlasov scheme to alleviate adaptive mesh refinement artifacts: a case study with Vlasiator (versions 5.0, 5.1, and 5.2.1). Geoscientific Model Development 15, 7903–7912. URL: https://gmd.copernicus.org/articles/15/7903/2022/, doi:10.5194/gmd-15-7903-2022.

[19] Pillet, V., Labarta, J., Cortes, T., Girona, S., . Paraver: A tool to visualize and analyze parallel code .

[20] Von Alfthan, S., Pokhotelov, D., Kempf, Y., Hoilijoki, S., Honkonen, I., Sandroos, A., Palmroth, M., 2014. Vlasiator: First global hybrid-Vlasov simulations of Earth's foreshock and magnetosheath. Journal of Atmospheric and Solar-Terrestrial Physics 120, 24–35. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364682614001916, doi:10.1016/j.jastp.2014.08.012.

[21] Wagner, M., Mohr, S., Gimenez, J., Labarta, J., 2018. A Structured Approach to Performance Analysis. doi:10.1007/978-3-030-11987-4_1.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 112–123

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Towards Exascale Computing for Astrophysical Simulation Leveraging the Leonardo EuroHPC System

Nitin Shukla[a,*], Alessandro Romeo[a,*], Caterina Caravita[a], Michael Redenti[a], Radim Vavrik[b], Lubomir Riha[b], Andrea Mignone[c], Marco Rossazza[c], Stefano Truzzi[c], Luca Tornatore[d], Antonio Ragagnin[d], Tiago Castro[d], Geray S. Karademir[e], Klaus Dolag[e], Pranab J. Deka[f], Fabio Bacchini[f], Rostislav-Paul Wilhelm[f], Daniele Gregori[g], Elisabetta Boella[g]

[a] *CINECA, Casalecchio di Reno, Italy*
[b] *IT4Innovations, VSB-TU Ostrava, Ostrava, Czech Republic*
[c] *University of Turin, Turin, Italy*
[d] *INAF - Osservatorio Astronomico di Trieste, Trieste, Italy*
[e] *Universitäts-Sternwarte, Fakultät für Physik, Ludwig-Maximilians-Universität München, Munich, Germany*
[f] *KU Leuven, Leuven, Belgium*
[g] *E4 Computer Engineering SpA, Scandiano, Italy*

Abstract

Developing and redesigning astrophysical, cosmological, and space plasma numerical codes for existing and next-generation accelerators is critical for enabling large-scale simulations. To address these challenges, the SPACE Center of Excellence (SPACE-CoE) fosters collaboration between scientists, code developers, and high-performance computing experts to optimize applications for the exascale era. This paper presents our strategy and initial results on the Leonardo system at CINECA for three flagship codes, namely `gPLUTO`, `OpenGadget3` and `iPIC3D`, using profiling tools to analyze performance on single and multiple nodes. Preliminary tests show all three codes scale efficiently, reaching 80% scalability up to 1,024 GPUs.

Keywords: Exascale; EuroHPC; HPC; astrophysics; space plasma; simulation; performance analysis; CI/CD; GPU scaling; SPACE CoE

1. Introduction

High-performance computing (HPC) has transformed large-scale astrophysical simulations, enabling researchers to study complex phenomena such as dark matter interactions [1, 2], gravitational lensing [3, 4], magnetic reconnection [5, 6, 7], X-ray emissions from galaxy clusters [8], and γ-ray bursts driven by fireball beam–plasma interactions

* Corresponding author.
E-mail addresses: n.shukla@cineca.it (Nitin Shukla), a.romeo@cineca.it (Alessandro Romeo)

1877-0509
10.1016/j.procs.2025.08.238

[9, 10]. Advances in HPC hardware, particularly in GPU architectures and high-speed interconnect technologies, are driving progress towards exascale computing (10^{18} calculations per second). These improvements enable significantly larger, more detailed, and faster simulations. To fully exploit this capability, scientific codes need to adopt new programming models. The SPACE-CoE project brings together scientists, developers, and tech experts to redesign astrophysical software for exascale systems. It focuses on developing efficient, portable, and scalable applications, encouraging collaboration in the European astrophysics community through shared tools, data standards, and development practices.

The SPACE-CoE [11] focuses on adapting seven widely used open-source astrophysics codes, covering magneto-hydrodynamics, cosmology, general relativity, and space plasma physics, including `gPLUTO` [12], `OpenGadget3` [13], and `iPIC3D` [14], for which CINECA co-develops GPU-centric algorithms. While these codes are already optimized for CPU parallelism, leveraging the full power of modern EuroHPC GPU-based super computers requires further optimization to enhance performance and scalability. CINECA plays a key role in code redesign, optimization, profiling, and performance analysis, as well as supporting the project on the Leonardo system. This paper is organized as follows: Section 2 gives an overview of the codes and their POP3 [15] analysis. Section 3 covers GPU-focused development and code validation after each change. Section 4 outlines our benchmarking process and software engineering practices adapted for HPC. We present profiling and performance results for each application.

2. Code development workflows

In this section, we describe our collaborative work with the three European research teams involved in the development activity. We also provide a brief overview of the main features of the codes and how they leverage GPU computational capabilities. The three aforementioned astrophysical codes tackle a diverse range of scientific problems, involving different algorithmic methodologies and approaches.

2.1. Implementation of *gPLUTO*

`gPLUTO` is the GPU-enabled version of `PLUTO`, a multi-algorithm framework for solving the equations governing plasma dynamics at high Mach numbers. It supports a range of physical models, including the compressible Navier-Stokes equations, ideal Magneto Hydro Dynamics (MHD), relativistic MHD (RMHD) and resistive relativistic MHD (ResRMHD). The new code version is implemented in `C++` and employs Godunov-type finite volume solvers for hyperbolic and parabolic MHD conservation laws in up to three spatial dimensions, on both static and mapped grids. Currently, `gPLUTO` implements approximately 65% of the modules and features available in the original `PLUTO` code. Many of the missing features of `PLUTO`, such as adaptive mesh refinement (AMR) grids, heating and cooling mechanisms, dissipation process, etc., are being ported to the GPU. The code uses `MPI` to handle multiple tasks, while GPU acceleration is handled through `OpenACC` (Rossazza et al., submitted). GPU offloading using `OpenMP` is currently in progress and will be included in a future release of the code.

2.2. Implementation of *OpenGadget3*

`OpenGadget3` (Dolag et al., in prep.) is an open-source version of a `Gadget2` [16] code successor. It is a N-Body `C/C++` code and deals with several astrophysical scales. Gravitational force uses Barnes & Hut algorithm [17] for short range interactions and particle mesh algorithm to treat large scales, performed with the use of `FFTW3` libraries [18]. The code employs an advanced smoothed particle hydrodynamics (SPH) solver [19]. As an alternative hydro solver, the code is also equipped with a meshless finite mass method [13]. Moreover, this code also includes other baryonic physics processes, such as radiative cooling, star formation, energy feedback, radiative transfer, magnetic fields, and black holes. The code uses a hybrid `MPI + OpenMP` parallelization scheme, exploiting Hilbert space filling curve locality [20], and both inter-node and intra-node parallelism. GPU offloading is currently implemented via `OpenACC` [21], while support for `OpenMP` offloading is under development.

2.3. Implementation of iPIC3D

iPIC3D [14] is a semi-implicit Particle-in-Cell (PIC) C/C++ code developed primarily to study collisionless plasma dynamics at kinetic scales. Macro-particles, which are used to represent an ensemble of plasma particles, are evolved in a Lagrangian framework, whereas the moments (such as plasma density, current, pressure, etc...) and the self-consistent electric and magnetic fields are tracked on an Eulerian grid. The three main kernels of iPIC3D are Particle Mover, Moment Gatherer, and Field Solver. Due the implicit nature of the underlying algorithms, unlike explicit PIC methods, insufficiently resolved scales do not result in numerical instabilities. This allows us to choose time step sizes and spatial grid sizes 10–100 times greater than those used in traditional explicit PIC codes. iPIC3D is an OpenMP + MPI hybrid code, and two of its modules, the particle mover and moment gatherer, are offloaded to GPU using HIP (for AMD GPUs) and CUDA (for NVIDIA GPUs).

2.4. Leonardo HPC system and allocation of resources

The initial activities of the SPACE-CoE project was conducted on the EuroHPC Tier-0 supercomputer Leonardo, hosted and managed by CINECA. Leonardo consists of two main partitions [22]: the DCGP partition with 1,536 dual-socket nodes, each equipped with 112-core Intel Xeon 8480+ CPUs, 512 GB RAM, and 3.84 TB NVMe SSDs; and the Booster partition with 3,456 accelerated nodes, each with a 32-core Intel Xeon 8358 CPU and four customized NVIDIA A100 GPUs, totaling 13,824 GPUs. Each Booster node is equipped with 512 GB of memory and 64 GB of HBM2e memory per GPU. The GPUs are interconnected via NVLink 3.0 and connected to the host system through PCIe Gen4. Inter-node communication uses InfiniBand HDR with four 100 Gb/s ports per node. Leonardo Dragonfly+ network topology ensures high-bandwidth, low-latency connectivity through a two-level fat tree structure with spine and leaf switches, supporting scalable performance for large-scale simulations. Leonardo offers a pre-exascale computing capability of up to 240 Pflop/s.

Each SPACE-CoE code needs specific libraries, managed by the SPACK package manager [23]. These are organized into modules for different compilers and MPI versions. In addition to GNU compilers, DCGP uses Intel OneAPI compilers, while Booster uses NVIDIA HPC and CUDA compilers for GPU tasks. Computational resources on Leonardo are provided through allocations from ISCRA and EuroHPC, enabling jobs of up to 128 nodes on DCGP and up to 256 nodes on Booster. Additionally, CINECA has allowed larger runs during special *pre-exascale days*.

2.5. CPU performance assessment with Extrae

This section focuses on evaluating CPU code performance prior to any GPU adaptation. As the first step, we assessed the current status of the codes using the HPC performance analysis suite based on Extrae [24]. The profiling results, obtained in collaboration with IT4I, provide insights into code behavior and performance bottlenecks, forming the basis for optimization and porting to the Leonardo HPC system.

The performance analysis process is similar for most project codes. It begins with carefully selecting and setting up test cases for measurement. To get the most useful results of the analysis for the following optimization work, it is important to use a production-level test case that performs all the required computational routines ideally at the target scale in terms of computational resources, i.e. number of nodes, CPU cores, GPU accelerators, etc. At that scale, the typical simulation runs with the duration of many hours would generate an enormous amount of performance data. To decrease the size of the collected data to a manageable level, it is necessary to limit the simulations only to several units or tens of iterations, e.g. time steps, keeping the target scale.

Among the three CoE codes only OpenGadget3 and iPIC3D exhibited scalability issues with potential for improvement. In contrast, the CPU version of gPLUTO (PLUTO) had already undergone some optimization before the start of the SPACE project. As a result, no further scalability analysis or optimization was required for PLUTO at the beginning of the SPACE project.

Manual navigation and identification of relevant sections of complex simulation codes through the collected performance data tend to be very time-consuming work and prone to inaccuracies. Both issues might be addressed by instrumenting the selected regions of interest (RoI), meaning inserting simple non-intrusive probes of the particular tracing tool into the source code, typically at the beginning and end of the region.

Such an instrumented code and configured test case are used to perform the actual tracing. For `OpenGadget3` and `iPIC3D` analysis the following tool-chains were used on Leonardo DCGP: `GCC 12.2.0`, `OpenMPI 4.1.6`, `Extrae 4.0.6`, `HDF5 1.14`, `FFTW 3.3.10` (OG), `OpenBLAS 0.3.24` (OG), `GSL 2.7.1` (OG), `PETSc 3.21.1` (IP).

2.5.1. *OpenGadget3*

In the case of `OpenGadget3`, we simulated a box with a cosmological size of 30 Mpc/h and 256^3 particles were used. For this analysis, simulations were performed using from 1 to 16 nodes, where both gravity and hydrodynamics were enabled. The last 15 time steps were selected as RoI. Figure 1 left panel shows that the speedup of RoI gradually diverges from the optimum, with the efficiency below 80% on 8 nodes (896 `MPI` processes and 7 `OpenMP` threads per process). The hierarchical multiplicative model of efficiency metrics on Figure 1 right panel identifies Serialization efficiencies (SerE) and `OpenMP` Communication efficiencies (OCE)[1] as the main limiting factor of the RoI scaling. The detailed explanation of the model and the metrics can be found at POP3 CoE learning materials [15].

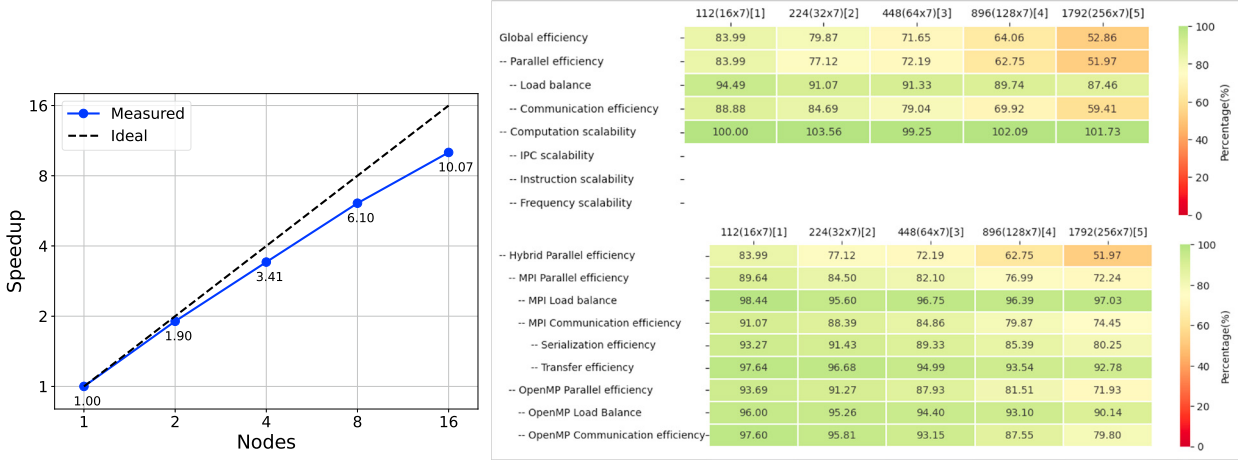


Fig. 1: Extrae anlysis of `OpenGadget3` on Leonardo DCGP for up to 16 nodes: strong scaling (left panel) and efficiency metrics [%] (right panel). Note that the computation scalability sub-metrics were unavailable at this time due to incompatible tool suite components.

Each RoI time step consists of the following set of high-level routines that were instrumented and further analyzed: Domain decomposition intensity decision (DD) subdivided in the decision criteria computation (DD1), and the actual execution (DD2), which can either perform a full domain decomposition or only transfer particles that moved out of their domain; Gravitational accelerations (GRAV); Densities (DENS); Hydro-accelerations (HYDRO); Non-standard physics (PHYS).

The SerE describes any loss of efficiency due to dependencies between processes causing alternating processes to wait. The lowest SerE (80%) was observed in the DD2 routine; however, due to its relatively short execution time, this routine is not the primary contributor to performance degradation in this test case. The GRAV routine, although the most time-consuming, exhibits a high SerE of 92%, making it a low priority for SerE optimization. In contrast, the DENS routine, being the second longest and having a SerE of 86%, emerges as the most promising candidate for optimization. A simulation assuming an ideal network with infinite bandwidth and zero latency, where messages are transferred instantaneously, showed that each process spends approximately 12% of its runtime waiting in the `MPI_Alltoall` function for dependent processes.

Regarding the OCE, which captures the synchronization and scheduling overhead induced by the `OpenMP constructs`, the lowest values 36% and 24% were observed in DD1 and DD2 respectively, though, being the shortest routines, they are probably not worth optimizing with the given test case. On the other hand, rather small room for improvement remains in the longest GRAV routine showing the best OCE 93%. The best candidate for optimization is again the DENS routine with OCE 80%, followed by HYDRO with 88% and much shorter PHYS with 67%. In

[1] See https://co-design.pop-coe.eu/metrics/hm/omp_communication_efficiency.html

DENS, each process spends a significant part of the total execution time in `OpenMP` runtime due to very small granularity of the parallel functions, mostly in `find_hsml` and `compute_unified_gradients` functions with 24% and 10% in average, respectively.

2.5.2. *iPIC3D*

We consider the test case of a Maxwellian distribution with 95^2 particles per cell with 2 species on a 2D grid with a resolution of 2240×1120 cells. The simulations were limited to four cycles and were conducted using between 2 and 64 nodes, with the second cycle selected as RoI. The strong scaling test in Figure 2 left panel reveals a clear speedup degradation on 64 nodes (7168 processes). The efficiency metrics in the right panel of Figure 2 point to drop in Transfer efficiency (TE) caused by an extreme growth of time in `MPI` communication and its latency.

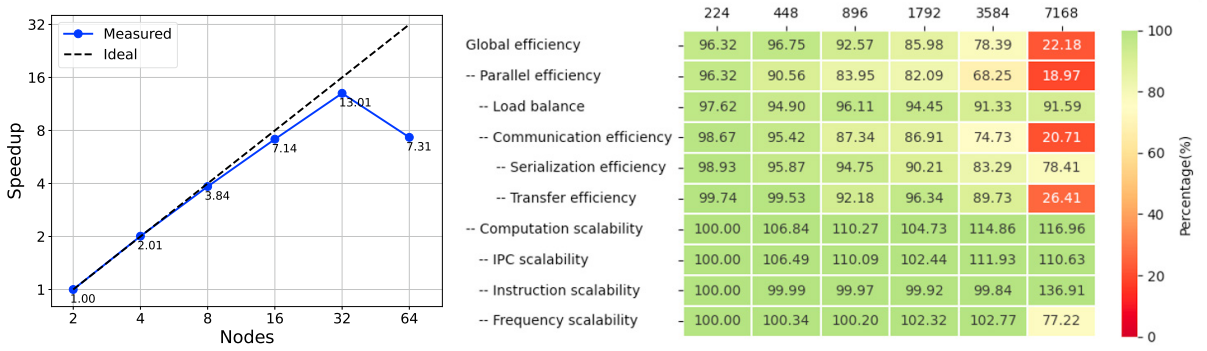


Fig. 2: Extrae anlysis of `iPIC3D` on Leonardo DCGP for up to 64 nodes: strong scaling (left panel) and efficiency metrics [%] (right panel).

Each cycle (RoI) is composed by the three high-level routines: CalculateField, ParticlesMover, and GatherMoments. The strongest TE deterioration at 64 nodes comes from CalculateField with 2%, followed by GatherMoments with 41%. That 2% TE together with 20% of Instruction scalability and other inefficiencies translates, in practice, into 0.2× speedup, i.e. 5× slowdown of the routine compared to the base run. This is caused by an excessive use of `MPI_Barriers` bounding sequences of very small `MPI_Sendrecv_replace` messages with no computation in between, resulting in significant accumulated latency of the `MPI` calls. Conversely, the ParticlesMover shows very good efficiencies and almost perfect scaling. In all routines, very high sensitivity for the system preemption is also detected.

3. Specific GPU offloading strategies

To leverage the compute capability of GPUs, all three codes adopt different numerical techniques, as each code is unique and uses distinct algorithms, each with its own set of challenges, performance bottlenecks, and numerical requirements. Additionally, these codes consist of numerous interlinked source files and are designed to simulate a wide range of physical mechanisms. As a result, optimizing or offloading to the GPU is not straightforward without focusing on a simplified, yet representative, core of the application. Therefore, we decided to extract mini-apps from the main applications, identify the relevant kernels, and analyze them to improve the overall performance of the full simulation.

Despite in the diversity numerical techniques, porting code to heterogeneous systems mainly relies on two principles: exposing parallelism and minimizing CPU-GPU data transfer. Data locality is key to GPU performance, as it minimizes memory transfers by running compute-heavy code on the GPU. Best practices include using private variables for thread independence and coalesced memory access for efficient bandwidth use. However, performance is still limited by numerical algorithms. For instance, `OpenGadget3` is memory-bound and needs new algorithms to fully utilize GPUs, while `gPLUTO` and `iPIC3D` have already offloaded their most compute-intensive tasks.

3.1. Profiling activity

We used NVIDIA Nsight tools and POP3 CoE resources to profile and analyze code performance, starting with CPU-only optimizations informed by POP3 analysis [24, 15]. Our strategy focused on minimizing CPU-GPU data transfers, optimizing kernel occupancy, and improving work distribution to reduce register spilling. We used profiling tools to identify performance-critical sections, analyzed memory access for coalesced patterns, and reduced data transfer overhead by overlapping communication with computation using asynchronous streams. Further, we optimized Streaming Multiprocessor (SM) occupancy and workload balance by investigating warp efficiency and control flow divergence, and reduced synchronization overhead by minimizing global atomics and barriers. This comprehensive approach guided iterative optimizations, enabling us to address bottlenecks and achieve high performance on the Leonardo Booster system [25].

3.2. Porting of gPLUTO

The core of the gPLUTO algorithm involves five main steps, which are typically repeated as many times as the number of stages employed by the Runge-Kutta solver. These steps include: boundary exchange/calculation, mapping of the conservative vectors to primitive vectors, reconstruction (i.e. interpolation) of the cells interfaces values, solving Riemann problem to calculate the fluxes at the interfaces, computing the right hand side of the conservation law. The divergence-free condition is controlled through the choice of one among different (mutually exclusive) algorithms (e.g. constrained transport, divergence cleaning, etc...). The only function that involves CPUs (or GPUs) intercommunication is the boundary exchange; this one, as well as all the other offloaded functions, was profiled and optimized for GPU parallelization: functions have been fine-tuned for coalesced memory access using a custom Array class (as shown in [12]), ensuring that independent threads access consecutive and unique memory addresses in both read and write operations. Managed memory is used to simplify memory management between the CPU (host) and GPU (device). gPLUTO also handles boundary conditions and ghost cells with non-blocking MPI communications across domains, using asynchronous data movement across multiple GPUs. This setup enables fast communication with nearby tasks (8 neighbors in 2D, 26 in 3D) and fills ghost zones between processes. Memory is optimized by inlining frequently used functions to reduce jumps and load/store costs, while also helping the compiler access memory efficiently in GPU shared memory. Memory paths for multi-dimensional arrays are also determined at compile time via C++ templated functions.

3.3. Porting of OpenGadget3

In OpenGadget3, the N-body gravitational problem is solved using the Barnes & Hut algorithm, which employs a hierarchical oct-tree structure to compute gravitational forces. While the benefit is a relatively low complexity of $O(n \log n)$ [17], the method is memory-bound and poorly suited for GPU offloading due to irregular tree traversal, uncoalesced memory accesses, and frequent data transfers triggered by tree reconstruction at each simulation step. The recursive implementation of the tree build also introduces thread divergence and performance bottlenecks on GPUs, despite the Peano-Hilbert decomposition improving spatial and memory locality. Further inefficiencies arise because only a subset of particles is active at each step, requiring reordering and additional condition checks that degrade memory coalescence and warp efficiency. Previous GPU-friendly adaptations [26] have not fully addressed these limitations or captured the complexity of OpenGadget3 cosmological simulations. To address this, a new approach is under development that groups particles by spatial and memory locality (e.g., using a common center of mass), limits computations to nearby particles within small Hilbert curve segments, and uses direct summation within a fixed radius (Tornatore et al., in prep.). This reduces conditional branching, minimizes data movement, and improves warp occupancy. Additionally, non-essential data structure members were removed to reduce register spilling and enhance GPU thread performance. A more detailed explanation of this method will be provided in a forthcoming publication.

3.4. Porting of iPIC3D

The scalability of iPIC3D depends significantly on the number of particles employed, which determines the runtime of the Particle Mover (updating position and velocity of the particles at each cycle). The Moment Gatherer

module interpolates particle attributes to the grid (and vice versa) to compute charge and current densities and the pressure tensor. Both the Particle Mover and Moment Gatherer tend to be compute-bound for most input configurations. This makes them highly suitable for GPUs. The Field Solver, however, uses a generalized minimal residual algorithm (GMRes) to update the electromagnetic fields. GMRes is based on matrix-free Krylov subspace projections, which can easily saturate the memory on GPUs, thereby reducing the efficiency of the overall algorithm. GMRes also requires the computation of inner products of vectors, thereby necessitating multiple internode `MPI` communications. As this module is not as compute-bound as the other two, it runs exclusively on CPUs.

Following extensive optimization of the CPU-only version and the implementation of asynchronous non-blocking communication across the `MPI` processes, the Particle Mover and Moment Gatherer modules were offloaded to GPUs, while the field solver continues to run on CPU. This introduces an overhead of data communication between the GPU and CPU at every time step. However, this is an acceptable trade-off considering the speedups obtained in the computation of the GPU-offloaded modules.

4. Analysis and discussion of performance and scalability

Here, we describe our benchmarking efforts, tools, and methodology for evaluating the scalability and efficiency of these three parallel codes. A key principle in our approach is selecting problem sizes that fully utilize a single computational node, ensuring meaningful measurements of parallel performance. This strategy allows us to systematically assess how the full applications scale across multiple nodes while maintaining optimal resource utilization.

4.1. Automation of continuous integration and benchmarking tools

Continuous Integration and Continuous Deployment (CI/CD) are modern practices that automate the building, testing, and deployment of code. They streamline team collaboration, reduce human error, accelerate bug fixes, and enhance overall software quality. We teamed up with IT4I to set up CI/CD workflows for our three code-bases using their GitLab Runner system [27]. Regular benchmarking helps us spot performance issues and make sure our programs fully use modern hardware. It also catches slowdowns when we update code or add features, keeping performance steady over time [28]. To make benchmarking easier and work across different supercomputing systems, we use ReFrame [29], a Python tool for creating tests and benchmarks for HPC. ReFrame keeps system details separate from test designs, letting us write flexible tests that run smoothly on any system, speeding up both code checks and performance improvements.

4.2. Results for `gPLUTO`

Numerical benchmarks for `gPLUTO` have been carried out on both of Leonardo main computing partitions: DCGP and Booster. Two test cases were selected for this evaluation, the 3D Orszag-Tang vortex and the propagation of a 3D circularly polarized Alfvén wave. The scientific background and setup details for these tests are thoroughly discussed in [12]. The Orszag–Tang vortex involves complex shock interactions and the development of small-scale structures, making it ideal for assessing both numerical stability and performance under high-resolution conditions. On the other hand the circularly polarized Alfvén wave test provides an exact nonlinear solution to the ideal MHD equations and requires high accuracy and minimal numerical dissipation to preserve wave characteristics over long integration times, serving as a rigorous test of computational precision.

Weak scaling tests: The Orszag–Tang test simulations were conducted using a periodic 3D-Cartesian domain in double-precision arithmetic. The algorithm involved are the WENOZ reconstruction method, the HLLD Riemann solver, a third-order Runge-Kutta (RK3) time integration. To preserve the divergence-free condition of the magnetic field ($\nabla \cdot B = 0$) a constrained transport algorithm ([30], [31]) was used. A per-node resolution of $704 \times 704 \times 352$ was selected to optimize GPU memory usage, achieving a balance between high occupancy and minimal register spilling. Figure 3 left panel presents the parallel efficiency achieved on Leonardo Booster (up to 512 nodes) and Leonardo DCGP (up to 128 nodes) for this test, reaching approximately 88% and 90% efficiency, respectively. Table 1 compares absolute walltime values when running on both partitions the same weak scaling setup. Results indicate that, on average, executions on the DCGP partition are approximately an order of magnitude slower than those on the Booster partition, highlighting a significant performance gap between the two architectures.

Similar to the 3D Orszag-Tang test case, the circularly polarized Alfvén wave is a periodic 3D-Cartesian domain. The employed resolution is consistent with the previous Orszag-Tang test, and also in this case the computations are performed using double-precision arithmetic. Used algorithms are the same of the previous 3D Orszag-Tang test. As illustrated in the left panel of Figure 3, Leonardo Booster achieves an efficiency of 97% for 256 nodes in this case.

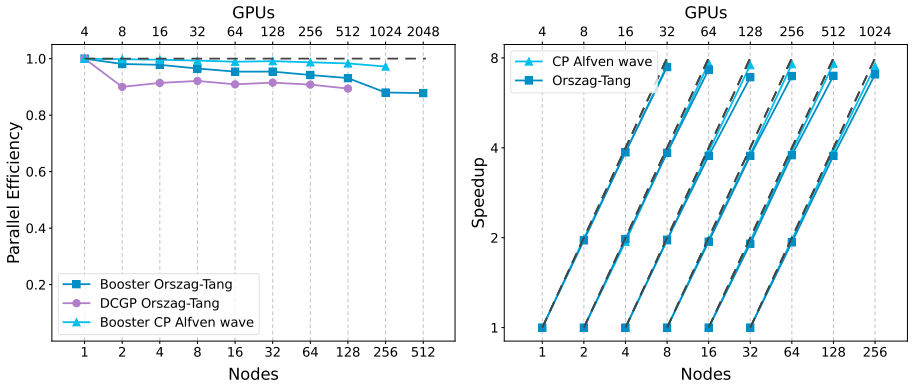


Fig. 3: Weak scaling tests for `gPLUTO` on both Leonardo Booster and DCGP partitions (left panel). Strong scaling tests for `gPLUTO` on Leonardo Booster (right panel). The black dashed lines indicate ideal scaling.

Nodes	T_{GPUs} (sec)	T_{CPUs} (sec)	Speedup
1	312	2982	9.55
2	318	3300	10.38
4	319	3263	10.23
8	323	3236	10.02
16	327	3281	10.03
32	327	3257	9.96
64	331	3283	9.92
128	335	3336	9.96

Table 1: CPU-GPU walltime comparison of `gPLUTO` 3D Orszag-Tang weak scaling test on Leonardo DCGP and Booster partitions.

Group	Nodes	Base Res	OT	CPA
1	1 to 8	$832^2 \times 416$	0.93	0.96
2	2 to 16	832^3	0.92	0.95
3	4 to 32	1664×832^2	0.86	0.95
4	8 to 64	$1664^2 \times 832$	0.87	0.96
5	16 to 128	1664^3	0.87	0.94
6	32 to 256	3328×1664^2	0.88	0.94
Mean	1 to 256	All res	0.90	0.95

Table 2: Strong scaling efficiencies for 3D Orszag-Tang (OT) and circularly polarized Alfvén (CPA) `gPLUTO` simulations performed on Leonardo Booster. Each group lists minimum obtained performance relative to its base resolution and node count.

Strong scaling tests: Strong scaling tests were performed with the same algorithm configurations of the weak scaling tests except for the technique to preserve the solenoidal condition of the magnetic field. This time the algorithm used was the divergence cleaning method ([32], [33]). This allowed for choosing a grid of $(832 \times 832 \times 416)$, slightly larger than that used in the weak scaling tests. This is because the divergence cleaning algorithm consumes less device memory than the constrained transport method. It is important to note that achieving ideal GPU strong scalability becomes increasingly challenging as the number of nodes grows. This difficulty arises due to overhead associated with inter-node communication, reduced computational workload per GPU, and increased synchronization costs, all of which can limit parallel efficiency at higher node counts. To address this, we conducted six distinct strong scaling campaigns for both Orszag-Tang and circularly polarized Alfén problems, as detailed in Table 2. Within each group, the computational resolution was held constant while the number of nodes and `MPI` processes was doubled three times. This approach ensures that the problem size is sufficiently large to fully utilize GPU memory at the higher number of nodes within each group, thereby maximizing hardware occupancy and minimizing inefficiencies. Strong scaling speedup values are presented in the right panel of Figure 3, demonstrating nearly 89% efficiency for the maximum number of nodes in each group for the Orszag-Tang problem, while an average efficiency of 95% is achieved for the circularly polarized Alfvén problem.

4.3. Results for OpenGadget3

Performance tests for OpenGadget3 were run on the Leonardo Booster and DCGP systems. Although DCGP was mainly used to test near-full-physics setups and for comparisons with the GPU versions, the main results come from the Leonardo Booster system. Since GPU support is still in progress, current results reflect only limited physics setups. Simulations model the evolution of the Universe using particles representing dark matter and gas within a 3D cosmological box. These particles interact through gravity, and gas particles also interact with hydrodynamic forces using SPH. Gravity and hydrodynamics each account for $\approx$ 40% of the runtime in a standard production run. Consequently, this setup is a good representation of the GPU performance of the entire code. High-resolution simulations require many particles and large volumes, which increase the computational demands of the short-range Barnes & Hut part. Simulation performance generally depends on redshift (the cosmic time). As matter is more homogeneously distributed at high redshifts, the calculation of gravitational forces via the gravity tree are faster ($\sim$ 10 $\times$ *speedup*), compared to lower redshifts ($\sim$ 2 $\times$ *speedup* at $z = 0$), when dense structures form, and the calculation becomes more complex and consequently more costly. So, performance varies with redshift and must be evaluated accordingly, resulting in an effective speedup of $\sim 3\times$.

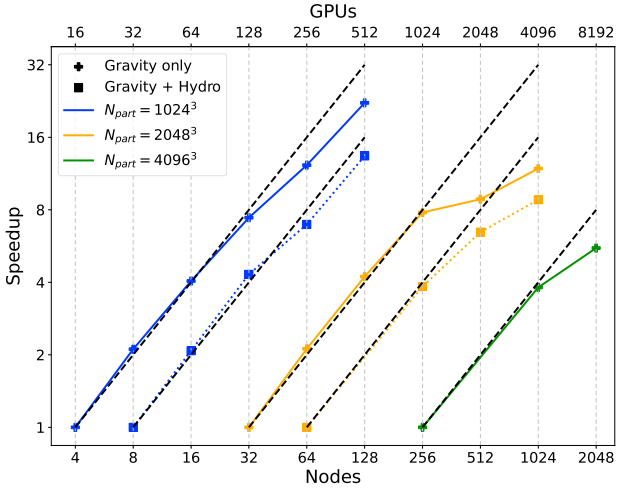


Fig. 4: Strong scaling tests for OpenGadget3 gravity-only and gravity + hydrodynamics on Leonardo Booster. The black dashed lines indicate ideal scaling.

We ran strong scaling tests at high redshift ($z \sim 50$), using three cosmological boxes with 1024^3, 2048^3, 4096^3 particles and box sizes of 120, 240, and 480 Mpc/h, respectively. This ensured the problem size was large enough per process to avoid communication issues at high node counts, which could affect scaling results. While using the same code configuration, we run these tests using initial conditions with and without gas particles. We used 4 MPI tasks per node (1 per GPU) and 8 OpenMP threads per task.

As shown in Figure 4, the 1024^3 case exhibits a smooth and consistent scaling behaviour, maintaining over 80% efficiency up to a 32$\times$ resource increase. The 2048^3 case shows an efficiency drop beyond a 16$\times$ scaling factor in both gravity and hydro setups, with domain decomposition issues beyond 512 nodes. While only executed in the gravity-only setup due to limited computational resources, the 4096^3 case demonstrates speed-up of up to a 4$\times$ scaling. In addition, a full simulation to $z = 0$ with 2048^3 particles in gravity-only mode was performed to compare the run-times between DCGP and Booster. The right panel of Figure 4 shows speedup versus cosmic expansion factor a ($=1/(1+z)$). Most computational time occurs at $a \gtrsim 0.2$, where efficiency drops due to the Barnes & Hut algorithm struggling with deeper tree structures from increased clustering at lower redshifts. Smaller time steps are needed early on due to stronger gravitational accelerations, but performance gains are measured at larger a, yielding a 2-3 $\times$ speed-up. Despite being relatively low, this speedup is significant for simulations requiring 10^7 or more core-hours. Since short-range gravitational force calculations cause most performance issues, this supports our approach of optimizing the tree traversals, as discussed in Section 3.

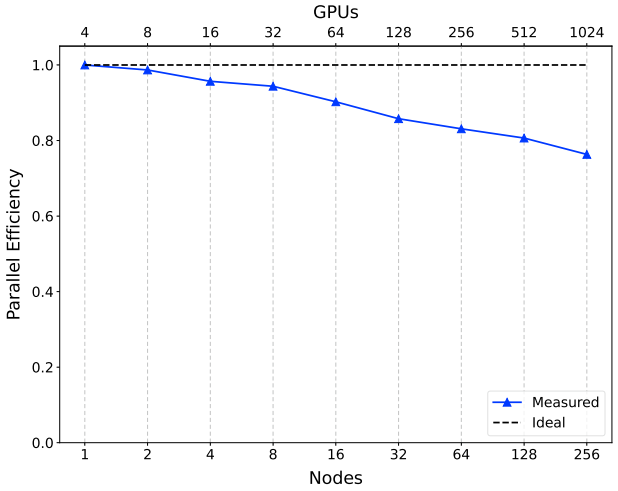


Fig. 5: Weak scaling test of `iPIC3D-GPU` on Leonardo Booster. The black dashed line indicates ideal scaling.

4.4. Results for *iPIC3D*

We show a performance comparison of the CPU and GPU versions of `iPIC3D` for the case of a Maxwellian distribution in 2D with $20 \times 20 \times 20$ particles per cell and 4 particle species on Leonardo DCGP and Booster, respectively (Table 3). We observe that the Moment Gatherer obtains a speedup of 100 whereas the Particle Mover achieves a speedup of a factor of 40, which determines the overall speedup of the code. As the Field Solver runs exclusively on CPUs, no speedup is achieved for this module. Furthermore, we illustrate the weak scaling of `iPIC3D-GPU` on

Module	`iPIC3D-GPU` (Leonardo Booster)	`iPIC3D-CPU` (Leonardo DCGP)	**Speedup**
Particle Mover	0.542 s	21.891 s	40.4
Moment Gatherer	0.123 s	12.271 s	99.8
Field Solver	0.185 s	0.183 s	0.98
Total	**0.870 s**	**35.007 s**	**40.2**

Table 3: Comparison of the CPU and GPU version of `iPIC3D` for a 2D Maxwellian distribution test case with $20 \times 20 \times 20$ particles per cell and 4 total species.

Leonardo Booster, the results of which are reported in Fig. 5. We consider the same Maxwellian distribution in 2D, where the number of grid cells was progressively increased from 128^2 to 2048^2. Each simulation modeled 4 particle species with 180×180 particles per cell per species. The code demonstrated an efficiency of 78% up to 1024 GPUs, corresponding to 256 nodes on the Leonardo Booster.

5. Summary

This paper presents the strategy and early achievements of the EuroHPC SPACE Center of Excellence (SPACE-CoE) in adapting three flagship astrophysical simulation codes i.e. `gPLUTO`, `OpenGadget3`, and `iPIC3D` , for exascale computing on the Leonardo supercomputer at CINECA. Through collaborative efforts, key modules were successfully offloaded to GPUs, enabling a transition from CPU to GPU architectures using profiling and optimization techniques. Preliminary results on the EuroHPC Leonardo system show notable scalability up to 1,024 GPUs with efficiencies around 80-97%. The study highlights `gPLUTO`'s near-ideal weak and strong scaling, `OpenGadget3`'s bottlenecks with the Barnes & Hut algorithm mitigated through restructuring, and `iPIC3D` 's substantial GPU acceleration, achieving up to 100× speedups in certain modules and 78% weak scaling efficiency by leveraging selective GPU offloading and asynchronous CPU-GPU communication. By combining performance profiling, code modularization, continuous integration, and GPU-specific kernel optimization, the project demonstrates how interdisciplinary collaboration can modernize complex simulation tools to fully exploit emerging exascale infrastructure.

Acknowledgements

This work was supported by the SPACE CoE, funded by the EU and several partner countries under grant No. 101093441. We also thank ISCRA and the EuroHPC Joint Undertaking for access to the computational resources on Leonardo supercomputer at CINECA (Italy). This work was partially funded by the POP3 project (grant No. 101143931), supported by the EuroHPC Joint Undertaking, its member countries, and additional funding from the Czech Ministry of Education (ID: MC2401) and e-INFRA CZ (ID: 90254). We also would like to acknowledge the NVIDIA experts for their invaluable assistance in porting the code to GPUs during and after dedicated hackathon events, especially Matt Bettencourt and Filippo Spiga.

References

[1] S. Sarkar, Is dark matter self-interacting?, Nat Astron 2 (2018) 856–857. doi:10.1038/s41550-018-0598-6.
[2] K. Schoeffler, et al, Can plasma physics establish a significant bound on long-range dark matter interactions?, Phys. Rev. D 111 (2025) L071701. doi:10.1103/PhysRevD.111.L071701.
[3] A. Romeo, et al, Simulations for 21 cm radiation lensing at eor redshifts, Mon. Not. R. Astron. Soc. 474 (2) (2017) 1787–1809. doi:10.1093/mnras/stx2733.
[4] R. Croft, et al, Weak lensing of the lyman α forest, Mon. Not. R. Astron. Soc. 477 (2) (2018) 1814–1821.
[5] S. Markidis, et al, Collisionless magnetic reconnection in a plasmoid chain, Nonlin. Processes Geophys. 19 (2012) 145–153. doi:10.5194/npg-19-145-2012.
[6] G. Mattia, et al, Resistive relativistic mhd simulations of astrophysical jets, Astronomy & Astrophysics 679 (2023) A49. doi:10.1051/0004-6361/202347126.
[7] S. Ferro, et al, Comparative simulations of kelvin–helmholtz induced magnetic reconnection at the earth's magnetospheric flanks, Phys. Plasmas 31 (5) (2024) 052902. doi:10.1063/5.0191674.
[8] S. Borgani, et al, X-ray properties of galaxy clusters and groups from a cosmological hydrodynamical simulation, Mon. Not. R. Astron. Soc. 348 (3) (2004) 1078–1096. doi:10.1111/j.1365-2966.2004.07431.x.
[9] N. Shukla, et al, Conditions for the onset of the current filamentation instability in the laboratory, J. Plasma Phys. 84 (3) (2018) 905840302. doi:10.1017/S0022377818000314.
[10] E. Boella, et al, Interaction between electrostatic collisionless shocks generates strong magnetic fields, New J. Phys. 24 (2022) 063016. doi:10.1088/1367-2630/ac6ef1.
[11] N. Shukla, et al, Eurohpc space coe: Redesigning scalable parallel astrophysical codes for exascale, in: 22nd ACM Int. Conf. Comput. Front, ACM, New York, NY, USA, Cagliari, Italy, 2025, p. 7. doi:10.1145/3706594.3728892.
[12] M. Rossaza, Gpu porting of the pluto code for computational plasma physics using openacc, Ph.D. thesis, University of Turin (2025). URL https://iris.unito.it/handle/2318/2065332
[13] F. Groth, et al, The cosmological simulation code OPENGADGET3 - implementation of meshless finite mass, Mon. Not. R. Astron. Soc. 526 (1) (2023) 616–644. arXiv:2301.03612, doi:10.1093/mnras/stad2717.
[14] S. Markidis, et al, Multi-scale simulations of plasma with iPIC3D, Math. Comput. Simul. 80 (7) (2010) 1509–1519. doi:10.1016/j.matcom.2009.08.038.
[15] POP3: Performance Optimization and Productivity Centre of Excellence, https://www.pop-coe.eu/, european Centre of Excellence on HPC performance optimization, project duration 2024–2027, coordinated by Barcelona Supercomputing Center (2024).
[16] V. Springel, The cosmological simulation code GADGET-2, Monthly Notices of the Royal Astronomical Society 364 (4) (2005) 1105–1134. arXiv:astro-ph/0505010, doi:10.1111/j.1365-2966.2005.09655.x.
[17] J. Barnes, P. Hut, A hierarchical O(N log N) force-calculation algorithm, Nature 324 (6096) (1986) 446–449. doi:10.1038/324446a0.
[18] M. Frigo, S. G. Johnson, The design and implementation of FFTW3, Proceedings of the IEEE 93 (2) (2005) 216–231, special issue on "Program Generation, Optimization, and Platform Adaptation".
[19] A. M. Beck, et al, An improved SPH scheme for cosmological simulations, Mon. Not. R. Astron. Soc. 455 (2) (2016) 2110–2130. arXiv:1502.07358, doi:10.1093/mnras/stv2443.
[20] A. Ragagnin, N. Tchipev, M. Bader, K. Dolag, N. J. Hammer, Exploiting the Space Filling Curve Ordering of Particles in the Neighbour Search of Gadget3, in: Advances in Parallel Computing, 2016, pp. 411–420. doi:10.3233/978-1-61499-621-7-411.
[21] A. Ragagnin, K. Dolag, M. Wagner, C. Gheller, C. Roffler, D. Goz, D. Hubber, A. Arth, Gadget3 on GPUs with OpenACC, Parallel Computing: Technology Trends 27 (2020) 209–218. doi:10.3233/APC200043.
[22] M. Turisini, et al, Leonardo: A pan-european pre-exascale supercomputer for hpc and ai applications, J. Large-Scale Res. Facilities 9 (01 2024). doi:10.17815/jlsrf-8-186.
[23] T. Gamblin, et al, The spack package manager: Bringing order to hpc software chaos, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC15), ACM, 2015, pp. 1–12. doi:10.1145/2807591.2807623.
[24] Barcelona Supercomputing Center, Extrae: The Extrae Tracing Tool, Barcelona Supercomputing Center, version 4.0, Available at https://www.ebrains.eu/tools/extrae (2025).
[25] L. Tornatore, et al., Code Modules and Kernels, SPACE-COE project document (2024).

[26] M. Burtscher, K. Pingali, Chapter 6 - an efficient cuda implementation of the tree-based barnes hut n-body algorithm, in: GPU Computing Gems Emerald Edition, Applications of GPU Computing Series, Morgan Kaufmann, Boston, 2011, pp. 75–92. doi:https://doi.org/10.1016/B978-0-12-384988-5.00006-1.

[27] B. Commercon, et al., Code release (alpha), SPACE-COE project document (2024).

[28] S. Williams, et al, Roofline: An insightful visual performance model for multicore architectures, Commun. ACM 52 (4) (2009) 65–76. doi:10.1145/1498765.1498785.

[29] V. Karakasis, et al, Enabling continuous testing of hpc systems using reframe, in: Tools and Techniques for High Performance Computing, Springer, Cham, 2020, pp. 49–68.

[30] C. Evans, et al, Simulation of Magnetohydrodynamic Flows: A Constrained Transport Model, Astrophys. J. 332 (1988) 659. doi:10.1086/166684.

[31] P. Londrillo, L. D. Zanna, On the divergence-free condition in godunov-type schemes for ideal magnetohydrodynamics: the upwind constrained transport method, J. Comput. Phys. 195 (1) (2004) 17–48. doi:10.1016/j.jcp.2003.09.016.

[32] A. Dedner, et al, Hyperbolic Divergence Cleaning for the MHD Equations, J. Comput. Phys. 175 (2002) 645–673. doi:10.1006/jcph.2001.6961.

[33] A. Mignone, P. Tzeferacos, A second-order unsplit Godunov scheme for cell-centered MHD: The CTU-GLM scheme, J. Comput. Phys. 229 (2010) 2117–2138. arXiv:0911.3410, doi:10.1016/j.jcp.2009.11.026.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 124–135

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Tailoring AI for Turkish Law: Domain-Specific Fine-Tuning of Small Language Models for Legal Expertise

New Mind AI Team[a,*]

[a]*New Mind AI, YTU Teknopark, Sariyer, Türkiye*

Abstract

The development of specialized Artificial Intelligence (AI) for niche legal domains, such as Turkish law, is hampered by the cost and generality of large language models (LLMs). This study investigates the efficacy and efficiency of fine-tuning a moderately sized SLM (Small Language Model), Meta-Llama 3.1 8B, to create expert models for ten distinct Turkish legal sub-domains (Energy, Land, Competition, Tax, Healthcare, Labor, Intellectual Property, Data Protection (KVKK), Capital Markets, Environmental Law). Leveraging the EuroHPC Karolina supercomputing infrastructure (IT4Innovations), we employed Parameter-Efficient Fine-Tuning (PEFT) techniques (LoRA/QLoRA) orchestrated via Axolotl and significantly accelerated using the Unsloth library. Models were trained on curated domain-specific datasets. Performance was rigorously evaluated against the base 8B model using the LightEval framework, incorporating lexical, semantic, and AI-judge metrics. Results demonstrate that the fine-tuned models achieve substantial performance improvements in domain-specific understanding and task handling compared to the base model. This study's evaluation is scoped to comparisons against the 8 billion parameter base model to demonstrate the effectiveness of domain-specific fine-tuning. A direct performance benchmark against very large proprietary models like GPT-4 or Claude 3 was not conducted due to data privacy constraints preventing the use of our specialized datasets with external APIs. Our findings validate that fine-tuning smaller, accessible models like Llama 3.1 8B offers a computationally efficient, economically viable, and effective pathway for developing specialized legal AI expertise, presenting a practical alternative to reliance on larger, general-purpose models, particularly for resource-constrained or privacy-sensitive applications. This work provides both a methodological demonstration on HPC resources and tangible assets for the Turkish legal tech sector.

Keywords: Turkish Legal Domain; Small Language Models; Domain Adaptation; Parameter-Efficient Fine-Tuning (PEFT); Legal Tech; Natural Language Processing (NLP)

∗ Corresponding author. Tel.: +90-212-777-8890
E-mail address: aiteam@newmind.ai

1877-0509
10.1016/j.procs.2025.08.239

1. Introduction

Large Language Models (LLMs) are increasingly reshaping professional domains, with significant transformative potential observed in the legal sector [1]. These models offer powerful capabilities for processing and generating human language, presenting opportunities to automate or augment tasks such as legal research, document review, contract analysis, and summarization [2]. However, the deployment of state-of-the-art LLMs, particularly the largest proprietary ones, faces substantial challenges. These include high computational costs for training and inference, significant energy consumption, and, critically, a frequent lack of the deep, nuanced understanding required for highly specialized domains [3].

This limitation is particularly pronounced within specific legal jurisdictions like Türkiye, whose distinct regulatory frameworks, terminology, and linguistic characteristics are often inadequately represented in general-purpose models trained predominantly on global, English-centric data [4]. Consequently, a critical need exists for AI tools precisely tailored to the intricacies of the Turkish legal system to enhance efficiency, accuracy, and accessibility for legal professionals [5]. Addressing these challenges, a promising direction involves leveraging smaller, more focused LLMs [6]. Research indicates that models with fewer parameters (e.g., 10-20 billion), when expertly fine-tuned on high-quality, domain-specific data, can achieve performance comparable, or even superior, to vastly larger models on targeted tasks, while being significantly more computationally efficient and economically viable to deploy [7, 8]. This efficiency makes them particularly well-suited for integration into enterprise agentic workflows, where multiple specialized AI agents can collaborate cost-effectively on complex legal processes [9]. Given this focus, the scope of our evaluation is centered on quantifying the performance uplift gained through domain specialization, comparing our fine-tuned models directly against their 8B-parameter base model. While we acknowledge the capabilities of very large proprietary models such as GPT-4 and Claude 3, direct comparative benchmarking was intentionally excluded. This decision was driven by data privacy protocols, as our sensitive, domain-specific Turkish legal datasets cannot be processed via third-party APIs.

Despite the potential of smaller, fine-tuned models, a significant gap exists: the lack of accessible, high-performance LLMs specifically tailored for the diverse and intricate sub-domains within Turkish law. Legal professionals currently lack specialized AI tools capable of accurately handling the nuances of areas such as Turkish Energy Law, Competition Law, or Data Protection Law (KVKK), thereby hindering the effective adoption of advanced AI technologies within the Turkish legal sector.

The primary objective of this research is to develop and rigorously evaluate specialized LLMs for ten distinct Turkish legal sub-domains. This is achieved by fine-tuning a readily available, moderately sized base model, Meta-Llama 3.1 8B Instruct, leveraging the high-performance computing resources of EuroHPC.

Our central hypothesis is that targeted, parameter-efficient fine-tuning (PEFT) of this 8-billion parameter LLM on synthetically generated, curated, and domain-specific datasets can produce a suite of specialized 'expert' models. We posit that these fine-tuned models will exhibit significantly enhanced performance on domain-specific tasks compared to the base model, reaching a practical level of proficiency and representing an efficient, economically viable alternative to relying solely on larger, general-purpose LLMs for these specialized applications.

The key contributions of this work are:

- The development of ten fine-tuned LLMs tailored for distinct Turkish legal domains: Energy, Land, Competition, Tax, Healthcare, Labor, Intellectual Property, KVKK, Capital Markets, and Environmental Law.
- A practical demonstration of efficient fine-tuning methodologies (PEFT using LoRA/QLORA, optimization with Unsloth, orchestration via Axolotl) executed on EuroHPC's Karolina supercomputing infrastructure.
- A comprehensive quantitative evaluation comparing the performance of the fine-tuned models against the base model using a combination of lexical, semantic, and AI-judge metrics facilitated by the LightEval framework.
- Empirical evidence supporting the feasibility and effectiveness of fine-tuning smaller LLMs to create high-performing, domain-specific expert models suitable for deployment in specialized enterprise settings, particularly agentic workflows.

The remainder of this paper is organized as follows: Section 2 reviews related work concerning LLMs in law, domain adaptation techniques, the comparison of small versus large models, and AI for Turkish language

processing. Section 3 details the methodology, covering the base model, dataset creation, fine-tuning procedure, computational environment, and evaluation strategy. Section 4 presents the quantitative results and analysis. Section 5 discusses the interpretation of these findings, challenges encountered, and potential future research directions. Finally, Section 6 provides concluding remarks summarizing the study's achievements and impact.

2. Related Work

The research presented in this paper builds upon several intersecting fields within NLP and AI, particularly concerning language models, domain adaptation, and their application in specialized fields like law. This section reviews relevant literature in these areas.

Large Language Models in Law. The advent of LLMs such as GPT-4 [10], Claude [11], and PaLM [12] has significantly impacted various professional domains, including the legal sector. These models have demonstrated capabilities in tasks like legal research [13], document summarization [14], contract review and analysis [15], legal question answering [16], and even preliminary legal drafting [17]. Specialized legal AI platforms, often leveraging these foundational models (e.g., mecellem.com, Harvey AI, CoCounsel by Casetext/Thomson Reuters), aim to enhance lawyer productivity and streamline workflows [18]. However, the deployment of these large, general-purpose LLMs in legal practice faces several limitations. Firstly, their substantial computational requirements necessitate significant resources for both training and inference, leading to high operational costs [19]. Secondly, while broadly capable, their knowledge might lack the depth and nuance required for highly specialized legal sub-domains or specific jurisdictions like Turkey [20]. Concerns regarding data privacy, security, and the potential for generating inaccurate or "hallucinated" information remain significant hurdles for critical legal applications [21]. These limitations motivate the exploration of more targeted and efficient approaches.

Domain Adaptation and Fine-Tuning. Domain adaptation techniques are widely employed to overcome the generality of large pre-trained models and enhance their performance on specific tasks or domains. Full fine-tuning, which involves updating all parameters of a pre-trained model on a domain-specific dataset, can yield high performance but is computationally expensive and requires large datasets to avoid overfitting or catastrophic forgetting [22]. PEFT methods have emerged as a compelling alternative, aiming to adapt LLMs with significantly fewer trainable parameters [23]. To adapt the base model efficiently, we employed Parameter-Efficient Fine-Tuning (PEFT) methods, specifically LoRA [24] and its quantized variant QLoRA [25], which adapt models by training a small number of additional parameters while keeping the base model frozen. These PEFT methods are particularly relevant for creating specialized models efficiently, aligning well with the objectives of this study. Another relevant technique in legal AI is RAG [26]. RAG enhances LLM outputs by first retrieving relevant documents or passages from a knowledge base (e.g., legal statutes, case law) and then providing this context to the LLM alongside the user query. While RAG helps ground responses in factual data and mitigate hallucinations, it primarily addresses knowledge access at inference time, whereas fine-tuning instills domain-specific knowledge and behavior directly into the model's parameters [27]. Fine-tuning and RAG can also be used complementarily.

Small vs. Large Language Models. While the trend in recent years has been towards increasingly larger LLMs [10, 12], a growing body of research highlights the effectiveness and efficiency of smaller language models (SLMs), typically defined as models with fewer than 10-20 billion parameters [28]. Studies have shown that SLMs, when properly fine-tuned on high-quality, domain-specific data, can achieve performance comparable or sometimes superior to much larger models on specific tasks [29, 30]. The advantages of SLMs are significant, particularly for enterprise deployment. They require substantially fewer computational resources for both fine-tuning and inference, resulting in lower costs and reduced energy consumption [31]. Their smaller size facilitates deployment in resource-constrained environments, potentially on-premises or even on-edge devices, enhancing data privacy and control. This makes SLMs a practical choice for building specialized "expert" models tailored for specific business units or workflows, such as the agentic systems targeted in this work [32]. Our research investigates whether an 8-billion parameter model (Meta-Llama 3.1 8B) can be effectively specialized for complex Turkish legal domains.

AI for Turkish Language Processing. Turkish presents unique challenges for NLP due to its agglutinative morphology and complex syntactic structures [33]. While significant progress has been made in Turkish NLP, including the development of Turkish BERT models [34] and some general-purpose Turkish LLMs [35], resources

and research specifically targeting the Turkish legal domain remain relatively scarce compared to English. Existing work might focus on text classification for legal documents or basic information extraction [36, 37]. However, the development and public availability of high-performing generative LLMs specifically trained or fine-tuned for diverse Turkish legal sub-domains are limited. The lack of large, publicly available, high-quality annotated datasets for specialized Turkish legal tasks further compounds this challenge [38]. This research aims to contribute to closing this gap by creating and evaluating specialized models for ten distinct Turkish legal areas, leveraging recent advances in LLMs and PEFT techniques.

3. Methodology

Base Model. The foundation for our specialized models is Meta-Llama 3.1 8B Instruct [39]. This model was selected for its strong baseline performance as a state-of-the-art open-weight model, its manageable 8-billion parameter size offering a balance between capability and computational tractability suitable for efficient fine-tuning and economical deployment (qualifying it as an SLM), its permissive license allowing research and development, and its instruction-tuned nature which makes it readily adaptable to specific downstream tasks via structured prompting.

Datasets and Domains. To develop specialized expertise, we targeted ten distinct sub-domains within Turkish law: Energy & Natural Resources, Land & Zoning, Competition, Tax, Healthcare, Labor, Intellectual Property (IP), Data Protection (KVKK), Capital Markets, and Environmental Law. Constructing effective training datasets for each domain necessitated a hybrid approach. We began by curating publicly available legal texts, regulations, and jurisprudence, forming a foundational corpus. However, to ensure sufficient high-quality examples for instruction-following, we significantly supplemented this core data with synthetically generated content.

This synthetic data generation relied on a highly efficient, multi-layered LLM pipeline architected for scalability and quality control (specific models detailed in Table 1), a visual overview of which is provided in Figure 1. The process commenced with a Generator stage where multiple LLMs (e.g., Llama 3.1 70B, Qwen2.5-72B) operated in parallel on domain documents to produce diverse initial question candidates. These candidates were then rigorously evaluated in a Critic stage, employing another LLM augmented with Retrieval-Augmented Generation (RAG) drawing from relevant legal sources. This stage systematically assessed questions against criteria like relevance, clarity, and legal grounding. Finally, a deterministic Fuser model processed the critique, synthesizing the positive aspects of candidate questions while eliminating or rephrasing problematic elements to yield highly refined, optimized final questions.

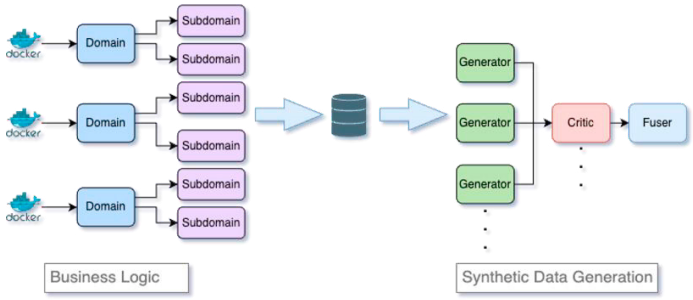


Fig. 1. High-level overview of the synthetic data generation pipeline, from domain processing to the Generator-Critic-Fuser architecture for producing high-quality question-answer pairs.

Table 1. Synthetic data generation pipeline models configuration.

Type	Model	Top_k	Temperature	Max Tokens	Samples
generator	meta-llama/Llama-3.3-70B-Instruct	1	0.3	3000	1
generator	Qwen/Qwen2.5-72B-Instruct	1	0.3	5000	1
critic	Qwen/Qwen2.5-72B-Instruct	1	0.3	16384	1
fuser	Qwen/Qwen2.5-72B-Instruct	1	0.0	4096	1

Following the question generation and refinement, the same RAG-enhanced architectural framework was leveraged to generate high-quality answers corresponding to the finalized questions. This sequential process ensured consistency and contextual grounding. The entire pipeline was designed for parallel execution across domains (supporting up to 100 concurrent workflows), enabling the efficient production of approximately 10,000 high-quality question-answer pairs per domain.

To ensure the integrity of the training data, we embedded several quality control (QC) mechanisms throughout this pipeline. The QC process included multiple automated and manual steps. Initially, all generated questions underwent automated filtering based on a set of heuristic rules, such as character length constraints (15-250 characters), removal of duplicates, and syntactic checks to ensure proper question formatting. The 'Critic' stage served as the next layer of automated QC, where an LLM systematically discarded questions flagged for irrelevance, lack of clarity, or poor legal grounding. Following automated filtering, a random subset (approximately 5%) of the synthetically generated data from each domain was subjected to manual spot-checks by our team to validate its coherence and domain accuracy. This iterative human-in-the-loop validation provided a crucial final quality gate before the data was accepted into the training set.

The total combined training dataset size across all domains comprised 149,571 text examples. Detailed statistics regarding average token length, minimum/maximum lengths, and data size per domain category can be found in Table 2. To structure the data for fine-tuning, we employed two widely used instruction formats: the Alpaca format [40] and the ChatML template [41]. This allowed us to train two variants for each domain and compare the impact of the input structure.

Table 2. Summary statistics by category.

Category	Avg. Token	Min	Max	Data Size
Capital	364.66	197	831	10,495
Environmental	365.76	200	1623	15,070
Fund	415.43	219	2935	9,155
Intellectual	389.97	200	3407	11,503
Land	382.21	199	1230	13,348
Competition Law	292.76	72	1967	18,000
Healthcare Law	295.63	72	4424	18,000
Tax Law	284.34	78	918	18,000
Data Protection Law	295.6	85	2778	18,000
Energy	305.32	90	2751	18,000

The selection of ten distinct legal sub-domains was based on established areas of legal specialization within the Turkish professional landscape. Although we acknowledge that overlaps exist, for example, between healthcare and labor law or environmental and energy law, each domain possesses a unique corpus of legislation, specific terminology, and distinct regulatory frameworks. Our hypothesis is that creating highly specialized "expert" models requires this level of granularity. By training each model on a focused dataset, we aim to instill deep, nuanced knowledge that might be diluted in a more generalized model. This approach allows for a direct evaluation of specialization, with the potential for future work to explore knowledge transfer and domain fusion.

Fine-tuning Procedure. The fine-tuning process was orchestrated using the Axolotl framework [42], selected for its flexibility in managing diverse models, PEFT methods, and convenient YAML-based configuration system. Significant acceleration and resource optimization were achieved through integration with the Unsloth library [43], which provides highly optimized kernels, efficient memory management strategies, and leverages formats like BF16, drastically reducing training times compared to standard Hugging Face implementations.

Our primary PEFT method was Low-Rank Adaptation (LoRA). The specific LoRA configuration included a rank (r) of 16 and a LoRA alpha (alpha) of 32, effectively controlling the dimensionality and scaling of the adaptation matrices. A dropout probability of 0.05 (lora_dropout) was applied to the LoRA layers to mitigate overfitting. We targeted a comprehensive set of modules for LoRA adaptation, including the query, key, value, and output projections in the attention layers (q_proj, k_proj, v_proj, o_proj), as well as the gate, up, and down projections in the feed-forward layers (gate_proj, up_proj, down_proj). LoRA biases were configured to be trainable (bias: all), and the eva method was used for initializing LoRA weights. Residual Scaled LoRA (use_rslora) was not employed in this configuration (use_rslora: false).

The training hyperparameters were set as follows: we used a learning rate of $3e^{-5}$ with a cosine learning rate scheduler (lr_scheduler type: cosine) and applied weight decay of 0.01. Training was conducted for 3 epochs (num_train_epochs: 3) for each domain. We utilized a per-device batch size of 4 (batch_size: 4) and gradient accumulation steps of 4 (gradient_accumulation_steps: 4), resulting in an effective batch size of 16. Gradient norms were clipped to a maximum value of 1.0 (max_grad_norm: 1). Data packing (packing: false) was disabled. To further conserve memory, especially given the model size and comprehensive LoRA target modules, gradient checkpointing (use_gradient_checkpointing: true) was enabled. While we experimented with quantized methods like QLoRA (e.g., using 4-bit NormalFloat quantization) during development, the results reported here correspond to the specified LoRA configuration trained using BF16 precision facilitated by Unsloth. The overall training duration varied per domain based on dataset size but was substantially reduced by these optimizations.

Computational Environment. All fine-tuning experiments were executed on the Karolina supercomputer at IT4Innovations National Supercomputing Center, accessed via a EuroHPC Joint Undertaking Benchmark Access grant. We utilized Karolina's GPU nodes, each equipped with 8x NVIDIA A100 GPUs. Our software stack was built on PyTorch (2.4.1+cu121) and CUDA (12.1), employing Hugging Face Transformers (4.48.0) for model handling, Axolotl for orchestration, and Unsloth (2025.1.6) along with libraries like Flash Attention and xFormers for optimized training kernels and memory management. Job submission and resource management were handled via Slurm, with Python (3.11.2) as the primary programming language and OpenMPI (4.1.6) available. Training monitoring and visualization were performed using Weights & Biases (0.19.4), which confirmed high resource efficiency with typical GPU and memory utilization between 85-90%.

Evaluation Strategy. Model performance was evaluated using the LightEval framework (version 0.7.0) [44], a lightweight library for LLM evaluation. Our evaluation strategy was designed to provide a multi-faceted assessment, comparing each fine-tuned model (both Alpaca and ChatML variants) against the original Meta-Llama 3.1 8B Instruct base model. We employed three categories of metrics:

- Lexical Metrics: Assessing surface-level similarity and overlap between generated text and reference answers using standard NLP metrics: BLEU (Bilingual Evaluation Understudy) [45], ROUGE (Recall-Oriented Understudy for Gisting Evaluation), specifically ROUGE-L [46], F1-Score (calculated based on token overlap). These were averaged to provide a Lexical Avg. score as in Table 3.
- Semantic Metrics: Measuring the similarity in meaning between generated outputs and reference answers using sentence embeddings. We utilized pre-trained Sentence Transformers models, specifically all-MiniLM-L6-v2 [47] for general English/multilingual capability and emrecan/bert-base-turkish-cased-mean-nli-stsb-tr [48] potentially for Turkish-specific nuances. Cosine similarity between embeddings was calculated and averaged as shown in Table 3.
- AI Judge Metrics: Employing a capable LLM (gpt-4o-mini) as an evaluator to assess the quality of model outputs based on criteria crucial for legal applications: Relevancy: Is the answer pertinent to the legal question asked? Coherence: Is the answer logically structured and easy to understand? Accuracy: Does the answer reflect correct legal information or reasoning (within the scope of the provided context or the model's trained knowledge)? Scores for these aspects were averaged to produce an OpenAI Judge Avg. as given in Table 3.

The evaluation process involved processing a substantial number of tokens across various tasks, exceeding 242 million tokens in total for the LightEval benchmarks considered as shown in Appendix A, Table A.5, ensuring a robust assessment.

4. Results

This section presents a detailed analysis of the model performance, focusing on quantitative metrics, training efficiency, and qualitative observations.

4.1. Quantitative Performance

The core performance evaluation compared the twenty fine-tuned models (ten legal domains, each in Alpaca and Chat-template formats) against the baseline Meta-Llama 3.1 8B Instruct model. We utilized the LightEval framework and aggregated results across lexical, semantic, and AI-judge metrics as described in Methodology.

Table 3 summarizes the key evaluation results. It presents the average scores for each model across the three metric categories: Lexical Average, Semantic Average, and OpenAI Judge Average. Detailed performance scores for each fine-tuned model variant and the base model across the individual benchmark tasks (Arc, Hellaswag, TruthfulQA, Winogrande) evaluated using LightEval are provided in Appendix A, Table A.4. Furthermore, while a full benchmark against open-weight 70B models was beyond the current project's scope, we provide their scores on the same general evaluation benchmarks for context.

Table 3. Comprehensive evaluation results for fine-tuned models vs. base model.

Model Name	Lexical Avg.	Semantic Avg.	OpenAI Judge Avg.
Base Model (Meta-Llama 3.1 8B Instruct)	0.379	0.842	0.594
Alpaca-capital	0.411	0.928	0.368
Chat-template-capital	0.443	0.942	0.545
Alpaca-competition	0.430	0.937	0.495
Chat-template-competition	0.344	0.920	0.367
Alpaca-energy	0.425	0.940	0.502
Chat-template-energy	0.335	0.918	0.368
Alpaca-environmental	0.412	0.917	0.428
Chat-template-environmental	0.426	0.895	0.490
Alpaca-fund	0.441	0.920	0.354
Chat-template-fund	0.462	0.935	0.487
Alpaca-health	0.395	0.924	0.425
Chat-template-health	0.337	0.921	0.399
Alpaca-intellectual	0.398	0.894	0.313
Chat-template-intellectual	0.464	0.943	0.593
Alpaca-kvkk	0.414	0.938	0.477
Chat-template-kvkk	0.334	0.919	0.389
Alpaca-land	0.391	0.902	0.325
Chat-template-land	0.436	0.934	0.532
Alpaca-tax	0.400	0.928	0.456
Chat-template-tax	0.332	0.919	0.373

Semantic Performance Analysis. A primary indicator of successful domain adaptation is the model's ability to understand the meaning and context of specialized text. Our results show a clear and significant improvement in semantic understanding across all fine-tuned models. The base model achieved a Semantic Average score of 0.842. In contrast, every fine-tuned model surpassed this, with scores consistently ranging from 0.894 to 0.943. This

uniform uplift indicates that the models successfully assimilated the nuanced terminology and complex concepts of their respective legal domains. High semantic similarity is crucial in the legal field, where understanding the intent and meaning of a query is more important than generating lexically identical text to a reference answer. This result strongly supports our hypothesis that targeted fine-tuning instills deep domain-specific knowledge.

Lexical Performance Analysis. Lexical metrics (e.g., BLEU, ROUGE-L) measure the surface-level overlap between the model's output and the reference answers. While the base model scored 0.379, the fine-tuned models showed varied but generally positive results, with scores ranging from 0.332 to 0.464. For instance, the Chat-template-intellectual model achieved the highest lexical score (0.464). These metrics suggest an improved ability to replicate the structure and phrasing of the training data. However, lexical similarity is a secondary objective in this context, as legally valid answers can be formulated in many different ways. The moderate gains, combined with the strong semantic results, suggest the models learned to articulate concepts from the training data without merely memorizing them.

AI-Judge Performance Analysis. The AI-Judge metrics (evaluating relevancy, coherence, and accuracy) provide the most practical assessment of the models' utility. Here, the results are more nuanced and reveal key insights into the specialization process. The base model, a generalist, scored a high 0.594. While none of our specialized models surpassed this general score, several came remarkably close within their specific domain of expertise.

The Chat-template-intellectual model, for example, achieved a score of 0.593, demonstrating near-human-level performance on specialized Intellectual Property law tasks. Similarly, the models for Chat-template-capital (0.545) and Chat-template-land (0.532) showed strong performance, indicating high practical utility. This demonstrates that a much smaller, 8B parameter model can be specialized to perform at a very high level on specific, complex tasks. Conversely, some models like Alpaca-intellectual (0.313) scored lower, indicating that while semantic understanding was achieved (Semantic Avg. 0.894), generating outputs that were simultaneously relevant, coherent, and accurate was a greater challenge for that specific combination of domain and format. This variability underscores that performance is a function of not just the data, but also the interaction between the data and the instruction format.

Impact of Instruction Format. The choice of instruction format (Alpaca vs. Chat-template) had a significant and domain-dependent impact on performance, particularly in the AI-Judge evaluation. No single format was universally superior. In the Intellectual Property domain, the Chat-template model (0.593) vastly outperformed the Alpaca version (0.313). In contrast, for Competition Law, the Alpaca model (0.495) was clearly superior to its Chat-template counterpart (0.367).

This suggests that the conversational, structured nature of the Chat-template format may be better suited for domains requiring more nuanced, explanatory answers (like Intellectual Property), while the more direct question-answer style of the Alpaca format may be more effective for domains that are more fact-driven (like Competition Law). This finding highlights the importance of format selection as a key hyperparameter in the fine-tuning process.

4.2. Training Efficiency

The fine-tuning process demonstrated significant efficiency gains, with Unsloth accelerating training by up to 8.8x compared to standard Hugging Face procedures, allowing all 20 fine-tuning runs (10 domains x 2 formats) to be completed within the allocated compute time. Resource utilization on the Karolina GPU nodes was efficient, with typical GPU and memory utilization rates between 85-90% during training runs. Overall, the project utilized approximately 600 GPU hours for fine-tuning, evaluation, development, and testing. Based on the Karolina A100 GPU power consumption and utilization, the estimated CO2 footprint for the GPU usage during the project is calculated to be between 56.84 kg and 64.8 kg CO2.

4.3. Qualitative Analysis

We conducted a domain-specific fine-tuning of the LLaMA 3.1-8B-Instruct model for legal applications, utilizing synthetic data derived from our proprietary corpus. The fine-tuning process led to a marked improvement in the model's alignment with domain-specific content. When deployed within our existing RAG pipeline, the fine-tuned models exhibited enhanced performance in terms of citation precision, logical coherence, and legal knowledge representation.

5. Discussion

Interpretation of Findings. Our results confirm that PEFT effectively specialized the Meta-Llama 3.1 8B model into capable experts for ten distinct Turkish legal sub-domains, validating our central hypothesis. Significant improvements over the base model were observed across semantic similarity and, notably, AI-judge metrics reflecting practical utility, demonstrating successful assimilation of domain-specific knowledge as in Table 3.

Challenges and Limitations. Several limitations should be acknowledged. Firstly, while extensive, our datasets relied partly on synthetic generation; potential biases or limitations in scope and quality may exist, requiring ongoing curation. Secondly, our evaluation, though multi-faceted, relied on automated metrics and AI judges (gpt-4o-mini), which have inherent limitations compared to systematic human expert evaluation. Thirdly, this study focused on specializing the 8B model, and our evaluation was consequently scoped to measure the improvement gained over its base version. We did not perform direct comparative benchmarking against proprietary very large models like GPT-4 and Claude 3. This was a deliberate choice rooted in data governance; our privacy policies prohibit the sharing of our specialized and sensitive legal datasets with external, third-party APIs. While this limits a direct comparison with the largest closed-source models, it reflects a realistic constraint for many legal tech applications dealing with confidential data. Finally, generalizability beyond the ten targeted Turkish legal domains requires further investigation.

Regarding cross-lingual and cross-system applicability, it is important to distinguish between our methodology and our models. The data generation pipeline we developed is highly adaptable and could be applied to other languages and legal systems by substituting the foundational legal corpora. However, the fine-tuned models produced in this study are inherently specific to Turkish law. Due to the unique jurisprudential principles, statutes, and terminology of each legal system, these models would not be directly applicable to another country's legal framework (e.g., German or French law) without undergoing a similar, rigorous fine-tuning process with relevant local data. This specialization is a feature, not a bug, underscoring our thesis that high performance in niche domains requires tailored, domain-specific training.

Future Work. Building on this foundation, our future work will proceed in several key directions, with a particular focus on exploring domain interaction and model capabilities. A key next step is developing a single, comprehensive legal model trained on the aggregated dataset from all ten domains. The goal of this "generalist expert" is to determine whether a larger, multi-domain training set can produce a model that matches or exceeds the performance of the individual specialists across their respective tasks. This will help answer a crucial question: is it more effective to deploy a suite of specialized agents or a single, more powerful generalist? Further research will also involve leveraging EuroHPC resources to train a larger Turkish reasoning model, experimenting with diverse base SLMs, and enhancing datasets through continuous expert validation. We also plan to develop advanced, domain-specific RAG integrations and strengthen our evaluation protocols with systematic human expert assessment. Finally, real-world pilot studies integrating these models into legal professional workflows are planned to assess practical utility and gather user feedback, bridging the gap between research and practice.

6. Conclusion

This paper presented the successful development and evaluation of specialized language models for ten distinct Turkish legal sub-domains through the fine-tuning of a relatively small base model, Meta-Llama 3.1 8B. Leveraging the computational power of the EuroHPC Karolina supercomputer and employing efficient methodologies, including PEFT with LoRA/QLoRA, orchestrated by Axolotl, and significantly accelerated by the Unsloth framework, we demonstrated the ability to create domain-specific expert models.

Our key finding underscores the viability and effectiveness of this approach: fine-tuning smaller language models presents a practical, computationally efficient, and economically advantageous strategy for achieving high performance on specialized tasks within complex domains like law. The resulting models exhibited significant improvements over the general-purpose base model in handling domain-specific nuances, as evidenced by our comprehensive evaluation using lexical, semantic, and AI-judge metrics.

The successful specialization of the 8B model into ten distinct legal expert models offers a compelling alternative to relying exclusively on larger, more resource-intensive general-purpose LLMs. This work contributes valuable

resources to the Turkish legal tech ecosystem and provides strong empirical support for the use of specialized SLMs in building modular, cost-effective agentic workflows. Ultimately, this research highlights the potential of targeted domain adaptation on high-performance computing infrastructure to advance the development and democratization of specialized AI solutions across various knowledge-intensive fields.

Data and Model Availability

The synthetic datasets generated and used for fine-tuning in this study are proprietary and are not publicly available due to their direct connection to our core business logic and proprietary data sources. However, in the spirit of contributing to the research community, we plan to open-source two of our best-performing fine-tuned models for the Turkish legal domain. The selected models will be released on the Hugging Face Hub under an Apache 2.0 license. The models will be accessible at the following URL upon release: https://huggingface.co/newmindai.

Acknowledgements

This research was supported by the EuroHPC Joint Undertaking (EuroHPC JU) under the Benchmark Access grant agreement No EHPC-BEN-2024B11-003. The authors gratefully acknowledge the computational resources provided by the IT4Innovations National Supercomputing Center (Czech Republic) on the Karolina supercomputer, made available through the EuroHPC JU.

Appendix A. Detailed Benchmark Results

Table A.4: Detailed evaluation results on lightEval benchmarks (Arc, Hellaswag, TruthfulQA, Winogrande). Fine-tuned model names are shortened for brevity; all start with ’Llama-3.1-8B-Instruct-’ followed by the domain and format shown (e.g., ’energy-chat-template’). Best ‘Overall’ score per domain pair is bolded, the second best is underlined.

Domain	Model Suffix	Arc	Hellaswag	TruthfulQA	Winogrande	Overall
(BASE)	meta-llama/Llama-3.1-8B-Instruct	0.5358	0.3805	0.4631	0.4834	0.4657
Instruct	meta-llama/Llama-3.1-70B-Instruct	0.4445	0.2584	0.4444	0.5039	0.4128
Instruct	meta-llama/Llama-3.3-70B-Instruct	0.2980	0.2470	0.4480	0.5060	0.3748
Energy	energy-chat-template	0.5179	0.3708	0.4175	0.4992	<u>0.4514</u>
Energy	energy-alpaca	0.5256	0.3715	0.4250	0.4961	**0.4545**
Tax	tax-chat-template	0.5171	0.3707	0.4228	0.5055	<u>0.4540</u>
Tax	tax-alpaca	0.5188	0.3693	0.4338	0.5000	**0.4555**
Competition	competition-chat-template	0.5239	0.3750	0.4269	0.4976	<u>0.4559</u>
Competition	competition-alpaca	0.5222	0.3745	0.4353	0.5008	**0.4582**
KVKK	kvkk-chat-template	0.5188	0.3710	0.4267	0.5000	**0.4541**
KVKK	kvkk-alpaca	0.5102	0.3693	0.4375	0.4882	<u>0.4513</u>
Health	health-chat-template	0.5162	0.3719	0.4093	0.4968	**0.4485**
Health	health-alpaca	0.5188	0.3712	0.4017	0.4981	<u>0.4475</u>
Capital	capital-chat-template	0.5179	0.3724	0.4368	0.4889	<u>0.4540</u>
Capital	capital-alpaca	0.5085	0.3734	0.4495	0.4866	**0.4545**
Environmental	environmental-chat-template	0.5119	0.3746	0.4409	0.4858	<u>0.4533</u>
Environmental	environmental-alpaca	0.5026	0.3745	0.4566	0.4834	**0.4543**
Fund	fund-chat-template	0.5119	0.3718	0.4452	0.4834	<u>0.4531</u>
Fund	fund-alpaca	0.5009	0.3716	0.4582	0.4834	**0.4535**

Intellectual	intellectual-chat-template	0.5077	0.3726	0.4326	0.4882	0.4503
Intellectual	intellectual-alpaca	0.5034	0.3719	0.4472	0.4905	**0.4532**
Land	land-chat-template	0.5111	0.5939	0.5232	0.7103	0.5846
Land	land-alpaca	0.5094	0.5951	0.5395	0.7056	**0.5874**

Table A.5: Light eval tasks token analysis.

Dataset	Input Tokens				Output Tokens				Total Tokens	All Models Total (x20)
	Max	Min	Mean	Total	Max	Min	Mean	Total		
hellaswag	328	2	133	5337376	454	27	166	6658344	11995720	239914400
arc	297	42	100	11789	2	2	2	2344	14133	282660
truthful-qa	82	5	18	14976	43	2	15	12703	27679	553580
winograde	72	17	33	85137	2	2	2	5110	90247	1804940
								Total:	12127779	242,555,580.00

References

[1] Qin, W., & Sun, Z. (2024). Exploring the Nexus of Large Language Models and Legal Systems: A Short Survey. *arXiv preprint arXiv:2404.00990*.

[2] Mayer, T. (2023). AI and LLMs in Legal Technology: Revolutionizing Research and Document Analysis. *Advances in Computer Sciences*, **6**(1).

[3] Hadi, M. U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M. B., ... & Mirjalili, S. (2023). A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, **3**.

[4] Kaoutar, M. R., Chaima, B. J., Omar, B., & Outmane, B. (2024, October). Unlocking the Potential of Large Language Models in Legal Discourse: Challenges, Solutions, and Future Directions. In 2024 *Sixth International Conference on Intelligent Computing in Data Sciences (ICDS)* (pp. 1-7). IEEE.

[5] Mercan, G., & Selçuk, Z. V. (2024). Artificial intelligence (AI) activities in legal practices. *International Journal of Eurasian Education and Culture*, **9**(25), 131-144.

[6] Bhatnagar, M., & Huchhanavar, S. (2025). Enhancing legal assistance with AI: a comprehensive approach to intent classification and domain specific model tuning. *Artificial Intelligence and Law*, 1-29.

[7] Gajulamandyam, D. K., Veerla, S., Emami, Y., Lee, K., Li, Y., Mamillapalli, J. S., & Shim, S. (2025, January). Domain Specific Finetuning of LLMs Using PEFT Techniques. In *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 00484- 00490). IEEE.

[8] Han, Z., Gao, C., Liu, J., Zhang, J., & Zhang, S. Q. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

[9] Li, X., Wang, S., Zeng, S., Wu, Y., & Yang, Y. (2024). A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, **1**(1), 9.

[10] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & McGrew, B. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

[11] *Introducing Claude*. (n.d.). https://www.anthropic.com/news/introducing-claude

[12] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., ... & Fiedel, N. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, **24**(240), 1-113.

[13] Zhong, H., Xiao, C., Tu, C., Zhang, T., Liu, Z., & Sun, M. (2020). How does NLP benefit legal system: A summary of legal artificial intelligence. *arXiv preprint arXiv:2004.12158*.

[14] Bommarito II, M., & Katz, D. M. (2022). GPT takes the bar exam. *arXiv preprint arXiv:2212.14402*.

[15] Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., & Androutsopoulos, I. (2020). LEGAL-BERT: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

[16] Elwany, E., Moore, D., & Oberoi, G. (2019). Bert goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding. *arXiv preprint arXiv:1911.00473*.

[17] de Oliveira Fornasier, M. (2021). Legal education in the 21st century and the artificial intelligence. Revista Opinião Jurídica, **19**(31), 1-32.

[18] Lai, J., Gan, W., Wu, J., Qi, Z., & Yu, P. S. (2024). Large language models in law: A survey. *AI Open*.

[19] Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L. M., Rothchild, D., ... & Dean, J. (2021). Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

[20] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

[21] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM computing surveys*, **55**(12), 1-38.

[22] Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

[23] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019, May). Parameter-efficient transfer learning for NLP. In *International conference on machine learning* (pp. 2790-2799). PMLR.

[24] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). Lora: Low-rank adaptation of large language models. *ICLR*, **1**(2), 3.

[25] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, **36**, 10088-10115.

[26] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, **33**, 9459-9474.

[27] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020, November). Retrieval augmented language model pre-training. In *International conference on machine learning* (pp. 3929-3938). PMLR.

[28] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

[29] Schick, T., & Schütze, H. (2020). Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

[30] Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., ... & Metzler, D. (2021). Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*.

[31] Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., ... & Wei, F. (2024). The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, **1**.

[32] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., ... & Wen, J. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, **18**(6), 186345.

[33] Oflazer, K. (1994). Two-level description of Turkish morphology. *Literary and linguistic computing*, **9**(2), 137-148.

[34] Stefan-It. (n.d.). GitHub - stefan-it/turkish-bert: Turkish BERT/DistilBERT, ELECTRA, ConvBERT and T5 models. GitHub. https://github.com/stefan-it/turkish-bert

[35] Turker, M., Ari, M. E., & Han, A. (2024). Vbart: The turkish llm. *arXiv preprint arXiv:2403.01308*.

[36] Turan, T., & Küçüksille, E. U. (2024). Legal text classification in Turkey: A machine learning approach to divorce and zoning decisions. *Uluslararası Mühendislik Tasarım Ve Teknoloji Dergisi*, **6**(2), 53-63. https://doi.org/10.70669/ijedt.1491511.

[37] Çetindağ, C., Yazıcıoğlu, B., & Koç, A. (2023). Named-entity recognition in Turkish legal texts. *Natural Language Engineering*, **29**(3), 615-642.

[38] Çetinoğlu, Ö., & Cöltekin, Ç. (2023). Two languages, one treebank: building a Turkish–German code-switching treebank and its challenges. *Language Resources and Evaluation*, **57**(2), 545-579.

[39] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., ... & Vasic, P. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

[40] Tatsu-Lab. (n.d.). Tatsu-Lab/STANFORD ALPACA: Code and documentation to train Stanford's alpaca models, and generate the data. GitHub. https://github.com/tatsu-lab/stanford alpaca

[41] Openai. (n.d.). openai-python/chatml.md at release-v0.28.0 · openai/openai-python. GitHub. https://github.com/openai/openai-python/blob/release-v0.28.0/chatml.md.

[42] Axolotl-Ai-Cloud. (n.d.). GitHub - axolotl-ai-cloud/axolotl: Go ahead and axolotl questions. GitHub. https://github.com/axolotl-ai-cloud/axolotl.

[43] Unslothai. (n.d.). GitHub - unslothai/unsloth: Finetune Qwen3, Llama 4, TTS, DeepSeek-R1 & Gemma 3 LLMs 2x faster with 70% less memory!. GitHub. https://github.com/unslothai/unsloth.

[44] Huggingface. (n.d.). GitHub - huggingface/lighteval: Lighteval is your all-in-one toolkit for evaluating LLMs across multiple backends. GitHub. https://github.com/huggingface/lighteval.

[45] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

[46] Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[47] Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

[48] emrecan/bert-base-turkish-cased-mean-nli-stsb-tr · Hugging Face. (2024, June 16). https://huggingface.co/emrecan/bert-base-turkish-cased-mean-nli-stsb-tr.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 136–145

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Porting Epistasis Detection Methods to EuroHPC Supercomputers

Ricardo Nobre[a,b,*], Aleksandar Ilic[a,b], Leonel Sousa[a,b]

[a] *INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal*
[b] *Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal*

Abstract

Epistasis detection searches have been accelerated throughout the years resorting to different types of parallel processors. GPU-accelerated approaches that we have proposed achieve the highest per-device and per-node performance reported in the literature for exhaustive search methods. This paper reports our efforts to port our GPU-based solutions to large-scale GPU-accelerated systems. In particular, it documents key developments for retargeting our epistasis detection codes to the GPU-accelerated partitions of the LUMI and MeluXina EuroHPC supercomputers. This paper touches on the different parallelization approaches taken, providing explanations on their implementation, and documents the efforts undertaken to make codes that have originally been developed in CUDA compatible with the state-of-the-art AMD GPUs available on the LUMI supercomputer. Overall, high-scalability has been achieved, resulting in the new improved codes reaching unprecedented levels of epistasis detection performance on the targeted systems. In particular, one of the codes achieved a parallel efficiency of 96% on 1024 nodes in comparison to single-node execution.

Keywords: epistasis detection; supercomputers; parallel processing; GPU programming; hardware interoperability

1. Introduction

Genome Wide Association Studies (GWAS) have successfully found several associations between genotype and a number of human conditions [9]. However, many more associations are likely to be discovered. Such discoveries can have deep implications on society through improving personalized treatment [1, 5, 18], mitigating the spreading of viruses [4], and several other genetics-enhanced high impact use cases, such as forensics investigations [17].

The massive parallel processing potential of modern supercomputers is often achieved through the use of GPU accelerators. As a matter of fact, the most recent TOP500 list (November, 2024) [19] ranks nine supercomputers with GPUs as the ten most powerful computer systems. The use of GPUs enables packing a much higher performance capability per rack and per unit of energy than if relying only on CPUs, while also benefiting from a more mature ecosystem than other comparatively powerful types of accelerator devices. Also contributing to their deployment in large numbers in supercomputers is the fact that there is a significant overlap between the applications that can be

* Corresponding author. Tel.: (+351) 21 310 0294 ; fax: (+351) 21 314 5843.
E-mail address: ricardo.nobre@inesc-id.pt

1877-0509
10.1016/j.procs.2025.08.240

accelerated with GPUs – applications with a high degree of parallelism – and those that can benefit from the massive peak processing capability achieved through horizontal large-scale replication of modern computer nodes.

The European High-Performance Computing Joint Undertaking (EuroHPC JU) has procured ten state-of-the-art supercomputers [2], all of which have GPU-accelerated partitions. While most have NVIDIA GPUs, LUMI [7], which coincidentally is the most powerful one currently running and available to end-users – LUMI reaches 386 PFLOP/s of sustained performance, JUPITER [3] is expected to reach 1 EFLOP/s of sustained performance but is not yet operational – relies on AMD GPUs. Thus, in order to make software able to exploit the LUMI GPU partition (LUMI-G), which is a requirement to leverage the potential of this supercomputer, one must develop software in such a way that enables compilation targeting AMD GPUs and that is able to make efficient use of their hardware capabilities.

In this paper, we document the efforts undertaken, including the main challenges faced and how they were addressed, to port a number of our key exhaustive epistasis detection bioinformatics codes to modern supercomputing platforms. These programs were mostly developed in CUDA and outperform other solutions across various interaction orders when run on the same or comparable GPU hardware. Specifically, our codes achieve throughput that matches or exceeds that of other solutions utilizing CUDA cores or tensor cores, thereby representing the state-of-the-art in high-throughput epistasis detection methods. The aforementioned efforts entailed porting codes from CUDA to HIP, as well as adding a layer of parallelization to add support for multiple nodes in a way that is suited/specialized at/to the different codes. Thus, making them primed to achieve high utilization of the targeted systems: LUMI [7] – which presents an opportunity to expand our solutions to AMD GPUs and enables performing runs at a very large scale – and MeluXina [8] – which featured the most powerful NVIDIA GPUs when we started the reported developments.

The developments reported in this paper can be of interest to both bioinformaticians and supercomputer users/programmers with some interest in the topic. In particular, the source code used to perform the experiments reported in this paper [1] can potentially be of use to other developers of high-throughput bioinformatics solutions.

Section 2 presents the tackled bioinformatics application in a level of detail that is sufficient to understand the undertaken porting and parallelization. Section 3 introduces the state-of-the-art codes adapted for execution on the EuroHPC supercomputers. The porting / parallelization efforts are detailed in Section 4. Section 5 analyses the achieved performance and scalability for a comprehensive battery of tests. Finally, Section 6 presents concluding remarks.

2. Epistasis detection in a high-performance computing context

Searching for statistical associations between genotype and phenotype considering genetic markers individually is often not enough [20]. Epistasis detection is focused on finding non-additive genetic interactions, i.e. epistatic interactions, with influence on a given trait under study. These searches are accomplished through the processing of datasets representing the genetic configurations for a set of cases and controls, i.e. case-control datasets. The markers represented in these datasets are typically single nucleotide polymorphisms (SNPs), each representing variation at a single nucleotide in the DNA of at least 1% of a population under study. SNPs are identified through processing multiple DNA sequences (from different individuals of a given population), which are compared with the genetic data of a reference sequence as part of another foundational computational bioinformatics process called variant calling.

Epistasis detection represents a combinatorial search problem that is very computationally intensive due to the large evaluation space to cover. This is especially the case if one relies on exhaustive methods, designed to be as precise as possible, and considering high-order SNP combinations (involving three or more SNPs), which opens up more possibilities for finding genotype-phenotype correlations. Notice that while the expected processing time increases linearly with the number of samples to process, it increases extremely fast with the number of SNPs in the input dataset, and even faster with the interaction order. For example, for 100,000 SNPs, a pairwise search results in 5.0 Giga (10^9) combinations of SNPs to evaluate. However, a third-order / fourth-order search entails evaluating 166.7 Tera (10^{12}) / 4166.4 Peta (10^{15}) SNP combinations. This is due to the amount of combinations increasing by $\approx 4\times$, $\approx 8\times$ and $\approx 16\times$ in second-order, third-order and fourth-order, when doubling the amount of SNPs to process.

In exhaustive epistasis detection searches, for each of the SNP combinations to evaluate, a score is calculated on top of the counts of the different possible genotypes that are seen in cases and controls. This score expresses the

[1] Source code and datasets available at: https://github.com/hiperbio/epistasis-supercomputers

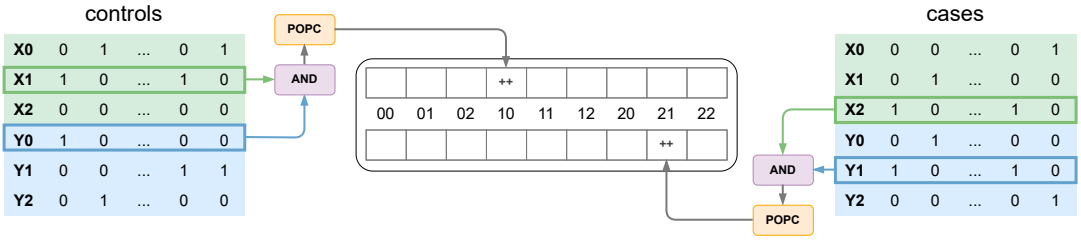


Fig. 1: Construction of contingency table for a pair of SNPs relying on one-hot encoding representation and bitwise processing.

Table 1: Epistasis detection solutions subjected to porting efforts targeting EuroHPC supercomputers that are tackled in this paper.

Approach	Public code repository	Order(s)	Prog. model(s)	Targeted hardware
CUDA-Episdet [14]	`hiperbio/cuda-episdet`	2nd, 3rd	OpenMP, CUDA	Any NVIDIA GPU (Maxwell to Ampere)
Tensor-Episdet [10, 13]	`hiperbio/tensor-episdet`	2nd, 3rd	CUDA	Turing GPUs (also supports Ampere)
Epi4Tensor [12]	`hiperbio/Epi4Tensor`	4th	OpenMP, CUDA	Ampere GPUs (also supports Turing)
Crossarch-Episdet [11]	`hiperbio/crossarch-episdet`	2nd, 3rd	OpenMP, SYCL	CPU/GPU with SYCL support via oneAPI

association strength between the SNP combination being evaluated at a given time and the phenotype under study, i.e. the impact of that genetic configuration in determining the existence of a given trait or condition. For challenging datasets in regard to samples, it is the genotype counting, and not the scoring calculation, that determines the achieved performance [13]. Hence, it is often the focus of optimization on high-throughput approaches. Fig. 1 exemplifies this process, considering a kind of representation often used in high-throughput approaches in the literature that one-hot encodes genotype information and stores data pertaining to cases and controls in separate memory regions.

There are three base genotypes resulting from combining one allele from each of the parents. These are the homozygous major, heterozygous, homozygous minor genotypes. Using this type of representation, each SNP is represented by six bitvectors, three for cases and three for controls, each with as many bits as the amount of samples it represents. Counting the occurrences of the different genotypes on cases and controls entails filling a table with 2×3^k cells, where k is the interaction order. Notice that in third and fourth order searches there are 27 ($= 3^3$) and 81 ($= 3^4$) genotypes that result from combining the three base genotypes. Combining and counting genotypes using such representation can be done using bitwise AND and POPC (population count, i.e. count set bits), which are fast on most modern devices.

Epistasis detection is highly data-parallel, since it relies on applying the same operations to different combinations of SNPs: genotype counting, scoring calculations, reduction and identification of the set (or sets) of SNPs most associated with phenotype. Some of the state-of-the-art approaches, including those developed by us, already target the type of GPU hardware that is available on supercomputers. However, several of the implementations of those methods only support NVIDIA GPUs (e.g. [10, 12, 13, 14]). This is particularly the case with the approaches that make use of the matrix processing units in modern microarchitectures – all have been developed around the use of NVIDIA GPUs [6, 10, 12, 13]. Furthermore, most of the existing approaches are not implemented in a way that exploits the true potential of supercomputers. As a matter of fact, with the exception of CoMet [6], used for epistasis searches on the Summit supercomputer, no other tensor-core approach in the literature supports multi-node GPU-based execution.

3. High-throughput codes extended to EuroHPC supercomputers

Table 1 introduces the epistasis detection solutions that have been extended to efficiently use EuroHPC supercomputers. All listed solutions have in common the fact that they rely on exhaustive methods, i.e. the complete SNP combination space is evaluated. However, these approaches differ in regard to key aspects that can be ground for significant specialization, such as the interaction order, the programming models used and/or the targeted hardware.

When introduced, CUDA-Episdet [14] was the fastest approach out of those compatible with a broad range of consumer GPU hardware – although still NVIDIA-only – achieving on average 3× higher performance than MPI3SNP [16], a state-of-the-art approach similar in the hardware features used, on the same GPUs from four microarchitectures (Maxwell 2.0, Pascal, Volta, Turing). CoMet [6], another approach from the literature that had been

recently introduced as well, was able to achieve higher performance, but only when targeting GPUs with the Volta microarchitecture, which were exclusive to the data-center (Tesla V100) and high-end prosumer market (Titan V).

Around the same time, we also introduced an epistasis detection solution [10] that later evolved and became known as Tensor-Episdet [13]. In its earliest reported development stage, it already achieved 6× to 8× higher performance at second-order and third-order searches on a mid-range Turing GPU (GeForce 2070S) in comparison to a supercomputer-grade GPU (V100) using the state-of-the-art CoMet approach [6]. It achieved that feat through the use of the second generation matrix units available in the Turing microarchitecture, which was made possible with a carefully designed algorithm that allowed translating the output of tensorized XOR+POPC (newly introduced operation) to that one would obtain using tensorized AND+POPC (not supported in Turing). Another interesting feature of that approach is that it is able to construct a complete contingency table, making it compatible in principle with any objective scoring function other than the one used (K2 Bayasian scoring), deriving a significant part of its values from that of other counts, which speeds up computations quite significantly. Around the time it was publicly released as Tensor-Episdet [13], it included key improvements such as the use of an internal dataset representation that is more suited at efficient use of the GPU compute and memory subsystems. As a result, it achieved in its later form 7× (second-order) and 11× (third-order) higher performance on the GeForce 2070S – 10× and 18× on a Titan RTX – when compared to CoMet on a single V100. Experiments including an additional objective scoring function (mutual information) showed empirically that the method of constructing contingency tables could be used with different statistics calculations and that, at least for challenging datasets, it was the determinant for achieving high performance.

Epi4Tensor [12] extended upon Tensor-Episdet in two fronts. These are the exploitation of new features available in a newer microarchitecture (Ampere), in particular the newly introduced tensorized AND+POPC support on the GPU matrix engines, and the generalization of the method to fourth-order searches. From its inception, this software has been subject to optimization on systems with A100 GPUs. While the use of XOR+POPC is still supported as a means to provide backwards compatibility with Turing GPUs, Epi4Tensor primarily uses AND+POPC (introduced in the Ampere microarchitecture). Thus, not requiring the translation between these two types of operations. To achieve efficient use of the matrix processing units, a scheme specialized to fourth-order searches has been devised to count the occurrences of genotypes. In particular, the developed method precombines two sets of two blocks of SNPs on the general purpose cores of the GPU and then combines the result of each operation on its matrix cores. Epi4Tensor supports multiple GPUs, but only on the same node, achieving the comparatively high performance of 835.4 tera sets processed per second scaled to sample size on an HGX-A100 8-GPU system [15]. As is the case of Tensor-Episdet, Epi4Tensor has been implemented in CUDA and relies on the CUTLASS library for accessing the GPU matrix cores.

Crossarch-Episdet [11] is a port of CUDA-Episdet that uses the DPC++/SYCL abstraction layer for programming heterogeneous hardware. The use of SYCL provides an open alternative to single-architecture proprietary languages, with support for different vendors, as well as different types of devices, through the same source code. In addition to targeting CPUs, it is capable of targeting GPUs from different vendors (AMD, Intel, NVIDIA) through the OneAPI toolchain. Support for Intel is incorporated into the oneAPI Base Toolkit, with support for AMD and NVIDIA GPUs granted through Codeplay oneAPI plugins. This code represents one of our first efforts to achieve performance portability in the context of epistasis detection methods through open standards programming languages and tools.

4. Porting the codes to EuroHPC supercomputers

This section focuses on describing the main steps undertaken to introduce compatibility with AMD GPUs and to use multiple nodes, both important to make use of the hardware resources available through the EuroHPC JU initiative.

4.1. Adding support to AMD GPUs through the HIP programming model

All codes that have been programmed in CUDA (CUDA-Episdet, Tensor-Episdet, and Epi4Tensor) were ported to AMD's HIP C++ programming model, thus enabling compatibility with the AMD GPUs on the LUMI GPU-accelerated partition through the ROCm software stack. Porting the different codes involved a varying degree of effort. In addition to porting from CUDA to HIP, which was to a significant extent helped with the use of the hipify code translation tool, additional code modifications had to be applied to take into account the different GPU microarchitecture. For example, in NVIDIA a warp has 32 threads, while a wavefront (AMD's equivalent to NVIDIA's warp)

has 64 threads in the microarchitecture of the MI250X GPU (CDNA2). Moreover, some functions used did not have an equivalent in HIP. In particular, the epistasis detection codes relying on tensor cores – Tensor-Episdet and Epi4Tensor – rely from their inception on the CUTLASS collection of CUDA C++ template abstractions for high-performance matrix-matrix multiplication and related computations, which has no direct equivalent on the ROCm software stack.

The use of CUTLASS, in relation to cuBLAS – which is arguably the de facto library for matrix-multiplication on NVIDIA hardware – enables a lot of additional opportunities for customization and fine-grained tuning. Not only does it provide a fine-grained control over how the matrix multiplication operations are mapped to the multiprocessors in the GPU – called compute units (CUs) in AMD GPUs and streaming multiprocessors (SMs) in NVIDIA GPUs – through the specification of the operation tile sizes at various levels, but it also provides support for additional operation precisions. The CUTLASS library provides access to the 4-bit and 1-bit operations natively supported on the tensor cores of modern NVIDIA GPU microarchitectures, which are not supported under cuBLAS. In particular, 1-bit operations are useful for epistasis detection searches, since the representation of case-control datasets in a one-hot encoding scheme enables the use of bitwise operations as part of matrix operations performed to derive the genotype counts composing the contingency tables required to calculate genotype-phenotype association strength scores.

Having the capability to natively processes genetic data represented in a one-hot encoding scheme on matrix processing hardware is of very high value. This has been extensively exploited using XOR+POPC (Tensor-Episdet and Epi4Tensor) and AND+POPC (Epi4Tensor) in Turing and Ampere microarchitectures. However, those operations are not supported on the matrix processing hardware of the MI250X GPU. Thus, we decided to use hipBLAS – an interoperability layer that supports rocBLAS and cuBLAS as backends – on the ports targeting the AMD hardware. As is the case of cuBLAS, hipBLAS/rocBLAS supports operation with integer and floating-point datatypes.

For the correct construction of contingency tables, as long as 32-bits are used for accumulation – either INT32 or FP32 – in order to enable the processing of large datasets in regard to samples, it is indifferent which data type is used for representing the matrix inputs. This is the case because one is representing 1-bit data as a result of the dataset format used. It still leaves the question about which differences in performance are to be expected using the datatypes supported on the AMD MI250X GPUs through the rocBLAS library. The two numerical precisions supported that have the highest peak theoretical throughput are INT8 and FP16, which have the particularity of being equally rated at 383 TFLOP/s. In the scalabiliy results herein presented, we are using FP16 inputs and FP32 for accumulation.

4.2. Adding support for multiple nodes with multiple GPUs through MPI

While some of our epistasis detection codes originally had support for multiple GPUs on the same system, e.g. Epi4Tensor targeted the HGX A100 8-GPU platform [15] using OpenMP threads to orchestrate different GPUs, none of those codes originally supported multiple nodes. Naturally, this had to be addressed in order to enable exploiting the massive performance potential accessible through efficient utilization of modern GPU-accelerated supercomputers. In all cases, i.e. for all codes considered, we have adopted a parallelization scheme where different Message Passing Interface (MPI) processes were assigned a different GPU (one per process). For some codes, we deemed sufficient to utilize a static partitioning / scheduling scheme. However, for other codes we have used dynamic scheduling.

All codes under study rely on a set of nested loops to evaluate combinations of SNPs. Partitioning of work between different GPUs is done at the level of the outermost loop (1 work unit equals 1 outerloop iteration). In both CUDA-Episdet and Crossarch-Episdet, a SYCL port of the former, the host assigns the same number of combinations to process as part of each iteration of the outerloop. This makes static assignment of work to the different MPI processes viable in what pertains work distribution and scheduling. However, in the two codes using tensor cores (Tensor-Episdet and Epi4Tensor), due to the way combinations between SNPs are expressed – in blocks of SNPs, where each block with a given index is to be combined with itself and with all blocks with higher index – different loop iterations entail performing different amounts of work per outerloop iteration. Thus, we rely on a dynamic scheduling approach for Tensor-Episdet and Epi4Tensor to achieve a balanced load distribution, which is a requirement for high scalability. MPI process 0 (i.e. rank 0) distributes work to other MPI processes (as many as the number of targeted GPUs), which evaluate SNP combinations as a result of executing the assigned loop iterations. Thus, if using 8 GPUs (2 MeluXina GPU nodes), there are 9 MPI processes in total. Whenever the MPI process associated with a GPU (i.e. a worker process) terminates executing the loop iteration assigned to it at a given time, it asks MPI rank 0 (i.e. the coordinator process) for an additional loop iteration. The application terminates when all loop iterations have been processed.

In order to make efficient use of the resources used, i.e. the requested nodes, the aforementioned parallel processing scheme has been implemented relying on having one of the nodes with an extra MPI process. That node, in addition to MPI worker processes also hosts the MPI coordinator process. Notice that the CPUs experience minimal load, serving mostly to orchestrate computations rather than crunching data. This is especially the case for the codes using matrix/tensor cores (Tensor-Episdet and Epi4Tensor), where virtually all the computational burden falls on the GPUs.

Parallelization between nodes has been implemented in a way that contention on internode communication links is also not a problem. The amount of information transferred between nodes, as well as MPI processes in the same node, is very minimal in relation to computation. Communication only includes messages from the worker processes to the coordinator process asking for more SNP evaluation rounds to execute, indices of loop iterations transferred from the coordinator to worker processes representing the work to do, and sending the best found scores and corresponding sets of SNPs from the worker processes to the coordinator at the end of a given epistasis detection run. In addition to the message sizes being quite small, the frequency of communication is also very low. However, if required, communication can be further minimized through adjusting the SNP block size parameter, which represents the granularity with which operations are performed on each evaluation round accelerated with GPU matrix/tensor cores.

In addition to a high computation to communication ratio, to achieve high scalability it is important for work distribution to be performed in a fine-grained enough manner. However, very fine grained distribution of work through using a smaller SNP block size results in a drop in throughput using the matrix/tensor cores. Epi4Tensor was particularly challenging in this regard in the initial versions extended with MPI, with datasets other than those with extremely large amounts of SNPs not resulting in evenly divided workload between the targeted GPUs. The load balancing issues were mitigated by fusing the two outermost loops. This allows to achieve a more fine-grained work distribution without requiring increasing the SNP blocks size, the main determiner of throughput on the matrix/tensor cores.

Load balancing at runtime through an extra MPI process is attained specifying to the SLURM workload manager the total number of processes (as many as the GPUs used plus 1), the number of GPUs per node (4 for MeluXina, 8 for LUMI), and the GPU binding strategy used. Notice that while one of the GPUs on one of the nodes gets assigned two processes, as one more process is used than the amount of GPUs, being one the coordinator process and the other the worker processes, only the latter actually uses the GPU. As an alternative to carefully setting the SLURM settings, in systems where achieving the desired behavior through the previously referred approach poses challenges, the codes that use dynamic scheduling could be modified to make the process with rank 0 also perform work in addition to the coordination of other processes. Thus, requiring only the use of as many MPI processes as the number of GPUs used.

5. Performance and scalability study

This section introduces the experimental setup and the results achieved on the two targeted supercomputers: LUMI and MeluXina. The results are discussed in the context of the different codes addressed and the applied modifications.

5.1. Software stack used

Most of the experiments on MeluXina rely on CUDA Toolkit 12.2 and GPU driver 535.161.08. Only the reported experiments for Tensor-Episdet, which have been performed earlier, relied on CUDA 11.7 and GPU driver 515.105.01 (the systems' default when performing the experiments). The DPC++/C++ Compiler (icpx) from oneAPI 2025.01, after installation of the corresponding Codeplay plugins for AMD and NVIDIA GPUs, was used to compile Crossarch-Episdet on LUMI and MeluXina. On LUMI, the HIP compiler driver (hipcc), which is used as a wrapper to the Clang/LLVM toolchain from ROCm, replaces nvcc in the compilation of Tensor-Episdet and Epi4Tensor. The experiments reported on this paper for this supercomputer have been performed relying on ROCm 6.0.3. The CUTLASS matrix processing library is used by Tensor-Episdet (v1.33) and Epi4Tensor (v3.4.1) to access the tensor cores of the NVIDIA GPUs on MeluXina. Different versions of this library have been used due to changes in the API. As a replacement to the CUTLASS library (NVIDIA-only), when targeting AMD GPUs with Tensor-Episdet and Epi4Tensor, the hipBLAS marshaling library has been used (serving as a wrapper to the rocBLAS library).

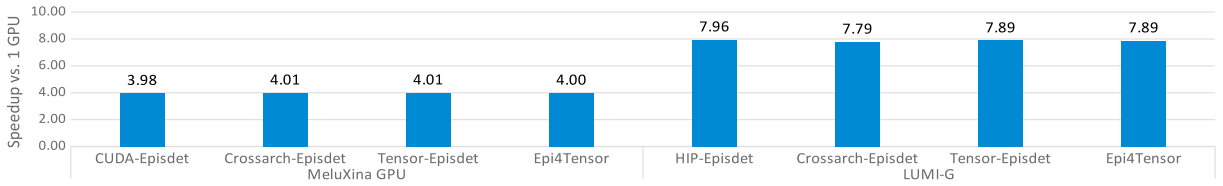


Fig. 2: Speedups for single-node runs. CUDA-Episdet (8192 SNPs), Crossarch-Episdet (8192 SNPs), Tensor-Episdet (16384 SNPs) and Epi4Tensor (2048 SNPs). Tensor-Episdet and Epi4Tensor have been executed with 524288 samples on MeluXina, all other runs process 32768 samples.

5.2. Datasets used

The scalability study reported in this paper shows data for runs with synthetic datasets of different dimensions with respect to SNPs and samples (half cases/controls). The use of synthetic datasets is sufficient to evaluate the considered codes. Since those perform searches in such a way that the particular genotypic information does not have an impact on the type and number of operations performed. The reported experiments are the result of processing datasets that have overall enough samples to achieve performance saturation for the tackled codes on the targeted GPUs.

The scalability experiments reported for CUDA-Episdet and Crossarch-Episdet – both performing third-order searches – rely on datasets with 32768 samples, on LUMI and MeluXina. The experiments for the approaches using tensor/matrix cores (Tensor-Episdet and Epi4Tensor) were performed with 32768 (LUMI) and 524288 (MeluXina) samples. While Tensor-Episdet and Epi4Tensor make extensive use of matrix units on the NVIDIA and AMD GPUs, only the NVIDIA GPUs have native support for tensorized 1-bit operations. Thus, making GPUs on LUMI achieve performance saturation with comparatively less samples. Notice that for a given amount of SNPs a dataset (or dataset portion, i.e. a block of SNPs) with 32768 samples represented with a 16-bit datatype entails the processing of as many bits of information as a dataset with 524288 samples using individual bits to represent one-hot encoded data.

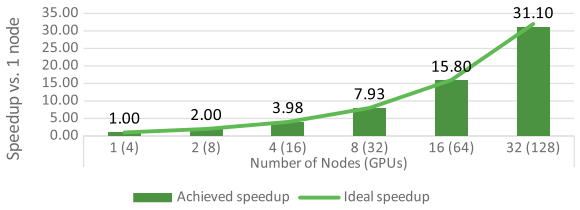
We present two distinct sets of experiments: one that evaluates intra-node scalability, and another that, using datasets with more SNPs, evaluates inter-node scalability. Intra-node experiments rely on 8192 SNPs for CUDA/HIP-Episdet and Crossarch-Episdet, and on 16384 SNPs and 2048 SNPs for Tensor-Episdet and Epi4Tensor, respectively. Inter-node experiments with Tensor-Episdet and Epi4Tensor use 32768 and 4096 SNPs, while those with CUDA/HIP-Episdet and Crossarch-Episdet use 16384 SNPs on MeluXina and 32768 SNPs on LUMI. The use of datasets with more SNPs allows one to assess the scalability in a scenario closer to what would happen for large-scale runs.

In the matrix-accelerated codes, genotypic data is combined in blocks of bitvectors representing data for SNPs. Runs with Epi4Tensor and Tensor-Episdet relied on block sizes of 32 and 256 SNPs (the codes' default setting), respectively. Runs with CUDA-Episdet and Crossarch-Episdet both relied on the processing of 131072 SNP combinations per evaluation round. CUDA-Episdet runs on Meluxina and LUMI relied on a work-group size of 32 and 128, respectively. In the case of LUMI, slightly higher performance has been achieved using a work-group size with double the amount of threads in the wavefronts on the CDNA2 microarchitecture. Crossarch-Episdet has been executed with 64 threads per work-group on both systems. For this code, work-groups of 128 threads resulted in worse performance compared to 64 threads. Moreover, using the same work-group size in both systems when running a code designed to be compatible with hardware from different vendors, allows to assess cross-device portability to the highest extent.

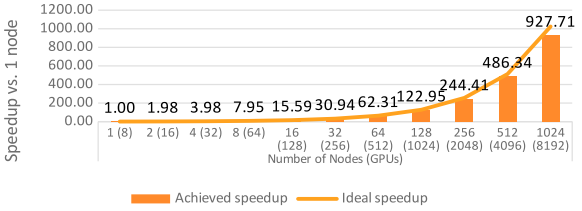
5.3. Scalability Results

The set expectations regarding performance using multiple GPUs and nodes have been achieved. To demonstrate the quality of the parallelization approaches used, we report the performance improvements achieved using the GPUs in a single node (intra-node parallelization), as well as those of using multiple nodes (inter-node parallelization). Fig. 2 depicts the speedups achieved using all available GPUs on a single MeluXina-GPU node (4 GPUs) and on a single LUMI-G node (8 GPUs). Fig. 3, Fig. 4, Fig. 5 and Fig. 6 depict the speedups achieved on multiple nodes (vs. single-node), for runs with CUDA/HIP-Episdet, Crossarch-Episdet, Tensor-Episdet, and Epi4Tensor, respectively.

As can be seen from the intra-node scalability experiments here reported, performance increased close to linearly with the number of devices present in a node. The MeluXina GPU-accelerated nodes have 4 NVIDIA A100's, while LUMI-G nodes have 4 AMD MI250X's. We count the 4 MI250X as 8 GPUs, as each is seen as two devices by

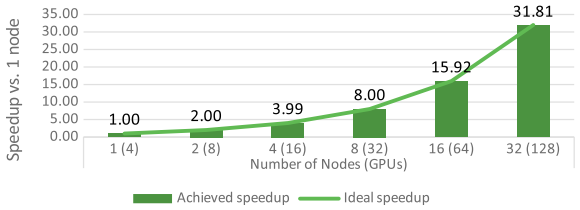


(a) MeluXina.

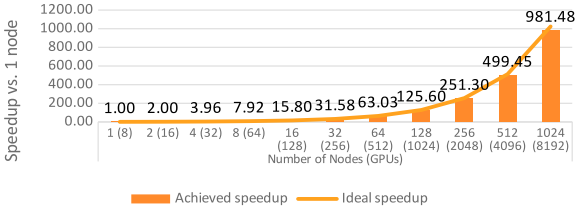


(b) LUMI.

Fig. 3: Speedups with CUDA/HIP-Episdet on multiple nodes (MeluXina: 16384 SNPs × 32768 samples, LUMI: 32768 SNPs × 32768 samples).

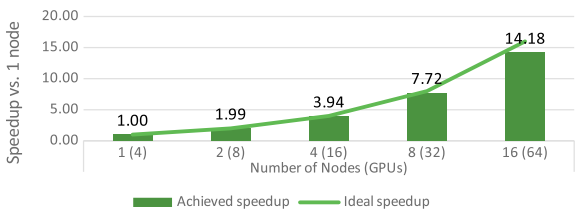


(a) MeluXina.

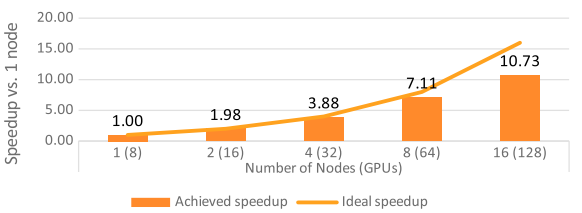


(b) LUMI.

Fig. 4: Speedups with Crossarch-Episdet on multiple nodes (MeluXina: 16384 SNPs × 32768 samples, LUMI: 32768 SNPs × 32768 samples).

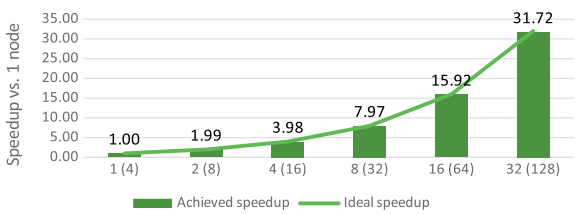


(a) MeluXina.

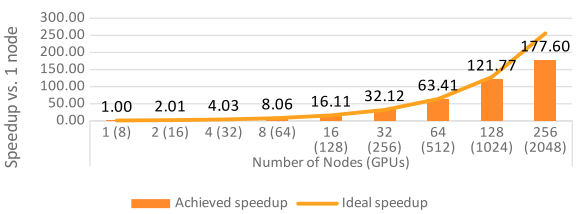


(b) LUMI.

Fig. 5: Speedups with Tensor-Episdet on multiple nodes (MeluXina: 32768 SNPs × 524288 samples, LUMI: 32768 SNPs × 32768 samples).



(a) MeluXina.



(b) LUMI.

Fig. 6: Speedups with Epi4Tensor on multiple nodes (MeluXina: 4096 SNPs × 524288 samples, LUMI: 4096 SNPs × 32768 samples).

the programming and execution environment, as well as by the SLURM workload manager. On a MeluXina GPU-accelerated node, the MPI-aware CUDA-Episdet achieved a speedup of 3.98× when using all available A100 devices to process a dataset with 8192 SNPs and 32768 samples. Execution on a LUMI-G node reveals that linear speedups (7.96×) are also achieved on the CUDA-to-HIP ported HIP-Episdet code in relation to the amount of resources used. The processing of the same dataset through Crossarch-Episdet on MeluXina and LUMI also achieves close to linear speedups (4.01× on MeluXina and 7.79× on LUMI). When executing Tensor-Episdet relying on all GPUs available on a single GPU-accelerated node, the performance at processing datasets with 16384 SNPs increased by a factor of 4.01× on MeluXina and 7.89× on LUMI. Using multiple GPUs with Epi4Tensor resulted in a linear speedup (4.00× on MeluXina and 7.89× on LUMI) for processing datasets with 2048 SNPs, in comparison to using a single GPU.

As a result of the way processing of SNP sets is organized, both CUDA-Episdet and Crossarch-Episdet achieved very high scalability, even using static work distribution and scheduling. CUDA-Episdet achieved speedups of up to 31.10× using 32 MeluXina nodes (i.e. 128 GPUs) in relation to single-node execution (i.e. 4 GPUs), at processing 16384 SNPs and 32768 samples. The user account policies on LUMI allowed targeting a much higher amount of nodes, resulting in a 927.71× performance improvement processing 32768 SNPs on 1024 nodes (8192 GPUs). Crossarch-Episdet showcases even higher scalability, with 31.81× and 981.48× speedups on MeluXina and LUMI.

Processing a dataset with 32768 SNPs through Tensor-Episdet on MeluXina (LUMI) resulted in speedups of 1.99× (1.98×), 3.94× (3.88×), 7.72× (7.11×) and 14.18× (10.73×) on 2, 4, 8 and 16 nodes, respectively. Inter-node scalability is lower in LUMI due to the fact each GPU chip in an MI250X accelerator is controlled by an MPI worker process, and as a result there is – in relation to MeluXina – double the amount of MPI workers for the same amount of nodes.

Processing 4096 SNPs through Epi4Tensor resulted in close to linear speedups being achieved on 32 MeluXina nodes (31.72×) and 128 LUMI nodes (121.77×). We could not test with 64 GPU-accelerated Meluxina nodes due to the limitations imposed upon the user accounts, but all points to this code being able to efficiently utilize more nodes. The achieved scalability was in part the result of performing loop fusion to the two outerloops in the original implementation. For example, without it close to linear speedups were only observed up to 8 nodes on MeluXina.

The extended CUDA-Episdet and its HIP port achieved a performance of 342.61 and 15951.32 tera sets per second scaled to sample size on MeluXina and LUMI. Crossarch-Episdet achieved a scaled performance of 317.56 and 11182.24 on LUMI, 92.7% and 70.1% of the performance of CUDA/HIP-Episdet. Tensor-Episdet achieved a scaled performance of 5212.88 and 226.36 on MeluXina and LUMI. Epi4Tensor obtained a scaled performance of 14346.33 and 2456.96 on MeluXina and LUMI. The codes using tensor cores achieved significantly higher performance on MeluXina due to the 1-bit capabilities on the A100 GPUs. There is however still potential to improve performance on LUMI. For example, a Tensor-Episdet run using 8-bit integers instead of 16-bit floating point on 16 LUMI-G nodes resulted in 1.62× higher performance, despite the MI250X having the same peak throughput in both precisions. This code achieved a scaled performance of 8.79 tera SNP triplets processed per second on a single MI250X (two GPU chips). This represents a performance that is significantly higher than previously reported in literature for AMD GPUs.

Notice that case-control datasets with real genotypic data often have hundreds of thousands of SNPs and that scalability – and as a result, performance – is expected to further increase if processing datasets with more SNPs than those considered in this paper. Therefore, the presented scalability results should be seen as a lower-bound of what the codes extended to leverage the high parallel processing capabilities of the EuroHPC systems are capable of achieving.

6. Conclusions

This paper reported on the adaptation of four state-of-the-art bioinformatics codes focused on performing high-throughput high-order epistasis detection searches – three originally developed in CUDA and one in DPC++ / SYCL – in such a way that enables the efficient use of the massive computational resources of modern large-scale systems. As part of that endeavor, we have ported the CUDA codes to HIP in order to enable the use of AMD GPUs through the ROCm software stack, and introduced intra-node and inter-node parallelization on all codes through MPI. We also reported on the used GPU work scheduling schemes, which have been specialized for the different codes. Scalability tests performed on two EuroHPC supercomputers – LUMI and MeluXina – resulted in extremely high performance being attained, as a result of high scalability being achieved when targeting large amounts of GPU-accelerated nodes.

While exceptional performance can be achieved leveraging systems with high node counts, the implementations targeting AMD GPUs are at a significant disadvantage compared to their NVIDIA counterparts, which are the only ones that support 1-bit processing on specialized matrix units via fused AND+POPC or XOR+POPC operations. It is part of our current efforts to reduce the existing performance gap through better tailoring the solution to AMD microarchitectures, such as CDNA2, which is that of the MI250X GPUs on LUMI. This is highly relevant, as energy efficiency, which is particularly important when performing large-scale searches, is expected to be significantly better on the A100 GPUs for the codes utilizing tensor cores. Notice that most of our epistasis detection codes have originally been developed targeting NVIDIA GPUs, thus there is likely room for improvements on systems with AMD GPUs.

Ongoing and future work also includes adapting our solutions to newer GPU microarchitectures, such as AMD CDNA3 (e.g. MI300X) and NVIDIA Hoppper (e.g. H100). We expect improvements in the order of 3× through targeting the larger amount of resources on these devices and possibly even greater improvements through exploring

novel hardware features. Finally, it is also part of our plans to continue exploring the use of DPC++/SYCL through the oneAPI programming interface. In particular, the Joint Matrix Extension, which gives access to the matrix units in GPUs through a cross-compatible layer, might enable us to use SYCL as the primary programming model even when developing our first implementations of novel epistasis detection solutions, for which performance is a crucial aspect.

Acknowledgements

This work was supported by FCT (Fundação para a Ciência e a Tecnologia, Portugal) through the projects UIDB/50021/2020 and LISBOA2030-FEDER-00869000 (2023.18110.ICDT, VERSACOMP), and by the European EuroHPC project POP3 under the grant agreement No 101143931. We acknowledge EuroHPC Joint Undertaking for awarding us access to LUMI at CSC, Finland, and MeluXina at LuxProvide, Luxembourg, through the advanced computing projects with references EHPC-BEN-2023B01-002 and EHPC-DEV-2024D03-052.

References

[1] Cullell, N., et al., 2018. Pharmacogenetic studies with oral anticoagulants. Genome-wide association studies in vitamin K antagonist and direct oral anticoagulants. Oncotarget 9, 29238–29258. https://doi.org/10.18632/oncotarget.25579.
[2] EuroHPC Joint Undertaking, . Our supercomputers. https://eurohpc-ju.europa.eu/supercomputers/our-supercomputers_en.
[3] Forschungszentrum Jülich, . JUPITER - Exascale for Europe. https://www.fz-juelich.de/en/ias/jsc/jupiter.
[4] Hartwig, F.P., Entiauspe, L.G., Nunes, E.M., Rodrigues, F.M., Collares, T., Seixas, F.K., da Silveira, M.F., 2014. Evidence for an epistatic effect between TP53 R72P and MDM2 T309G SNPs in HIV infection: A cross-sectional study in women from south brazil. PLOS ONE 9, 1–11. https://doi.org/10.1371/journal.pone.0089489.
[5] Im, C., Ness, K.K., Kaste, S.C., Chemaitilly, W., Moon, W., Sapkota, Y., Brooke, R.J., Hudson, M.M., Robison, L.L., Yasui, Y., Wilson, C.L., 2018. Genome-wide search for higher order epistasis as modifiers of treatment effects on bone mineral density in childhood cancer survivors. European Journal of Human Genetics 26, 275–286. https://doi.org/10.1038/s41431-017-0050-x.
[6] Joubert, W., Weighill, D., Kainer, D., Climer, S., Justice, A., Fagnan, K., Jacobson, D., 2018. Attacking the opioid epidemic: Determining the epistatic and pleiotropic genetic architectures for chronic pain and opioid addiction, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, IEEE Press, Piscataway, NJ, USA. pp. 57:1–57:14. https://doi.org/10.1109/SC.2018.00060.
[7] LUMI, . LUMI supercomputer. https://www.lumi-supercomputer.eu/lumi_supercomputer/.
[8] LuxProvide, . MeluXina. https://www.luxprovide.lu/meluxina.
[9] Niel, C., et al., 2015. A survey about methods dedicated to epistasis detection. Frontiers in genetics 6, Article 285, 1–19. https://doi.org/10.3389/fgene.2015.00285.
[10] Nobre, R., Ilic, A., Santander-Jiménez, S., Sousa, L., 2020a. Exploring the binary precision capabilities of tensor cores for epistasis detection, in: 2020 International Parallel and Distributed Processing Symposium (IPDPS 2020). https://doi.org/10.1109/IPDPS47924.2020.00043.
[11] Nobre, R., Ilic, A., Santander-Jiménez, S., Sousa, L., 2021. Cross-architecture high-order exhaustive epistasis detection on CPU and GPU devices. https://devmesh.intel.com/projects/cross-architecture-high-order-exhaustive-epistasis-detection-on-cpu-and-gpu-devices.
[12] Nobre, R., Ilic, A., Santander-Jiménez, S., Sousa, L., 2023. Tensor-accelerated fourth-order epistasis detection on GPUs, in: Proceedings of the 51st International Conference on Parallel Processing, Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3545008.3545066.
[13] Nobre, R., Ilic, A., Santander-Jiménez, S., Sousa, L., 2021. Retargeting tensor accelerators for epistasis detection. IEEE Transactions on Parallel and Distributed Systems 32, 2160–2174. https://doi.org/10.1109/TPDS.2021.3060322.
[14] Nobre, R., Santander-Jiménez, S., Sousa, L., Ilic, A., 2020b. Accelerating 3-way epistasis detection with CPU+GPU processing, in: 23rd Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2020). https://doi.org/10.1007/978-3-030-63171-0_6.
[15] NVIDIA Corporation, . NVIDIA HGX platform. URL: https://www.nvidia.com/en-us/data-center/hgx.
[16] Ponte-Fernández, C., González-Domínguez, J., Martín, M.J., 2020. Fast search of third-order epistatic interactions on CPU and GPU clusters. The International Journal of High Performance Computing Applications 34, 20–29. https://doi.org/10.1177/1094342019852128.
[17] Pośpiech, E., et al., 2018. Towards broadening forensic DNA phenotyping beyond pigmentation: Improving the prediction of head hair shape from DNA. Forensic Science International: Genetics 37, 241–251. https://doi.org/10.1016/j.fsigen.2018.08.017.
[18] Quan, Y., et al., 2018. Facilitating anti-cancer combinatorial drug discovery by targeting epistatic disease genes. Molecules 23. https://doi.org/10.3390/molecules23040736.
[19] TOP500.org, . The List – November 2024. https://top500.org/lists/top500/2024/11/.
[20] Zuk, O., Hechter, E., Sunyaev, S.R., Lander, E.S., 2012. The mystery of missing heritability: Genetic interactions create phantom heritability. Proc Natl Acad Sci U S A 109, 1193–1198. https://doi.org/10.1073/pnas.1119675109.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 146–156

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Pleias 1.0: the First Ever Family of Language Models Trained on Fully Open Data

Pierre-Carl Langlais[a,*], Pavel Chizhov[a,b], Mattia Nee[a], Carlos Rosas Hinostroza[a], Matthieu Delsart[a], Irène Girard[a], Anastasia Stasenko[a], Ivan P. Yamshchikov[a,b]

[a] *PleIAs, Paris, France*
[b] *THWS, Würzburg, Germany*

Abstract

Linguistic diversity and strong generalization in foundation language models are typically achieved by training on trillions of data tokens with very large model parameter counts. However, most such training datasets include substantial amounts of copyright-protected or private data that is not explicitly published under the licence that is permissive for LLM training, raising legal and ethical concerns. We introduce **Pleias 1.0**, a family of comparatively small foundation language models (with at most 3 billion parameters) trained *exclusively on public domain or permissively licensed data*. We release the model weights and training code so that our results are fully auditable and reproducible. Furthermore, we fine-tune our models for the Retrieval-Augmented Generation (RAG) task and demonstrate that these models – despite their smaller size – can outperform competitors that have orders of magnitude more parameters on RAG evaluations. All models, data, and code are released under open licenses, offering a new standard for transparency and compliance.

Keywords: fully open source language models; multilingual small language models;

1. Introduction

Large language models (LLMs) are typically trained on massive web-scraped corpora comprising hundreds of billions to trillions of tokens [8]. Prominent datasets such as C4, RefinedWeb, and Dolma consist largely of general web text, which often contains significant portions of copyrighted content and personal information [14, 15]. This practice raises legal and ethical issues, especially under emerging regulations (e.g., the EU AI Act) that demand transparency in data provenance and respect for privacy. Studies have also highlighted quality problems in web-derived data, including the presence of noise and toxic content [7], as well as inconsistencies and lower quality in non-English

* Corresponding author: Pierre-Carl Langlais.
E-mail address: pierre-carl@pleias.fr

1877-0509
10.1016/j.procs.2025.08.241

text [16, 21]. Moreover, the availability of web data for training is diminishing as major platforms containing potential training data restrict scraping and reuse.

These challenges have spurred interest in developing LLMs using only legally unencumbered data. Early efforts like the Pile dataset assembled a large collection of public text sources for training LLMs (825 GiB, 22 datasets) [8]. More recently, the AI community has launched projects to build models on transparent and open data (e.g., the Semantic Scholar Open Research Corpus [14] and the RedPajama training corpus [26]. However, to date, no language model of competitive performance has been trained using *only* data that is entirely in the public domain or under a permissive license. This work fills this gap.

We present **Pleias 1.0**, the first family of language models trained exclusively on open data at scale. Pleias 1.0 models set a new standard for legal compliance and transparency: all training data comes from the public domain or permissively licensed sources, making these models compliant by design with data provenance requirements and mitigating copyright concerns. Our models are relatively small in size (350M, 1.2B, and 3B parameters) yet are *multilingual*, adhering to instruction in eight European languages, and *high-performing on specialized tasks*. In particular, we target applications in retrieval-augmented generation (RAG), where a language model must accurately incorporate retrieved evidence into its outputs. We fine-tuned two of the Pleias models for RAG and found that they outperform other open models with up to 10× more parameters on our evaluation benchmarks.

Pleias 1.0 models are trained on the second core contribution presented in this work – **Common Corpus**[1], a new open multilingual dataset totaling approximately 2 trillion tokens. This corpus draws on diverse domains such as literature and historical text, government and legal documents, scientific publications, software code, and more. We developed a suite of data processing tools to maximize quality, including specialized models to correct OCR artifacts and filters to remove toxic or low-quality content. The resulting dataset provides broad linguistic coverage and high-quality text suitable for training state-of-the-art LLMs without relying on any proprietary web crawl data.

We train Pleias models from scratch on Common Corpus using an efficient distributed training pipeline built entirely with open-source tools. Despite their modest size, the Pleias models demonstrate strong capabilities. Notably, they exhibit best-in-class language coverage and adherence among models in the sub-1B scale: for example, our 350M model (Pleias Pico) fluently supports at least eight languages (English, French, Spanish, German, Italian, Dutch, Latin, Portuguese) and is less prone to unintended language switching, thanks to the balance of multiple languages in the data and a custom tokenizer design [5]. We also report that our training process was energy-efficient, yielding a significantly lower carbon footprint compared to other models of similar scale.

In summary, our contributions include:

- the creation of Common Corpus, a 2-trillion-token open multilingual dataset for LLM pretraining;
- the development of the Pleias 1.0 family of LLMs (350M, 1.2B, 3B parameters) trained entirely on this open corpus;
- novel training techniques and data processing pipelines to ensure data quality and efficient training;
- fine-tuning and evaluation showing that small, specialized models can match or beat much larger models on RAG tasks;
- the open release of all models, data, and code under permissive licenses to foster transparent research and practical adoption.

We hope this work paves the way for more ethical and accessible large language models. In the next section, we describe Pleias 1.0 pre-training dataset – Common Corpus – that we have collected, documented, and published for further use by the community. Section 3 describes the architecture of Pleias 1.0 and provides details on the training, including the carbon footprint of the models. Section 4 provides evaluations of the models. Section 5 is dedicated to the discussion of the difficulties we encounter and possible directions for further work.

[1] https://huggingface.co/datasets/PleIAs/common_corpus

2. Common Corpus: An Open Multilingual Dataset

2.1. Data Sources and Composition

Common Corpus is a new large-scale dataset composed entirely of text from sources that are either in the public domain or released under an open license. The corpus spans a wide range of domains and languages, reflecting our goal of broad coverage for a European multilingual foundation model. Major components of Common Corpus include:

- **Cultural Heritage Texts:** A collection of literature and historical texts that are out of copyright, including books, newspapers, and manuscripts from digital libraries and archives. For example, we incorporate texts from Project Gutenberg and national libraries (where copyright has expired) to represent classic literature and historical documents.
- **Legal and Government Documents:** A large body of legal texts and administrative documents released as open data. This includes legislative texts, court decisions, and government publications from jurisdictions that offer these under permissive licenses. For instance, we include portions of the EU's EuroParl and EurLex corpora (transcriptions of parliamentary sessions and legal directives) [17], and other public domain legal resources.
- **Scientific Publications:** Research papers and academic content available under open access terms. We draw on sources such as arXiv and the Semantic Scholar Open Research Corpus [14] for scientific and scholarly text in multiple languages. This component ensures the corpus contains high-quality formal writing and technical content.
- **Open Web Content:** Web text that is explicitly under open licenses. Unlike many LLM training sets that scrape random web pages, we restrict this category to permissively licensed sites and contributions. The largest portion here is Wikipedia (all languages), which is available under the CC BY-SA license. We also include other wiki-style or collaboratively created resources, and a filtered subset of Common Crawl data focusing only on documents with clear open licenses (e.g., certain government or news websites with open terms).
- **Code and Software Documentation:** Source code and software documentation released under open-source licenses. For this, we incorporated datasets like CodeSearchNet [11] (which collects code from GitHub under permissive licenses) and a filtered dump of Stack Exchange (which is CC BY-SA licensed Q&A content, including many programming discussions). This component introduces multilingual programming languages and technical problem-solving texts.
- **Other Niche Domains:** We also included various smaller open datasets to increase diversity, such as public domain subtitles, transcriptions of public speeches, ancient language texts (e.g., Latin corpora), and so on. All data sources were vetted to ensure compliance with our open license criteria.

Table 1 summarizes the language distribution of Common Corpus. The dataset is predominantly English (due to the abundance of English open content), but about 35–40% of the tokens are in languages other than English, making it one of the most multilingual open corpora of this scale. Notably, major European languages like French and German each contribute substantial portions, and the corpus includes content in dozens of languages overall. We prioritized including as many high-quality non-English sources as possible to improve the model's multilingual capabilities.

Table 1. Top 10 languages in Common Corpus by proportion of tokens.

Language	Percent of Tokens	Language	Percent of Tokens
English	64.4%	Dutch	1.45%
French	14.8%	Italian	1.26%
German	6.68%	Polish	0.67%
Spanish	2.62%	Greek	0.64%
Latin	2.03%	Portuguese	0.53%

Since Common Corpus is not the core focus of this paper, we refer the reader to the corresponding technical report for more details[2].

2.2. Data Processing and Quality Filtering

All data in Common Corpus undergoes an extensive processing pipeline to maximize quality and cleanliness. We develop several custom tools as part of this pipeline:

- **OCR Correction:** A significant portion of our cultural heritage data and some government documents come from scanned books or PDFs, which contain OCR (Optical Character Recognition) errors (e.g., incorrect characters, broken text segments). We trained a specialized language model called *OCRonos*[3] to automatically correct such digitization artifacts. OCRonos can identify common OCR issues like character substitutions, word splits/merges, and formatting errors, and output a cleaned text that remains faithful to the original. By applying OCRonos to all texts that were derived from image scans, we dramatically improved the readability and accuracy of these portions of the corpus. An important aspect is that OCRonos is multilingual and was itself trained on a mix of diverse OCR-ed texts, so it generalizes well across languages.
- **Text Segmentation:** For certain document collections (particularly historical archives), raw text extractions can be poorly segmented (e.g., multiple articles concatenated, or text order scrambled). We utilized another internal tool, *Segmentext*[4], a specialized model for text segmentation. Segmentext is designed to handle broken or unstructured text and determine proper segment boundaries even in the presence of noise. This helped in splitting long concatenated texts into coherent segments (e.g., separating individual articles in a newspaper scan batch).
- **Quality Filtering:** We applied systematic filtering to remove low-quality content. This included rule-based and model-based filters to eliminate gibberish or extremely repetitive texts, as well as to down-sample overly similar or duplicated content. We also filtered out content that was too short or too long, since extremely short texts are not very useful for language modeling, and extremely long texts (exceeding our context length significantly) might pose difficulties in training.
- **Toxicity and Bias Mitigation:** Although our data sources are generally cleaner than an uncurated web crawl, there still exists the possibility of toxic language, hate speech, or other undesirable content in a corpus of this size (for example, historical texts can contain archaic prejudiced language). To address this, we integrated a *toxicity classifier* in the pipeline to identify and remove highly toxic or hateful segments. This classifier was developed with a combination of keyword heuristics and a pre-trained model for toxicity detection. By filtering out flagged content, we aimed to reduce the incidence of harmful outputs from the trained model. We report further details about toxicity classification and filtering in a separate work [3].
- **Synthetic Data Augmentation:** In addition to real-world text, we augmented Common Corpus with a relatively small portion of synthetic text (approximately 30 billion tokens, ~1.5% of the total). This synthetic data was generated to provide examples of tasks or domains underrepresented in the corpus (such as conversational QA format, which can help with downstream fine-tuning). However, we were cautious to limit the amount of synthetic data. Recent studies have observed that excessive use of synthetic data can lead to performance plateaus or degradation in LLM training. In fact, Dohmatob et al. [6] show that mixing even a modest percentage of real data with predominantly synthetic data is crucial to maintain the benefits predicted by scaling laws. Guided by these findings, we only supplement with a small synthetic set to avoid dominating the training distribution.

After processing and filtering, the final Common Corpus is stored in a tokenized binary format for efficient reading during model training (see Section 3.2). Overall, our data pipeline significantly improves the signal-to-noise ratio of the corpus. We emphasize that while using only public data addresses legal transparency, it does not automatically

[2] https://huggingface.co/datasets/PleIAs/common_corpus
[3] https://huggingface.co/PleIAs/OCRonos
[4] https://huggingface.co/PleIAs/Segmentext

guarantee ethical perfection of the data (e.g., public domain texts can still contain biases or outdated viewpoints). Nonetheless, by carefully curating sources and filtering, we aim to produce a dataset that is not only legally clean but also of high quality for modeling.

3. Model Architecture and Training

3.1. Model Design

We train three model variants as parts of the Pleias 1.0 family, corresponding to parameter counts of roughly 350 million, 1.2 billion, and 3 billion. All are decoder-only Transformer language models with architectures inspired by recent open LLMs like LLaMA [24] and GPT-NeoX [4]. Table 2 summarizes the key architecture hyperparameters for each model size.

Table 2. Pleias 1.0 model architecture configurations. All models use decoder-only Transformer layers with SwiGLU activation and rotary positional embeddings.

Model	#Layers	Model Dim	FFN Dim	#Heads	KV Heads	Context Length
Pleias 1.0 350M	26	1024	2560	16	8	2048
Pleias 1.0 1.2B	22	2048	6144	32	8	2048
Pleias 1.0 3B	22	3072	12288	32	8	4096

All Pleias models use pre-normalization (LayerNorm at the start of each block) and employ the SwiGLU gated activation unit in feed-forward layers, following Shazeer [20]. We use Rotary Positional Embeddings (RoPE) [22] for encoding position within the self-attention mechanism, with a base period of 10,000. RoPE allows extrapolation to longer contexts, which we leverage by training the 3B model with a 4096 token context window (doubling the standard 2048 of the smaller models).

3.2. Tokenizer

We build a custom Byte-Pair Encoding (BPE) tokenizer for the Pleias models to accommodate the multilingual nature of Common Corpus. We train a tokenizer with 65,536 unique tokens on a representative sample of the corpus. We also use special tokens for padding, unknown words, and text beginnings and endings. Unlike some standard tokenizer configurations (e.g., the original LLaMA tokenizer), we remove certain English-centric preprocessing rules, such as those that split on specific Unicode punctuation or contract apostrophes, to treat all languages more uniformly. The resulting tokenizer provides more equitable subword segmentations across languages, ensuring that, for example, French or German text is not tokenized into disproportionately longer sequences than English.

We find that using a fresh tokenizer was preferable to reusing an existing one from a model like GPT-2 or LLaMA, because recycling tokenizers can introduce hidden biases or inefficiencies for languages not well-represented in the original tokenizer's training data [2]. By training on our corpus, the Pleias tokenizer achieves an average compression ratio (bytes to tokens) that is similar across our top languages.

For the RAG fine-tuned versions of Pleias (described in Section 3.5), we extend the tokenizer with a handful of special tokens used to delimit parts of the RAG prompt/response format. Specifically, tokens marking the start and end of a user query, the start and end of each retrieved source text, the beginning of the model's answer, etc., were added. These tokens enable the model to be aware of the structure in a RAG setting (e.g., keeping track of which text is a source and which is the answer) without ambiguity.

3.3. Training Setup and Hyperparameters

We train each model from scratch on the Common Corpus using an end-to-end open-source toolchain. Training was executed with the HuggingFace *Nanotron* library[5]. We first convert the entire corpus into packed tokenized sequences

[5] https://github.com/huggingface/nanotron

using the *Nanoset* data preparation tool[6]. This allows us to stream pre-tokenized batches during training, avoiding on-the-fly tokenization overhead.

For optimization, we use AdamW, and each training run used a linear warmup followed by a cosine decay learning rate schedule. The warmup phase lasts for 5–8% of total training steps (we used a slightly longer warmup for the larger models, based on early experiments). Table 3 details the training hyperparameters and parallelism setup for each model.

Table 3. Training configuration for Pleias 1.0 models.

Model	DP	TP	PP	Micro-batch	Gradient Accumulation	Total Batch Size	Peak LR → End LR
Pleias Pico (350M)	64	1	1	8	2	~2M	$3 \times 10^{-3} \rightarrow 3 \times 10^{-5}$
Pleias Nano (1.2B)	192	1	1	8	2	~6M	$1 \times 10^{-3} \rightarrow 1 \times 10^{-5}$
Pleias-3B Base	48	4	1	8	8	~12M	$5 \times 10^{-4} \rightarrow 5 \times 10^{-6}$

We perform training on two computing infrastructures. The 350M and 3B models are trained on the Jean Zay supercomputer in France (using H100 GPUs, under a Grand Challenge compute grant), while the 1.2B model was trained using *TractoAI*'s cloud platform[7]. In the latter case, we adapt our pipeline to TractoAI's serverless infrastructure backed by YTsaurus. This involved converting the pre-tokenized data into TractoAI table format and modifying Nanotron to read from this distributed storage, as well as custom handling of checkpoint saves. These engineering adaptations enable fault-tolerant, large-scale training on the cloud without proprietary software.

Each model is trained for a number of epochs on Common Corpus, with a curriculum between a "base" portion and a further "clean" portion of data. Specifically, for the 3B model, we trained for 2 epochs on the full Common Corpus (approximately 2 trillion tokens total) and then 1 additional epoch on a more aggressively filtered subset of the corpus (about 1 trillion tokens of the highest quality segments). This two-stage training (which we term an *annealing corpus*) is intended to first expose the model to very diverse data, then fine-tune it on cleaner data for the final phase. For the 1.2B model, we perform 1 epoch on the full corpus and 2 epochs on the clean subset. For the smallest 350M model, we find it sufficient to train on just the clean subset for 1 epoch (given computational constraints and the model's capacity). In total tokens seen, the 1.2B and 3B models each process on the order of 3–4 trillion tokens when counting multiple epochs, which is a substantial training budget for models of this size.

3.4. Carbon Footprint and Efficiency

One motivation for developing smaller, targeted models is to reduce the environmental impact of training and deploying LLMs. We calculate the carbon emissions from training our models using an online emissions calculator [12] that accounts for the energy draw of the GPU hardware and the carbon intensity of the power grid. Table 4 reports the training compute used and estimated emissions for each Pleias model, and compares them to other open models of similar scale.

Table 4. Training compute and estimated carbon emissions for Pleias 1.0 models, compared to other models. OpenELM refers to a model baseline of comparable size, and LLaMA refers to the hypothetical LLaMA training emissions scaled to a similar parameter count (based on publicly reported values). Emissions are in tonnes of CO_2 equivalent (tCO_2eq).

Model Size	#GPUs (type)	Time (days)	Pleias	Similar OpenELM	Similar LLaMA
~350M	64 (H100)	1.92	0.5 tCO_2	1.5 tCO_2	–
~1.2B	192 (H100)	5.0	4.0 tCO_2	5.5 tCO_2	107 tCO_2
~3B	192 (H100)	20.0	16 tCO_2	7.0 tCO_2	133 tCO_2

We observe that Pleias 1.0 models required an order of magnitude less CO_2 emissions to train compared to larger models like Meta's LLaMA (which in 7B-13B scale has been estimated at tens to hundreds of tons). Even against contemporaries in the small-model regime, Pleias models are very efficient. For instance, our 1.2B model consumed

[6] https://github.com/huggingface/nanotron/blob/main/docs/nanoset.md
[7] https://tracto.ai/

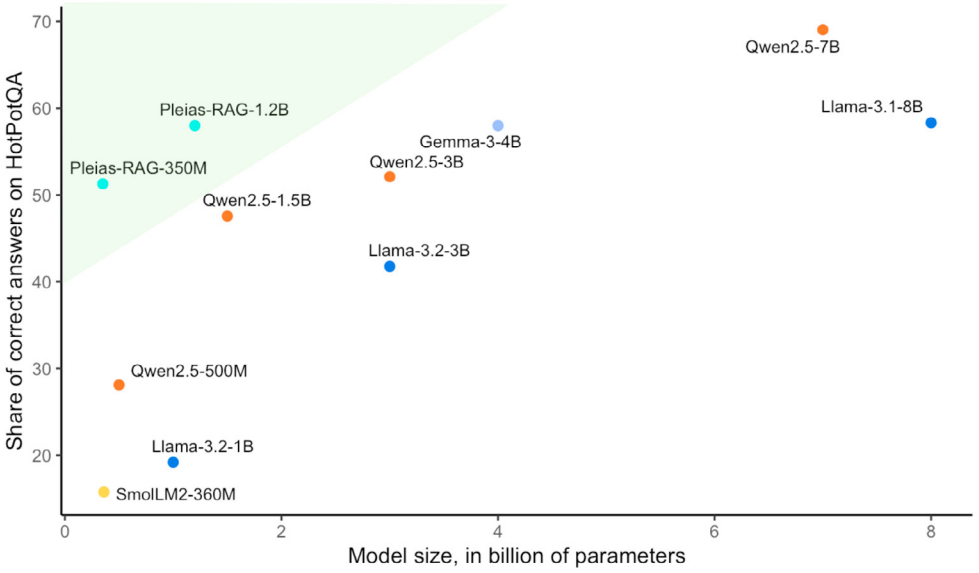


Fig. 1. Loss of RAG-fine-tuned Pleias 1.0 on non-English on HotPotQA [27]. Pleias 1.0 is in the Pareto-optimal part of the plot. For further details on the RAG-fine-tuned Pleias 1.0 see technical report [13].

about 4 tCO_2, which is about 30% lower than a comparable 1B-scale open model reported by Mehta et al. [18]. The efficiency is achieved through a combination of factors: smaller model size, a well-optimized training pipeline (high GPU utilization, large batch sizes), and the use of the hardware of the latest generation (NVIDIA H100), which offers better performance per watt.

The relatively low carbon footprint means that reproducing or fine-tuning these models is within reach of academic labs or smaller organizations that might have sustainability constraints. By releasing our training pipeline and configurations, we encourage others to build on this work in an energy-conscious way.

4. Evaluation

We evaluate the Pleias 1.0 models on both general language tasks and specialized retrieval-augmented generation tasks. Our focus is on demonstrating the capabilities of the RAG-tuned models relative to other open models of similar or larger scale. In addition, we provide some qualitative observations on eight European languages.

4.1. Fine-Tuning for Retrieval-Augmented Generation and RAG Evaluations

After pretraining the base models, we fine-tune two of them (350M and 1.2B) for retrieval-augmented generation, creating **Pleias 350M RAG** and **Pleias 1.2B RAG**, respectively. This fine-tuning is effectively an additional phase of language modeling training on a specialized corpus, rather than a typical small-sample supervised fine-tune. We construct a synthetic RAG training corpus of about 10 billion tokens comprising queries, retrieved passages, and annotated answers in separate regimes for trivial questions and questions that required structured reasoning. The content for this is derived from our Common Corpus knowledge sources (so as to remain in the realm of open data). The fine-tuning procedure continues training from the final checkpoint of the base model, using the same optimizer settings but at a lower learning rate (we reused the last learning rate from pretraining as the constant rate during RAG tuning). We detail the work on the RAG-fine-tuned models in the corresponding technical report [13].

The evaluations are performed on three question answering benchmarks: HotpotQA [27], 2WikiMultiHopQA [10], and MuSiQue [25], and compared them to the instruction-tuned models from LLaMA 3.1 [9], Qwen 2.5 [19], SmolLM 2 [1], and Gemma 3 [23] model families. To summarize the results, our models performed best compared to the

models of their size, and sometimes outperformed way larger ones. For example, PleIAs-RAG-350M outperformed all the tested counterparts, including the Qwen 2.5 7B, a model 20 times larger.

To highlight the multilingual capabilities of our models, we also translate the benchmarks to the main European languages. When evaluating the same set of models on the translated questions, we found that the performance of Pleias-RAG models remains high, with less than 5% of drop for the 1.2B model, while for the competitors, the general drop in performance ranges from 10% for Qwen 2.5 7B to more than 30% for SmolLM2 360M. This suggests that our models are more reliable in a multilingual setting, especially for their size range. This advantage can be largely attributed to the rich multilingual pre-training.

4.2. Multilingual Capabilities

Beyond English, the Pleias models demonstrate strong multilingual capabilities for the size of the models. Thanks to Common Corpus, which contains a rich variety of languages, our models can understand and generate several languages with high competency. As noted earlier, Pleias 1.0 350M is the first model of its size to "fully support" a range of major languages (at least eight). In practical terms, this means it can answer questions or hold a conversation in those languages without defaulting back to English or producing incoherent text.

We perform some supplementary tests to verify language adherence. For example, we asked each model a question in French and observed whether the answer stayed in French or switched to English. Pleias 350M and Pleias 1.2B reliably answered in French, whereas some baseline models like LLaMA-2 7B (chat) often responded in English to a French query, indicating a bias toward English. This adherence is important for a user experience perspective: a multilingual model should ideally reply in the language of the query unless instructed otherwise.

The custom tokenizer and balanced training data likely contribute to this behavior. Since our tokenizer does not over-fragment non-English text, the models do not see non-English inputs as overly rare or token-intensive. Combined with the fact that 35% of training tokens were non-English (with some languages like French making up nearly 15%), the models had sufficient exposure to learn those languages well. We also note that including Latin in the top languages is unusual for LLMs, which, however, is the case for Common Corpus because many public domain books and documents from history are in Latin; Pleias models gained some ability to read and write basic Latin, which might be beneficial for niche scholarly applications.

5. Encountered Problems

During the training of Pleias models, we encounter several problems that require changes and adaptations. In this section, we enumerate several of the most typical ones, which might help model developers in the future.

Cluster-framework compatibility. There are several open frameworks for distributed training of large models available, and their documentation usually does not include all the possible errors that can occur. The main sources of such errors are deep dependency libraries that are getting constantly updated and cluster setup difficulties that were not accounted for in the base documentation. To solve these problems, we consulted with the communities for the frameworks that were previously used in similar conditions and ran several tests with the main ones. In addition, for the smaller error, we used the search through StackOverflow[8] and LLM chatbots, mainly Claude[9].

Learning rate scheduling bug. When we run our training, Nanotron was a framework in the early stages of active development, so many new features were appearing, and together with them came the bugs. One of the bugs that affected our process was learning rate scheduling initialization, which was set up incorrectly when restarting from a checkpoint. Since we were doing regular audits of the training process, we were able to identify this issue early and submit it to the developers[10]. Since the framework was being rapidly developed and popular in the community, the issue was quickly identified, and we could integrate a solution to our local version of Nanotron before the actual integration of this fix into the framework.

[8] https://stackoverflow.com/questions
[9] https://claude.ai/
[10] https://github.com/huggingface/nanotron/issues/233

Cluster availability. We train the models on the Jean Zay supercomputer[11], which is used by plenty of other research groups, and, though having a considerable amount of allocated resources, might be overloaded. To overcome this difficulty, we started our training in advance and always chose the proper job timeout so that we do not clutter the Slurm[12] queue and do not get demoted by the scheduler. There are also two factors that allow us to predict higher cluster availability in the foreseeable future: right before and after the cluster maintenance, which happens quite regularly, and during the holiday periods.

Job timeout limit. To avoid long resource outages due to stagnating jobs, the Jean Zay cluster has an upper limit of 20 hours set for every job, so we cannot expect our jobs to run longer than this. To overcome this issue, we leveraged Slurm's arrayed jobs, which enabled us to set up a sequence of jobs running for the maximum allowed time. Close to the end of each job, we expected a checkpoint to be saved, so that the next job could start from this checkpoint. We also added a command to our Slurm scripts that would change the parameters of the current training run configuration, accounting for the job that is being started.

6. Conclusion

We introduce Pleias 1.0, a suite of small-scale language models trained entirely on open data, and demonstrate that these models can achieve outstanding results on specialized tasks like retrieval-augmented generation. By leveraging the Common Corpus – a massive, multilingual, public domain dataset – and carefully designing our training pipeline, we addressed key legal and ethical concerns (data transparency and licensing) without sacrificing performance. In fact, Pleias models match or surpass much larger models in our evaluations, all while being more efficient to train and run.

Our work underscores the importance of **data quality and domain-specific optimization**. Instead of blindly scaling model size or using indiscriminate web crawls, we focused on curating a training set that is legally clean and rich in the kinds of information our models need for high-value applications (like answering factual questions with evidence). The success of Pleias 1.2B and 350M in RAG tasks suggests that targeted smaller models can be viable alternatives to giant general-purpose models, especially for use cases where computational resources or latency are constraints.

We believe Pleias 1.0 sets a precedent for *fully open LLM development*: every component of our project – the data, the training code, the model weights, and even the evaluation data – is released openly. This enables anyone to reproduce or build upon our results, and provides a foundation for future research into compliant and transparent AI. Moreover, our approach is directly aligned with forthcoming regulatory requirements (such as the EU AI Act) that will likely necessitate knowing and disclosing exactly what data an AI model was trained on. Pleias 1.0 proves that such compliance is achievable today.

In future work, we plan to extend this paradigm in several directions. One is scaling up the model size while remaining within the open-data regime, to see if a 7B or 13B model trained on Common Corpus can further close the gap with the largest proprietary models. Another direction is expanding the multilingual coverage beyond mostly Indo-European languages, by incorporating more open data from diverse languages (e.g., expanding into Asian and African languages, where obtaining public domain text might require new collection efforts). We also aim to explore other task-specific fine-tunes of the base Pleias models, such as instruction tuning for general helpfulness or specialized domains like legal reasoning, again using only permissively licensed instruction data.

To the community, we release Pleias 1.0 models under the Apache 2.0 license, and the Common Corpus dataset under a CC BY license (with components in the public domain). We hope these resources will be useful for both researchers and practitioners. The 350M Pleias model, in particular, is light enough to run on consumer hardware, opening the door for truly local and private deployment of assistive LLMs that users can trust and verify. We publicly release the models and training data and invite the open science community to further contribute to creative applications of our models, as well as collaborate on the next iterations of open-data language models.

[11] http://www.idris.fr/eng/jean-zay/jean-zay-presentation-eng.html
[12] https://slurm.schedmd.com/documentation.html

Acknowledgements

The authors of this work acknowledge the HPC resource allocation by the European High Performance Computing Joint Undertaking (EuroHPC JU) on the Jean Zay cluster (compute grant #GC011015451).

References

[1] Allal, L.B., Lozhkov, A., Bakouch, E., Blázquez, G.M., Penedo, G., Tunstall, L., Marafioti, A., Kydlíček, H., Lajarín, A.P., Srivastav, V., Lochner, J., Fahlgren, C., Nguyen, X.S., Fourrier, C., Burtenshaw, B., Larcher, H., Zhao, H., Zakka, C., Morlon, M., Raffel, C., von Werra, L., Wolf, T., 2025. Smollm2: When smol goes big – data-centric training of a small language model. URL: https://arxiv.org/abs/2502.02737, arXiv:2502.02737.

[2] Arnett, C., 2023. Dangers of tokenizer recycling. Hugging Face Blog. https://huggingface.co/blog/catherinearnett/dangers-of-tokenizer-recycling.

[3] Arnett, C., Jones, E., Yamshchikov, I.P., Langlais, P.C., 2024. Toxicity of the commons: Curating open-source pre-training data. URL: https://arxiv.org/abs/2410.22587, arXiv:2410.22587.

[4] Black, S., et al., 2022. Gpt-neox-20b: An open-source autoregressive language model. arXiv preprint arXiv:2204.06745 .

[5] Chizhov, P., Arnett, C., Korotkova, E., Yamshchikov, I.P., 2024. BPE gets picky: Efficient vocabulary refinement during tokenizer training, in: Al-Onaizan, Y., Bansal, M., Chen, Y.N. (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA. pp. 16587–16604. URL: https://aclanthology.org/2024.emnlp-main.925/, doi:10.18653/v1/2024.emnlp-main.925.

[6] Dohmatob, E., Feng, Y., Kempe, J., 2024. Combating mode collapse in rlhf: On the connection between synthetic data and grokking in llms. arXiv preprint arXiv:2402.00159 .

[7] Elazar, Y., Bhagia, A., Magnusson, I., Ravichander, A., et al., 2023. What's in my big data? annotating and analyzing large web-crawled datasets. arXiv preprint arXiv:2308.00086 .

[8] Gao, L., Biderman, S., et al., 2021. The Pile: An 800gb dataset of diverse text for language modeling, in: Proceedings of NeurIPS Datasets and Benchmarks.

[9] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C.C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., et al., 2024. The llama 3 herd of models. URL: https://arxiv.org/abs/2407.21783, arXiv:2407.21783.

[10] Ho, X., Duong Nguyen, A.K., Sugawara, S., Aizawa, A., 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps, in: Scott, D., Bel, N., Zong, C. (Eds.), Proceedings of the 28th International Conference on Computational Linguistics, International Committee on Computational Linguistics, Barcelona, Spain (Online). pp. 6609–6625. URL: https://aclanthology.org/2020.coling-main.580/, doi:10.18653/v1/2020.coling-main.580.

[11] Husain, K., et al., 2019. Codesearchnet challenge: Evaluating the state of semantic code search. arXiv preprint arXiv:1909.09436 .

[12] Lacoste, A., Luccioni, S., Schmidt, T., Dandres, T., 2019. Quantifying the carbon emissions of machine learning. arXiv preprint arXiv:1910.09700 .

[13] Langlais, P.C., Chizhov, P., Nee, M., Hinostroza, C.R., Delsart, M., Girard, I., Hicheur, O., Stasenko, A., Yamshchikov, I.P., 2025. Even small reasoners should quote their sources: Introducing the pleias-rag model family. URL: https://arxiv.org/abs/2504.18225, arXiv:2504.18225.

[14] Lo, K., Wang, L.L., et al., 2020. S2ORC: The semantic scholar open research corpus, in: Proceedings of ACL.

[15] Longpre, S., Mahari, R., Muennighoff, N., Kabbara, J., Khazatsky, A., et al., 2023. The data provenance initiative: A large scale audit of dataset licensing & attribution in AI. arXiv preprint arXiv:2310.16787 .

[16] Luccioni, S., Viviano, J., 2021. What's in the box? an analysis of undesirable content in the common crawl corpus. arXiv preprint arXiv:2105.02732 .

[17] Martins, P.H., Fernandes, P., Alves, J., et al., 2024. EuroLLM: Multilingual language models for europe. arXiv preprint arXiv:2409.16235 .

[18] Mehta, S., Sekhavat, M., Cao, Q., Horton, M., Jin, Y., Sun, F., Mirzadeh, I., Najibikohnehshahri, M., Belenko, D., Zatloukal, P., Rastegari, M., 2024. Openelm: An efficient language model family with open training and inference framework, in: ICML Workshop. URL: https://arxiv.org/abs/2404.14619.

[19] Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., Qiu, Z., 2025. Qwen2.5 technical report. URL: https://arxiv.org/abs/2412.15115, arXiv:2412.15115.

[20] Shazeer, N., 2020. Glu variants improve transformer. arXiv preprint arXiv:2002.05202 .

[21] Sikasote, C., Machovac, D., Kliegr, T., 2021. Quality at a glance: An audit of web-crawled multilingual datasets. arXiv preprint arXiv:2103.12028 .

[22] Su, J., Shen, T., Zhu, Z., et al., 2021. Roformer: Enhanced transformer with rotary position embedding, in: Advances in Neural Information Processing Systems (NeurIPS).

[23] Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., bastien Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A.M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A.S., Das, A., Bapna, A., et al., 2025. Gemma 3 technical report. URL: https://arxiv.org/abs/2503.19786, arXiv:2503.19786.

[24] Touvron, H., et al., 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 .

[25] Trivedi, H., Balasubramanian, N., Khot, T., Sabharwal, A., 2022. MuSiQue: Multihop questions via single-hop question composition. Transactions of the Association for Computational Linguistics 10, 539–554. URL: https://aclanthology.org/2022.tacl-1.31/, doi:10.1162/tacl_a_00475.

[26] Weber, M., Fu, D.Y., Anthony, Q., Oren, Y., Adams, S., Alexandrov, A., Lyu, X., Nguyen, H., Yao, X., Adams, V., Athiwaratkun, B., Chalamala, R., Chen, K., Ryabinin, M., Dao, T., Liang, P., Ré, C., Rish, I., Zhang, C., 2024. Redpajama: an open dataset for training large language models. NeurIPS Datasets and Benchmarks Track .

[27] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W.W., Salakhutdinov, R., Manning, C.D., 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. URL: https://arxiv.org/abs/1809.09600, arXiv:1809.09600.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 157–164

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Automatic detection of agricultural field boundaries and seeded acres using super-resolution Satellite Earth Observation data and deep learning

Nils Helset[a], Konstantin Varik[b] and Alex Melnitchouck[c*]

a DigiFarm AS, Holsetgata 22, 2317 Hamar, Norway

b DigiFarm AS, Holsetgata 22, 2317 Hamar, Norway

c DigiFarm AS, Holsetgata 22, 2317 Hamar, Norway

Abstract:

DigiFarm is a Norwegian agricultural technology startup, established in 2019, focused on developing deep neural network (AI) models based on super-resolved Satellite Earth Observation (SatEO) data. The objective is to automatically detect agricultural field boundaries and planted areas from in-season vegetation. This paper presents the conceptual and technical overview of our work conducted under a GPU allocation grant on the LUMI supercomputer (Proposal ID: EHPC-AI-2024A01-080), awarded through the EuroHPC "AI and Data-Intensive Applications Access call". We detail the development of a novel deep neural network that intersects deep learning and computer vision to delineate agricultural fields. The model, which includes 12 land cover classification outputs, achieves an Intersection over Union (IoU) of over 0.94. This automated, low-cost, and scalable approach provides a significant improvement over existing manual digitization methods, which are time-consuming, resource-intensive, and costly. Our solution reduces the time-to-result from months to hours and increases the accuracy of detected field boundaries by 12-15% compared to existing Cadastral data. This represents a breakthrough for the agricultural industry, which has historically struggled with fragmentation and a lack of digital adoption.

Nils Helset*, Konstantin Varik and Alex Melnitchouck, DigiFarm AS, Norway (*Corresponding author: nils@digifarm.io)

Keywords: Super-resolution, Sentinel-2, Deep Learning, Field Boundary Detection, AI in Agriculture, LUMI, EuroHPC, IoU Accuracy

10.1016/j.procs.2025.08.242

1. Introduction:

Agriculture and food production are vital parts of the global economy. By some estimates, the world population is expected to grow from 7.2 billion today to 10 billion by 2050, which may require a 70% growth in global food supplies from current levels [1]. In parallel, crop production is predicted to drop by 2% for every degree of global warming, presenting a widely recognized need for farmers to increase food production efficiency and protect farmlands [2].

Currently, 40% of the world's agricultural fields are over-fertilized. Simultaneously, farmers are losing 15-20% in yield potential through inadequate input application [3]. This sub-optimal approach often involves applying a uniform amount of seed, fertilizer, and crop protection across an entire field. As in-field yield variability can be as high as 300%, farmers are not effectively leveraging management zones to optimize inputs, a practice commonly referred to as Variable Rate Technology (VRT). These input costs represent over 50% of a farm's total costs, highlighting the potential for technological innovation to increase efficiency and sustainability for both large industrial farms and smallholder farms (which represent 83% of the global agricultural fields).

A wide range of precision farming solutions have been proposed to address these challenges. These methods rely on digital imaging data from Earth Observation (EO) satellites to analyze crop conditions and support optimal operational decisions. The success of these methods depends on precise, georeferenced boundaries of individual fields to enable accurate analysis of satellite imagery and other data, such as soil information and yield monitor data.

In addition to analysis, farm management and operations require precise information about the farmed area. This information is crucial for calculating crop inputs, taxation, crop insurance, farming subsidies, and logistics. For instance, creating a VRT map for seed or fertilizer requires an accurate field boundary to be programmed into the machinery. Similarly, farming subsidies are calculated based on seeded acres, meaning the specific area where crops are growing, which is distinct from the legal field boundary. This distinction is also critical for crop insurance adjusters assessing potential damage.

However, the precise and up-to-date field boundaries needed for these applications are often unavailable. Existing large-scale field boundary data is frequently outdated and inaccurate. This data is typically managed by national agencies, such as the Norwegian Institute of Bioeconomy Research (NIBIO), whose data dates to the 1990s, or the EU's Land Parcel Identification System (LPIS). In the United States, the Common Land Units (CLUs) database was created in 2008 and has not been significantly updated since. For example, there are approximately 34.6 million field boundaries in the EU regions and 32 million in the US; manually redrawing these boundaries is prohibitively expensive and time-consuming. This reliance on outdated data creates a significant bottleneck for the entire agricultural value chain.

2. The Challenge of Scalable and Accurate Field Delineation

The core of our project, funded by EuroHPC, was to overcome the limitations of existing field boundary data and manual delineation processes. DigiFarm has spent the last four years developing a baseline model that achieves 8-10% higher accuracy than existing solutions. The project on LUMI was designed to advance this work by addressing two primary challenges: the high cost of commercial GPU resources and the long training times required for complex models.

- Unsustainable R&D Costs: Commercial cloud GPUs have been too expensive for heavy R&D workloads. The average price for an NVIDIA A100 single card unit across major cloud providers was $2.72 per hour (Table 1). As our models grow in complexity, requiring more data and hyperparameter tuning, these costs become prohibitive for an SME. Access to subsidized, cutting-edge infrastructure like LUMI is critical for

developing a competitive advantage.

- Long Training and Commercialization Cycles: Training advanced deep neural network models is a time-intensive process. Using conventional cloud infrastructure, this could take over six months, creating a challenging investment justification and delaying commercialization. Furthermore, the availability of high-performance GPUs in commercial clouds is a constant challenge. The ability to leverage the LUMI infrastructure significantly reduces model training time from weeks to days, accelerating our R&D cycle.

Table 1. Cost comparison across major cloud providers and GPU prices for DigiFarm in 2024

Cloud Provider	GPU	Cost Per Hour
Lambda Labs	NVIDIA A100	$1.29
Microsoft Azure	NVIDIA V100	$3.06
Google Cloud	NVIDIA V100	$2.48
Google Cloud	NVIDIA A100	$2.61
Amazon Web Services	NVIDIA A100	$3.50
Microsoft Azure	NVIDIA V100	$3.06
Oracle Cloud Infrastructure	NVIDIA A100	$3.05
Average		$2.72

The project's goal was to build a more complex and "universal" geographical model capable of reaching an IoU of 0.94-0.96+. Since 2019, our team of 28 full-time GIS engineers has generated over 5 million hectares of manual training data from 31 countries. This continuous effort is essential for improving model accuracy and expanding its applicability to new use-cases, such as autonomous robotics, drone spraying, and in-field object detection.

3. Methodology

Our approach is based on a computer-based system for delineating agricultural fields from satellite images, as detailed in our filed patent [6]. The system architecture can be broken down into three primary subsystems.

First, a pre-processing subsystem ingests multitemporal, multispectral satellite image sequences (primarily Sentinel-2). This subsystem performs essential radiometric calibration, atmospheric correction (converting pixel values to Bottom-of-Atmosphere reflectance), data assimilation, and geo-referencing to create a consistent, analysis-ready data stream. When data from multiple satellite systems is used, a fusion artificial neural network consolidates the various inputs into a single, harmonized sequence.

Second, a super-resolution subsystem increases the spatial resolution of the pre-processed data. This is not simple up sampling; it employs a resolution-enhancing Convolutional Neural Network (CNN) to generate high-resolution information from the multitemporal and multispectral data. This process increases the effective resolution of Sentinel-2 imagery from 10 meters to 1 meter per pixel, which is critical for accurately detecting boundaries of small fields.

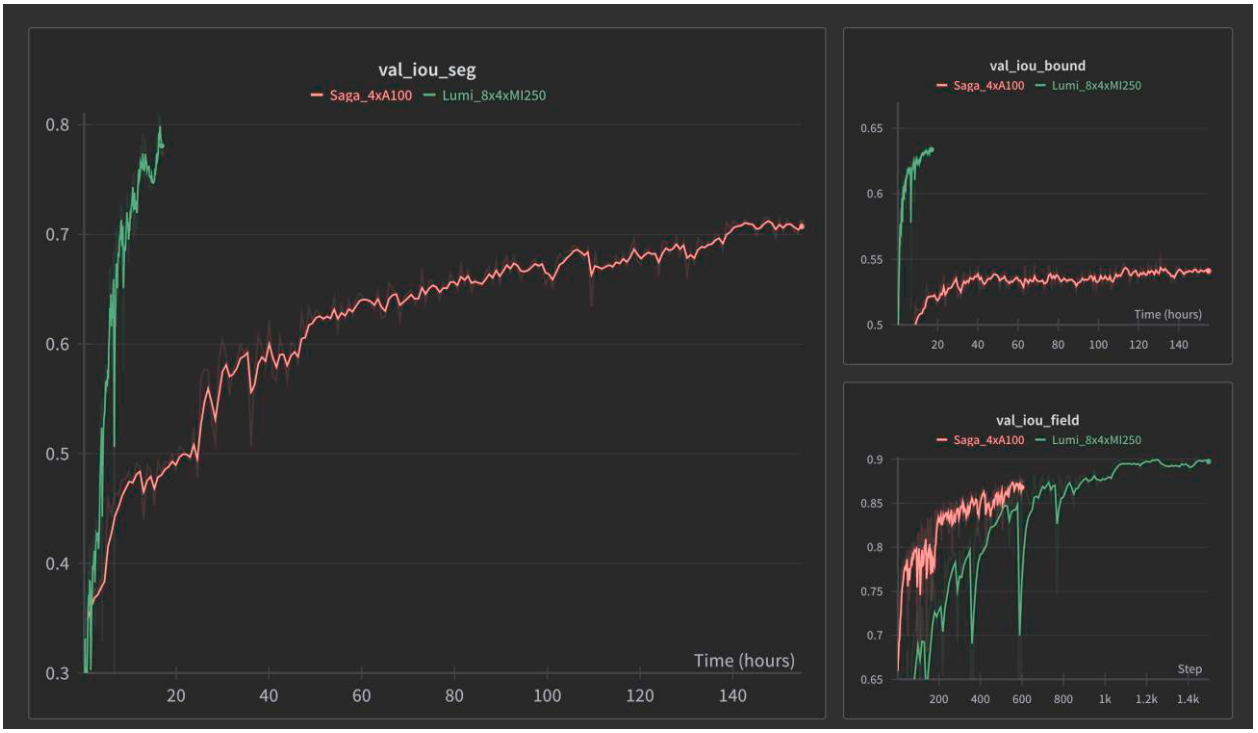
Third, a delineation subsystem uses a delineating artificial neural network (ANN) to classify pixels in the high-resolution images. Our model is based on a deep convolutional network with an encoder-decoder (U-Net) architecture [10]. A key innovation in our method is the generation of two distinct output masks: a "field extent" mask that classifies pixels as being part of a field or background, and a "field boundary" mask that explicitly identifies the edges between fields. By combining these masks (e.g., subtracting the boundary mask from the extent mask), we produce a highly accurate and clean final delineation.

The model is trained and validated using our extensive dataset of manually annotated images. We use Intersection over Union (IoU) as the primary metric to measure accuracy by comparing the model's predicted polygons with the ground-truth labels.

4. Results and Illustrations

The model training conducted on LUMI GPUs during the project period validated the significant increase in speed and performance compared to our previous training on conventional HPC systems with NVIDIA A100s. The key metrics below are visualized in Figure 1.

Figure 1. Comparison of model training performance on LUMI (green) vs. a conventional cloud HPC setup using NVIDIA A100s (red). (a) The val_iou_seg shows that the LUMI model achieves higher segmentation accuracy more quickly. (b) The val_iou_bound demonstrates superior boundary detection accuracy on LUMI. (c) The val_iou_field, our primary metric, stabilizes at ~0.94 on LUMI, a significant improvement over the A100 baseline.



1. Validation IoU for Segmentation (val_iou_seg): This metric measures how well the model performs at segmenting the image into agricultural fields versus non-fields. A higher value indicates a better overlap between the predicted and ground-truth regions. The LUMI-trained model (green line) reaches a higher IoU much faster (~20 hours) and shows superior performance, while the A100-based model (red line) improves more slowly and plateaus at a lower accuracy.

2. Validation IoU for Boundaries (val_iou_bound): This metric evaluates how accurately the model predicts the edges or borders of the segmented fields. The model trained on LUMI shows rapid performance improvement, achieving an IoU of ~0.64 early in training, whereas the A100-based model struggles around ~0.54.

3. Validation IoU for Field-level Predictions (val_iou_field): This is the most critical metric, measuring how well entire agricultural fields are predicted against ground-truth labels. The LUMI-trained model shows a significantly

faster learning rate, stabilizing at an impressive IoU of ~0.94, compared to the A100-based model which stabilizes at ~0.85 at a much slower pace.

Table 2. DigiFarm's current processes and run-times on deep neural network models on GPU infrastructure and HPC.

Process	Geographical Area Size	Type of GPU, Size & Quantity	Average time for process (in hours)
Deep-Resolution of Sentinel-2 from 10m to 1m per pixel resolution across all 12 spectral bands	1,000,000 hectares of land area also referred to as one Sentinel-2 tile (MGRS grid) which is 10,000 sq.km	NVIDIA A100, 24GB, process run on 1 GPU	~7
Field boundary delineation model (2, 4 and 12 classes)	1,000,000 hectares of land area also referred to as one Sentinel-2 tile (MGRS grid) which is 10,000 sq.km	NVIDIA A100, 24GB, process run on 1 GPU	~8-11
Field boundary delineation model and inference of results (2, 4 and 12 classes)	1,000,000 hectares of land area also referred to as one Sentinel-2 tile (MGRS grid) which is 10,000 sq.km	NVIDIA A100, 24GB, process run on 1 GPU	~9

The metrics above are further supported by the performance results of the final model trained on the LUMI infrastructure, detailed in Table 3.

Table 3. DigiFarm's current model training output and results, i.e. in terms of accuracy and performance measured in IoU (Intersection over Union), the most important being "val_iou_field" showing the accuracy of the automatic field boundary delineation achieving 0.94 IoU.

Metric	Best Performing Model	Peak Value	Speed to Converge
val_iou_seg	LUMI	~0.80	~20 hrs
val_iou_bound	LUMI	~0.64	~10 hours
val_iou_field	LUMI	~0.94	-

Figure 2 illustrates the technical architecture and data flow of the system. It shows the end-to-end process from ingesting raw Sentinel-2 data to delivering final, delineated field boundaries via an API.

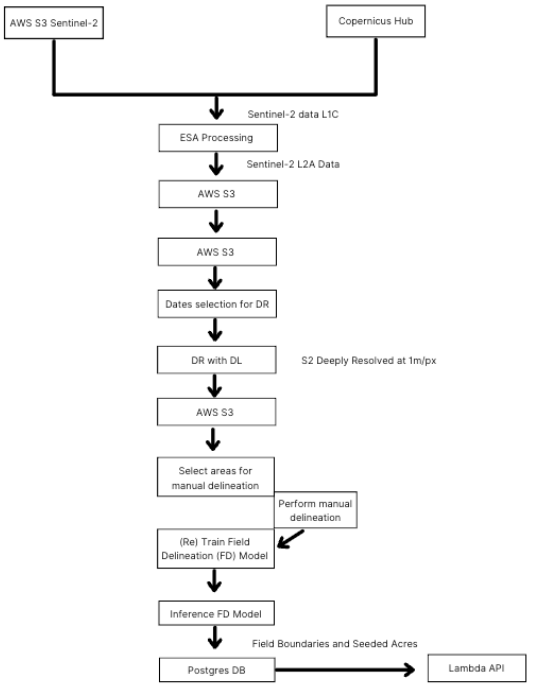
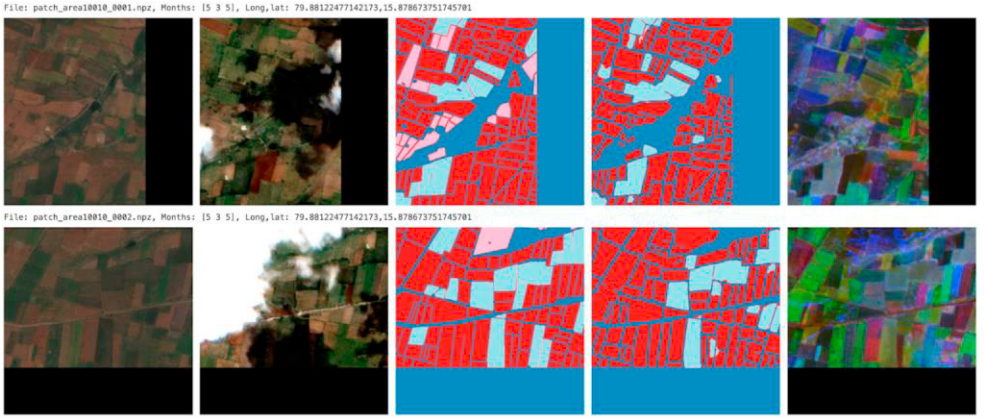


Figure 3 provides a visual example of the segmentation and inference output. It displays the progression from the original low-resolution image to the final, accurately delineated field mask based on a false-color vegetation index.

Figure 3. Example of the model's segmentation output. From left to right: (a) Original Sentinel-2 image (10m resolution); (b) Super-resolved Sentinel-2 image (1m resolution); (c) Ground truth label from manual annotation; (d) Model prediction output mask; (e) False color composite (using Enhanced Vegetation Index) to visually confirm vegetation state.



Finally, Figure 4 shows a practical application of the model in Alberta, Canada, where it automatically delineates seeded and unseeded areas within a crop field, a crucial capability for subsidy verification and farm management.

Figure 4. Example from Alberta, Canada, of the model automatically delineating seeded acres (within yellow boundaries) and unseeded acres based on in-season vegetation data. This demonstrates the model's ability to detect "internal" boundaries within a field.

5. Conclusion and Future Work

This work demonstrates the transformative potential of combining satellite earth observation data, deep neural networks, and large-scale GPU infrastructure for agricultural intelligence. Through the EuroHPC allocation and training on the LUMI supercomputer, DigiFarm significantly improved the performance and scalability of its field boundary delineation model — achieving an average IoU above 0.94 across 12 land cover classes and realizing a 4.2% performance gain over previous baselines.

The results highlight the benefits of optimized HPC environments for model convergence speed, inference efficiency, and spatial generalization. With over 5 million hectares of manually annotated training data and country-wide deployments across several EU regions, this research paves the way for large-scale, low-cost, and accurate field-level analytics crucial to agricultural subsidies, crop monitoring, and sustainable land management.

Future work includes expanding the work and modeling in terms of: (a) geographical reach, i.e. 83% of the world's agricultural field boundaries are smaller than 2 hectares, which are located in smallholder farming regions, which is significantly less digitised than the more advanced agricultural markets coupled with a lower ag-technology adoption rate and lack of publicly available data, DigiFarm intends to use the results and output of this project to further develop solutions for both the Africa and the APAC region. Additionally (b) further work includes leveraging the conceptual technology to develop object detection models for non-agricultural settings, including primary use-cases such as urban planning, maritime awareness and monitoring as well as defense and military.

Acknowledgements

DigiFarm gratefully acknowledges the funding and support received from Innovation Norway, Research Council of Norway, and the Norwegian Space Agency. The authors also wish to thank the EuroHPC Joint Undertaking (JU) for awarding this project access to the LUMI supercomputer, hosted by CSC in Finland, under the 'AI and Data-Intensive Applications' call (Proposal ID: EHPC-AI-2024A01-080). We thank the LUMI support team, Sigma2/Uninett, and our R&D collaborators. Dissemination activities included major presentations at Innovasjonstalen (Innovation Norway), EuroHPC Summit, ICE Rome, and SUREDOS24 among others.

Ethical Considerations and Reproducibility

This research was conducted in full compliance with ethical and data protection standards. DigiFarm does not collect or process any personal or farm-owner-level data. All training datasets are derived from open-access Sentinel-2 satellite imagery provided by the European Space Agency, and internal manual annotations conducted by trained GIS engineers. To mitigate biases and ensure high-quality labels, a 3-stage annotation and review protocol was applied across all training data. Final model evaluation was performed on unseen regions with diverse geographical characteristics. All models were trained using containerized workflows with documented dependencies, enabling full reproducibility on compatible HPC environments. While certain components of the system remain proprietary due to commercial deployment, key elements including our pre-processing pipeline and evaluation metrics are available for academic collaboration upon request. This project supports the principles of FAIR data, responsible AI, and open scientific collaboration.

References

[1] United Nations, Department of Economic and Social Affairs, Population Division (2019). World Population Prospects 2019: Highlights.
[2] Intergovernmental Panel on Climate Change (IPCC). (2019). Climate Change and Land.
[3] FAO. (2017). The future of food and agriculture – Trends and challenges.
[4] Helset, N., Varik, K., Melnitchouck, A. (2024). Deep Super-Resolution of Sentinel-2 Imagery for Agricultural Applications. In Review.
[5] European Space Agency, Copernicus Programme. Sentinel-2 User Guide.
[6] Melnitchouck, A. et al. (2022). Method and system for delineating agricultural fields in satellite images. Patent Application WO2023043317A1.
[7] Ruß, G., et al. (2018). “Multi-temporal detection of field boundaries in satellite images.” Computers and Electronics in Agriculture, 151, 374–384.
[8] Qiu, Y., et al. (2019). ”Deep learning for image-based large-scale crop classification.” Remote Sensing of Environment, 221, 430–443.
[9] Waldner, F., & Di Gregorio, A. (2021). “From pixels to parcels: A review of approaches to land cover mapping.” Remote Sensing of Environment, 252, 112102.
[10] Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., & Liang, J. (2018). “UNet++: A nested U-Net architecture for medical image segmentation.” Deep Learning in Medical Image Analysis.
[11] Baier, G., et al. (2021). “Mapping field boundaries using Sentinel-2 data and deep learning.” International Journal of Applied Earth Observation and Geoinformation, 100, 102338.
[12] Mulla, D.J. (2013). “Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps.” Biosystems Engineering, 114(4), 358–371.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 165–175

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Dynamic Resources Management for Exascale in ADMIRE Framework

Jesus Carretero[a,*], David E. Singh[a], Javier Fernandez-Muñoz[a], Rocco Sedona[b], Simone Pernice[e], Barbara Cantalupo[e], Marco Aldinucci[e], Massimo Torquati[e], Ahmad Tarraf[c], Emmanuel Jeannot[d], Felix Wolf[c]

[a] *Universidad Carlos III de Madrid, Leganes, Spain*
[b] *Forschungszentrum Jülich, Jülich, Germany*
[c] *Technical University of Darmstadt, Darmstadt, Germany*
[d] *INRIA Bordeaux, France, DDN-Japan and RIKEN-CCS, Japan*
[e] *University of Torino, Italy*

Abstract

Due to new workloads arising with workflows and AI applications, dynamic resource allocation can improve productivity across all system and application levels by adapting the applications' configurations to the system's resources. However, HPC system software is not suited nowadays to provide dynamic resource management (DRM) for computing or I/O resources in runtime. This paper presents the ADMIRE framework and its mechanisms for dynamic resource management. ADMIRE's efforts enable an integrated stack in which storage, compute, and orchestration layers were co-designed and validated across a shared, heterogeneous testbed.

Keywords: Dynamic resource management; Adaptivity; Intelligent control; Performance models; High-Performance Computing;

1. Introduction

With the increase of multimodal scientific simulations and new applications, such as machine learning models and workflows, the growing need to process huge data sets and manage system resources effectively is a major driver for building Exascale HPC systems today [14]. Even though dynamic resource allocation can improve productivity across all system and application levels by adapting the applications' configurations to the system's resources, the HPC system software is not suited nowadays to provide dynamic resource management (DRM) for computing or I/O resources at runtime due to several reasons.

Concerning the *I/O stack*, the flat storage hierarchies found in classic HPC architectures and the lack of flexibility to manage resources dynamically in HPC systems no longer satisfy the performance requirements of data-processing

* Corresponding author. Tel.: +34-916249458; Fax: +34-916249129
E-mail address: jesus.carretero@uc3m.es

1877-0509
10.1016/j.procs.2025.08.243

applications. Uncoordinated file access, combined with limited bandwidth, makes any centralized back-end parallel file system a serious bottleneck. Emerging multi-tier storage hierarchies can potentially remove this barrier. However, maximizing performance still requires careful control to avoid congestion and balance computational and storage performance. Unfortunately, appropriate interfaces and policies for managing such an enhanced I/O stack are still lacking.

Regarding *computing resource allocations*, current schedulers used in HPC systems, such as SLURM, rely mainly on rigid, static allocation of computing resources, giving nodes exclusively to one application. In this scenario, users deploy jobs, mainly following a batch mechanism, requesting resources based on their experience with the application and the system. The scheduler evaluates all the jobs requested in a period and deploys the applications, allocating the requested resources or blocking them until resources become available. Once a job is deployed, resources are linked to it until termination, independently of the needs of the application or the system. The scheduler typically considers the workload and available resources to perform these allocations. However, complex workflows and heterogeneous workload profiles can change the behaviour of the applications during runtime, generating inefficiencies that cannot be solved by the applications or the scheduler nowadays.

The main objective of the ADMIRE[1] [11] project is to establish controls by creating an system software stack that dynamically adjusts computation and storage requirements through intelligent global coordination of resources, malleability of computation and I/O, and new scheduling techniques for computing and storage resources along all levels of the system software stack. To achieve this, we developed a software-defined framework based on scalable monitoring and control principles, separated control and data paths, and the orchestration of key system components and applications through embedded control points.

In this paper, we present the ADMIRE framework, a software-only solution that allows substantially increasing the throughput of HPC systems and the performance of individual applications and, consequently, decreasing energy consumption. Our framework provides DRM of the systems at runtime through malleability [12], ad-hoc file systems that can exploit fast and power-efficient node-local storage tiers (RAM, NVMe, SSD) [1], and in-transit/in-situ processing facilities. The main contributions of this paper are:

- Presenting an adaptive HPC system software framework for HPC systems that utilize multi-hierarchical storage alongside dynamic resource management to boost local application and global system performance.
- Demonstrating how the intelligent control continuously analyses and adjusts how resources are used, improving throughput, reducing inefficiencies, and enabling a flexible, adaptive, and efficient HPC ecosystem.
- Evaluating our framework on a real system in production based on several popular applications.

An integrated and operational prototype has been validated with several use cases from various domains, including climate/weather, life sciences, physics, remote sensing, and deep learning, showing the feasibility of the solution proposed. The results obtained show that ADMIRE tools are able to enhance several performance metrics related to application makespan, reducing contention in back-end file systems, enhancing I/O performance, and optimizing energy consumption, among several others.

The structure of the paper is as follows. We introduce the ADMIRE framework and its components in Section 2. Section 3 presents an intelligent controller for the system resources. The modeling and predictive component of our framework is explained in Section 4, while Section 5 discusses our system prediction approach. Section 6 presents the co-design and results of applications. Finally, we provide a conclusion in Section 7

2. The ADMIRE Framework

Figure 1 shows the primary components of the framework and the use cases codesigned to assess the feasibility of our approach. It also shows how information, data, and controls are exchanged between the components. As observed in the figure, the heart of the framework is the intelligent controller, a distributed control system that integrates monitoring data from systems and applications to apply performance models and predictive AI techniques to make

[1] https://admire-eurohpc.eu/

decisions to reshape applications and ad-hoc storage system configuration at runtime. The Data Scheduler is responsible for the deployment and configuration of the Ad-hoc Storage System, the specification of bandwidth metrics, and the implementation of I/O and data scheduling policies. The Malleability Manager determines and suggests malleable actions related to each running application and ad-hoc storage system. These actions may produce the reconfiguration of processes/threads of a specific application or the deployment/removal of one or more instances of the Ad-hoc Storage System to balance the computation and I/O requirements optimally. The storage subsystem is represented in the upper part of the figure and consists of the Ad-hoc and Backend Storage Systems. The former is responsible for providing each application with an ad-hoc high-performance storage system tailored to the application's characteristics. The latter (Backend Storage) represents the parallel file system used by the HPC platform (e.g., Lustre or BeeGFS).

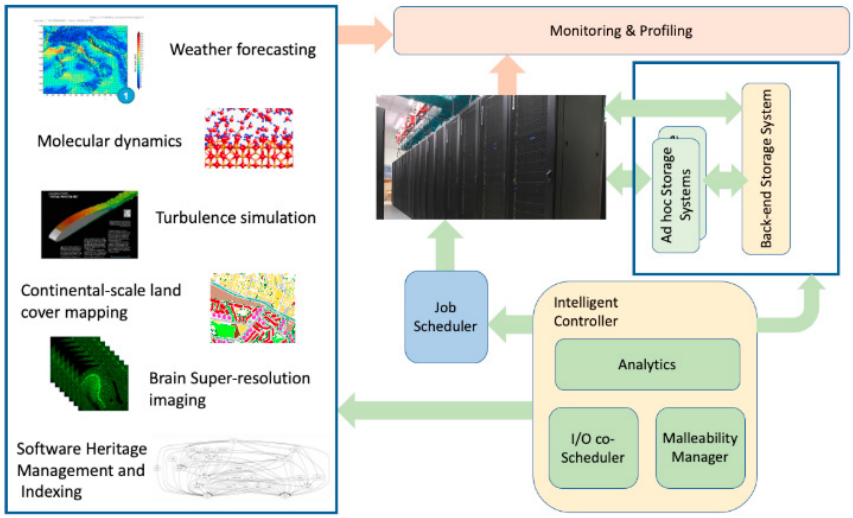


Fig. 1. ADMIRE architectural framework illustrating the interaction between the components and the application co-design aspect.

The applications ported to the framework are shown in the left part of Figure 1. ADMIRE-enabled applications can provide user-defined, application-specific information to the system to aid in the identification of I/O patterns and reconfiguration-safe states in which malleability commands can be executed. Both applications and storage systems are monitored by the Monitoring and Profiling (MP) component, which collects system-wide performance metrics at various levels (job, compute node, MPI, syscall, etc.) and stores them in an internal database. This component generates the performance models for each application, which are used to support the Malleability Manager and the Intelligent Controller in finding the most appropriate scheduling policy. A monitoring manager leverages the information stored in the database to generate performance models related to (1) each running application, (2) all storage systems (both ad-hoc and backend), and (3) the compute nodes.

3. The Intelligent Controller

The Intelligent Controller (IC) is a multi-criteria distributed component that integrates cross-layer data to have a holistic view of the system. The IC uses data to dynamically and intelligently steer the system components, improving the I/O system behavior and facilitating anticipatory decisions for resource allocation.

As illustrated in Figure 1, the Intelligent Controller is the main component in our framework, forwarding and processing data to the various other components. Overall, the IC behaves as a MAPE loop (Monitor-Analyze-Plan-Execute) and is in charge of coordinating the different architectural components to optimize the performance of the applications and, therefore, the system throughput. To this aim, the IC collects and joins cross-layer information through different data sources:

- A dynamic "control plane", composed of the monitoring and profiling system, the application manager, the Slurm manager, the malleability manager, and the I/O Scheduler.
- Information from the "data plane," composed of the ad-hoc and backend storage systems.
- Information provided by the applications and the HPC platform through the monitoring and profiling module.

The IC interoperates with all the components through the control points and related API, which are defined for each one specifically. From this perspective, the main roles of the IC are:

- Collecting information to define the current system status, including combined information from the control and data plane, the existing running applications, and historical job records.
- Providing different kinds of performance models according to the specific data sources and using them to predict potential performance bottlenecks in the system.
- Making intelligent analyses on status information and performance models to design a malleable schedule solution, considering suggestions provided by the Malleability Manager and the I/O scheduler.
- Providing programming support allowing applications to define basic malleability operations and to provide hints about dynamic resource management.

The primary role of the Intelligent Controller (IC) is to manage application execution automatically on a dynamic platform, where computing and I/O resources are provided according to the most appropriate provisioning policies. The IC decision-making process allows for application execution and reconfiguration according to parameters set by applying prediction models. This process is supported by monitoring and profiling features that can provide different metrics of the running applications, system state, and behavior. Data from the system and applications are fed into the Analytics module of the IC to predict the behaviour of the application and to make malleability decisions.

3.1. The Analytics Component

This paper focuses on the analytics component of the IC, which is distributed across different architecture modules based on the data sources used to feed the performance modelling tools and depending on the prediction objective. Several approaches have been investigated and implemented to model and predict the overall system behaviour, and strong interaction has been developed among the IC and ADMIRE components, mainly the monitoring and profiling modules. The fundamental components of ADMIRE Analytics are:

- Data sources.
- System behaviour prediction models.
- Performance (prediction) models.

The Analytics component of the IC can holistically integrate and analyse cross-layer system data to dynamically and intelligently steer the system to tune I/O performance at the system scale [5]. Three main types of performance **data sources** are provided as input to the Intelligent Controller (IC) [7]:

- **Per-Node Time-Series:** Each node in the ADMIRE-enabled supercomputer features a Prometheus time-series database (TSDB), which is used to persist the local performance counters. The node-level nature of the database avoids facing a data deluge when handling the combinatorial nature of jobs.
- **Tree-Based Overlay Network (TBON):** The LIMITLESS project provides support for spatially reducing performance data over time. In practice, it allows data samples to be summed up at high frequency, providing a real-time global view of the system.
- **Per-Job Profiles:** As understanding individual jobs is essential, the monitoring infrastructure also captures for each job the final values of all metrics (which are summative), allowing post-mortem analysis to generate performance models.

System behaviour and performance prediction models provided in ADMIRE are described in sections 4 and 5.

3.2. The Malleability Manager

One of the key aspects examined in the ADMIRE project is job malleability [19], which refers to extending or shrinking the resources of a job. In this context, the project distinguishes between compute and I/O malleability.

To make reasonable decisions, a dedicated component inside the Intelligent Controller, namely the Malleability Manager, must consider the system's current state, the jobs in the queue, user hints, and the scaling performance of an application. Consequently, this manager needs performance models that describe how the application's behaviour changes at different scales alongside continuous monitoring information. Components, such as the I/O scheduler, need more dynamic characterizations that describe the temporal I/O behaviour in terms of phases, for instance, rather than for entire executions. Consequently, we have developed and extended several tools to create a continuous modelling and monitoring framework that allows various components to acquire the needed information.

In particular, the ADMIRE project created a feedback loop between control (through the Intelligent Controller) and the developed monitoring framework to optimize system and application performance [2].

3.3. The I/O Scheduler

To achieve better performance in HPC systems within the ADMIRE framework, ad hoc storage systems are integrated and controlled by the Scord I/O scheduler. Scord provides a system-wide I/O scheduling to the parallel file system by implementing the IO-Sets method [8, 4] on top of the parallel file system (BeeGFS in the examples).

I/O scheduling strategies try to decide algorithmically which application(s) are prioritized (e.g., first-come-first-served or semi-round-robin) when accessing the shared backend storage system. The IO-sets method combines exclusive or fair-share accesses to the parallel file system by grouping applications according to their I/O frequency. The application priority, as defined by IO-Sets, is based on the periodicity of the I/O phases of an application. Scord uses FTIO [17] (Section 4) to extract the period of the I/O phases of the applications. Requests from clients generated by the file system client (FSC) reach the storage servers and are queued there for later processing by worker threads. I/O scheduling then becomes a matter of deciding in which order the requests are processed.

Our approach provides two advantages. Firstly, at the level of the parallel file system, the scheduling approach is transparent to all applications, and no intervention at either the application code or I/O library level is needed. Secondly, the parallel file system is already designed to handle a large number of requests and clients concurrently. Therefore, we can leverage its well-designed architecture to develop new QoS mechanisms, thereby avoiding adding a new point of overhead and failure in the systems (as would be the case with an external controller).

3.4. IC Integration

Concerning the platform integration, the GreatNector and Anomaly Detection tools are included in the IC decision process. We provided a simple driving scenario to demonstrate how moldability can be applied. However, the next step is defining a complete scenario for demonstrating how the available tools, starting with but not limited to GreatNector, can support the IC in I/O tuning and malleability management.

4. Performance Prediction Models and Temporal Characterizations

Performance model projections are essential for understanding I/O in modern computing systems. Running parallel programs on a given system requires satisfying the program's requirements by the back-end storage in its current state. However, the current state of the storage system can vary over time depending on the concurrent requirements of other jobs running on the machine. Therefore, building models that can predict and project I/O performance application-wise is essential. The ADMIRE project aimed to improve the understanding of these requirements so that applications can be correctly scheduled while considering these requirements. Consequently, the monitoring infrastructure allows for the application-wise projection of I/O performance, including predicting application behaviour using regression analysis alongside more dynamic predictions like characterizing the temporal I/O behaviour in the frequency domain. These projections provide valuable insights into the behaviour of the I/O subsystem, which can help improve the performance of parallel programs running on the system.

Performance modelling has a long research history [16]. A performance model is a mathematical formula expressing a performance metric of interest, such as execution time or energy consumption, as a function of one or more execution parameters (e.g., the size of the input problem or the number of processors). The IC connects three performance modelling tools: Extra-P, FlexMPI computational prediction model, and time-series-based performance model (TSBPM), along with a temporal characterization tool FTIO.

Extra-P [9] is an automatic performance-modelling tool that supports the user in identifying these bugs by generating empirical performance models that predict the scaling behaviour of the different parts of an application. For each call path of an application, Extra-P generates a performance model that predicts its scaling behaviour at different resource configurations (usually the number of processors and, optionally, other parameters like the problem size). The tool has a long research history, with recent updates adding noise-resilient empirical performance modelling capabilities based on Deep Neural Networks [16] and a strategy to reduce measurement efforts [15].

To balance the resource consumption of an application, we need to know the scaling behaviour of the computing, communications, and especially I/O functions. In particular, it is not enough to only examine the scaling of the execution time. Rather, other parameters, like the total bytes transferred, should also be modelled to paint the whole picture. With such information, for instance, the application execution can be optimized, especially in the presence of novel storage components like burst buffers exploited in the ADMIRE project. For example, knowing how many bytes the applications write at different scales could influence the kind of burst buffer (shared or per node) used and aid the decision on how many times the burst buffers need to be flushed. At the same time, the total times (e.g., total compute time and total I/O time) are also needed to examine how resource-demanding an application is at different scales. Together, these models could be very valuable for I/O contention avoidance strategies, job scheduling, and increased system performance.

FlexMPI computational prediction model (CPM) implements a mathematical model that allows the malleability policies to estimate the application performance. The execution time of a parallel application (T_{si}) depends on the computation time and the communication time: $T_{si} = T_{computation} + T_{communication}$.

In this work, we assume that the MPI application employs synchronous MPI communication operations and that the synchronization overhead counts as part of the application's communication time.

The execution time of a parallel application during a sampling interval depends on the processor performance and the system's network performance. The values that model both types of components are provided as input to the CPM. The monitoring component feeds the runtime performance metrics (collected via PAPI) to the CPM, which allows it to estimate the computation cost of the application. To effectively estimate the communication cost of the application, the CPM uses profiling data (collected by monitoring via PMPI) and the system network performance. The network performance depends both on the network topology and on the network technology, which is specific to the system environment. However, a reconfiguration action involves changing the number of processes and the workload distribution of the applications. These operations have related the overheads for the process creation and termination operations ($T_{overhead\_process\_reconfig}$), as well as for the data redistribution operations ($T_{overhead\_data\_redist}$). Therefore, the execution time of a malleable application that follows a reconfiguring action (T_{sir}) is computed as: $T_{sir} = T_{computation} + T_{communication} + T_{overhead\_process\_reconfig} + T_{overhead\_data\_redist}$

The overhead of data redistribution also depends on the system's network performance and the size of the data transferred. In FlexMPI both metrics are calculated at runtime. The overhead of process creation and termination depends on the number of processes spawned or removed, the operating system, and the size of the program binary. Therefore, these overheads are specific to the underlying hardware, the system environment, and the application characteristics. The following sections describe the way that each one of the costs is modelled.

In case the malleability policy decides to spawn or remove processes, the CPM takes into account the overhead of process creation and termination operations to predict the execution time of the next sampling interval. The time spent by FlexMPI on creating a dynamic process is different to the time spent on removing it. These overheads depend on various factors such as the operating system, the MPI implementation, and the size of the program binary. The CPM uses an offline benchmark that tests the costs of process reconfiguring actions in a particular environment.

A process spawning action in a Linux environment implies the OS's process manager creates the process through a fork-exec call, then allocates the address space to the process, and enables communication with other MPI processes through dedicated connections. The associated overheads depend on the particular execution environment of the FlexMPI application. A process removing action implies at the system level deallocating the process resources,

then all active connections with other MPI processes are closed, and finally, the process is safely finalized. In practice, the overhead of spawning a dynamic process is significantly higher than the overhead of removing a single process.

Time-series based performance model (TSBPM). In ADMIRE, we developed a monitoring tool, named LIMITLESS, which gathers performance metrics at the node level, enabling predictions specifically at the node level. However, through the integration of LIMITLESS and FlexMPI, this joint effort collects metrics for both nodes and applications, thereby enabling predictions for both aspects. Other monitoring tools can also be integrated to provide more data about the system.

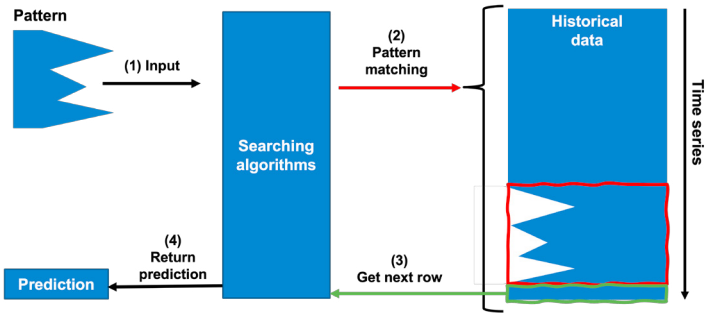


Fig. 2. Performance prediction based on time-series analysis and searching algorithms.

LIMITLESS adopts searching algorithms rather than regression or neural network models for predicting system and application performance. LIMITLESS leverages the collected metrics to manage a time-series index depending on the data source: when the time series comes from an idle node, the metrics are stored as general system behaviour. If a specific application (denoted as A) is running on the node, the corresponding performance metrics are stored as a performance model specific to A, representing an ensemble of n time series associated with its execution. When an application (X) starts its execution, the system identifies the relevant ensemble and looks for the pattern, using the partial time series generated from the start of X's execution. Once the system has found the pattern, the returned prediction is the next value of the time series (Figure 2 represents this set of operations). This solution makes sense when the application metrics between executions (with the same arguments) remain constant. Otherwise, this methodology couldn't predict the system's or application's performance.

FTIO (Frequency techniques for I/O) is a tool to characterize the temporal behaviour of an application. Since the behaviour of an application can vary during execution, it can be beneficial to characterize the current behaviour of an application rather than to have post-execution performance models. In particular, knowing when the *different phases* of an application occur can be very beneficial, for example, for contention avoidance algorithms like I/O scheduling [8] and burst buffer management [3]. This is where FTIO [17] comes in, which allows us to characterize the temporal I/O behaviour of an application in terms of phases. FTIO, which stands for frequency techniques for I/O, characterizes the I/O phases of an application using a single metric, namely the frequency. Additionally, it is a confidence metric alongside the identified frequency to gauge the confidence in the contained results. Furthermore, the tool can provide further metrics (e.g., bytes per phase) that build on the identified dominant frequency.

FTIO has two modes for providing results: offline (detection) and online (prediction). In the offline mode, FTIO reads post-simulation profiles or traces to detect the period of the I/O phases. Various existing tools, including Recorder or Darshan, and newly developed tools like TMIO and the Metric Proxy are supported. In particular, the online mode of TMIO [18] was developed for this purpose. Furthermore, the Metric Proxy has been recently extended to yield traces with a high sampling rate for FTIO. FTIO allowed the I/O scheduler Set10 to boost system utilization by 26% and reduce I/O slowdown by 56% [17]. More details are provided in [19, 8]. A recent addition to FTIO allowed executing it in parallel on thousands of gathered traces to quantify periodic I/O in HPC [21].

5. System Behaviour Prediction Models

The strategy of this component consists of training a clustering model that can handle jobs with heterogeneous I/O patterns, detect congestion, perform resource provisioning, and anomaly detection that will help the IC to identify

and predict potential risks and failures of applications. In ADMIRE, we have developed two models to predict system behaviour:

- Anomaly detection provides clustering of power time series on different nodes with the aim of predicting anomalies in the system behaviour.
- GreatNector: Based on a Markovian Model that uses application counter time series produced by the Metric Proxy [6], it provides Application Clustering that allows performing what-if analysis on the system status. Consequently, knowing which applications will be running on the system allows predicting the system behaviour in terms of performance degradation and queue congestion.

Anomaly Detection is a process of detecting rare events occurring in the system, which can help to expose hardware or software-related problems in near real time. The nature of the problems can vary from scheduler errors, hardware malfunctions, erroneous configurations, and more due to the enormous complexity of HPC systems. We have used autoencoders to implement anomaly detection. To show the feasibility of our solution, in Figure 3, we show the reconstruction error for the month of February, calculated from 150 nodes and the system trained with 4 time series sample every minute (2 CPU temperatures, power measurement, and the number of allocated CPUs). Processing time uses a sliding window of 20 minutes, moving every 10 minutes per step. The figure shows that our system was able to predict a possible anomaly 10 hours before the first reports of HPC failure were given by the monitoring system (2023-02-06 22:46:00).

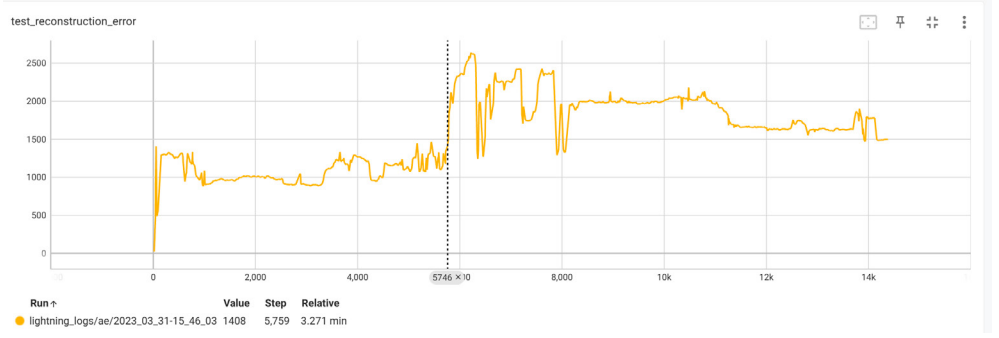


Fig. 3. Example of anomaly prediction using autoencoders.

GreatNector is a workflow developed to model applications' behaviour in HPC systems, characterized by three main steps, starting from the elaboration of the traces of the application, their clusterization into different groups which will represent the application templates characterizing all the applications with similar behaviour, and the system calibration and simulation to fit the average traces of the application templates. The steps are defined below:

- **Data processing**. Application traces are processed by a Python script by aggregating the time spent in a state and the number of calls of that state in a specific time interval (e.g., IO, MPI, etc).
- **Clustering**. There are two types of clusterizations: inter- and intra-clustering. The former fits and clusters the temporal data to highlight differences and similarities among the traces (inter-clustering). The obtained clusters' average trace will be exploited to define *template-application* as representative of all the traces that belong to the corresponding cluster. Given that the template-applications might be characterized by repetitive patterns, the intra-clustering approach simplifies the template-application into similar parts that will be used to calibrate the model.
- **Modelling**. In this step, GreatNector [13] uses the Stochastic Simulation Algorithm to simulate a Petri Net model based on a stochastic Markovian process. Unknown parameters, defined as the homogeneous groups characterizing a template-application, are calibrated and then used to run various what-if scenarios using GreatNector's simulation engine [20].

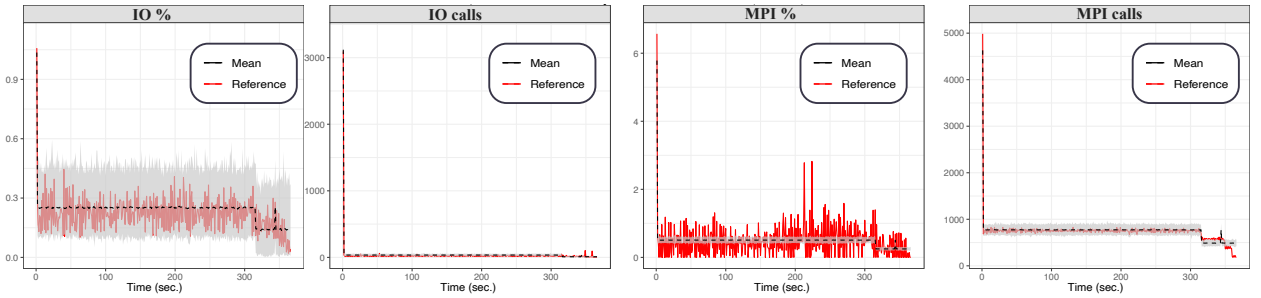


Fig. 4. Fitting results in the simulation of *QuantumEspresso* application with 8 processes using GreatNector.

As an example, QuantumEspresso was modeled, and the fitting results are shown in Fig.4. The dashed black line is the mean of the average trace considering 500 simulations and the red solid line the trace of calls. As may be seen, the average simulations capture the average pattern of the application.

6. Results

In ADMIRE, use cases played a central role not merely as targets for technology demonstration but as active drivers of co-design. The diversity of the use cases—ranging from real-time environmental forecasting to large-scale code archive analytics—allowed ADMIRE to validate its solutions across a broad spectrum of workloads, reinforcing both horizontal and vertical integration across the HPC ecosystem. The six use cases are: An environmental application (WaComM++) that focuses on modelling water quality and ecosystem dynamics; Quantum ESPRESSO, a widely used suite that provides a computationally intensive and GPU-accelerated workload; NEK5000, a high-fidelity spectral element code for fluid dynamics, often used in nuclear reactor and turbulence simulations; Remote Sensing, which deals with large-scale geospatial data analysis and deep learning workloads for remote sensing imagery; Life Sciences, representing biomedical workloads, employed large-scale omics data and machine learning pipelines; and Software Heritage Analytics application, an analytics use case mining the Software Heritage archive—a large collection of public source code. Those applications are programmed in C++, FORTRAN, and Python, so the ADMIRE framework provided interfaces for those 3 programming languages.

To enable co-design across applications and ADMIRE technologies, all tools were integrated and validated on a consistent HPC environment, a cluster hosted at the University of Turin. This platform, part of HPC4AI, features a diverse set of compute architectures [2].

Compute malleability was implemented using two complementary approaches, depending on the application's programming model. For traditional MPI-based applications such as NEK5000 and WaComM++, ADMIRE adopted Flex-MPI, a dynamic runtime extension of MPICH that tracks performance via hardware counters and MPI profiling interfaces. For non-MPI applications or MPI applications where FlexMPI was not suitable due to their complexity with communicators or data redistribution, start-stop model has been developed. In this model, the IC uses checkpoint stages of an application to restart it with the required number of new processes. For machine learning applications, we provide a connector for Elastic Horovod (see Table 1).

Use Case	IC integration	FlexMPI	Start-Stop	Elastic Horovod
Environmental Application	X	X		
NEK5000	X		X	
Quantum Espresso	X			
Remote Sensing	X			X
Life Sciences	X			X
Software Heritage Analytics				

Table 1. Overview of ADMIRE use case support across storage, integration, and dynamic resource management capabilities.

[2] https://hpc4ai.unito.it/

Flex-MPI was particularly effective in WaComM++ [10], where the computational load varies across simulation cycles depending on the behaviour of simulated particles. A simplified C implementation of WaComM++ was developed to facilitate the integration of Flex-MPI, allowing for precise experimentation with elasticity under controlled conditions. The ability to shrink or grow the number of active MPI ranks during execution enabled finer-grained control over performance and resource usage, aligning compute effort with problem complexity over time (See Fig. 5). Large-scale experiments have shown that, by adapting dynamically to the workload per iteration through ADMIRE tools, WaComM++ application turnaround can be reduced up to 45%.

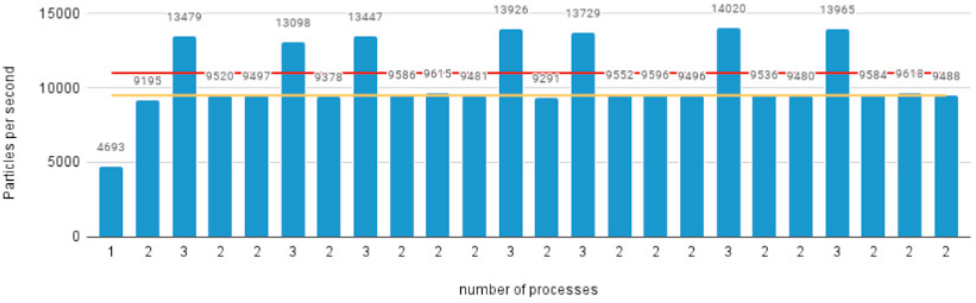


Fig. 5. Dynamic resources adaptation in WaComM++ with the computational load.

7. Conclusion

In this paper, the ADMIRE framework and its mechanisms for dynamic resource management have been presented. ADMIRE's efforts enabled an integrated stack in which storage, compute, and orchestration layers were co-designed and validated across a shared, heterogeneous testbed. All components were monitored and coordinated in real-time through an intelligent controller capable of interpreting performance alarms and initiating adaptation through near real-time analytic mechanisms, a plethora of performance prediction models that can be chosen dynamically.

Dynamic load balancing is provided even in complex cases, as we have two models for malleability: runtime, and stop-start. In the second case, when the application cannot be modified on the fly, the IC uses checkpoint stages to restart with the required number of new processes. However, it is important to mention that we have not adapted our solution to every kind of application, thus supporting dynamic resource management for some large-scale codes might not be possible with ADMIRE's framework.

Work is going on to interface the IC with PMIx and to provide management for MPI sessions. In addition, we are researching the application to composable hardware systems, so that we can provide greater flexibility, scalability, and efficiency in resource utilization compared to traditional, siloed architectures.

Acknowledgements

ADMIRE project received funding from the EuroHPC Joint Undertaking (JU) under grant agreement No 956748, as well as funding from Spain, Germany, France, Italy, Poland, and national research agencies. This publication was also partially funded by project PID2022-138050NB-I00, funded by the grant MICIU-AEI-10.13039-501100011033. Thanks to the University of Torino for the use of the HPC4AI cluster for the experiments.

References

[1] Al-Maaitah, N.O., Garcia-Blas, J., Sanchez-Gallegos, G., Carretero, J., Vef, M.A., Brinkmann, A., 2025. A comparative study of ad-hoc file systems for extreme scale computing. Future Generation Computer Systems 170, 107815. URL: https://www.sciencedirect.com/science/article/pii/S0167739X25001104, doi:https://doi.org/10.1016/j.future.2025.107815.

[2] Almaaitah, N.O., Singh, D.E., Özden, T., Carretero, J., 2024. Performance-driven scheduling for malleable workloads. The Journal of Supercomputing 80, 11556–11584.

[3] Aupy, G., Beaumont, O., Eyraud-Dubois, L., 2019. Sizing and partitioning strategies for burst-buffers to reduce IO contention, in: IPDPS'19, IEEE. pp. 631–640.
[4] Bandet, A., Boito, F., Pallez, G., 2024. Scheduling distributed i/o resources in hpc systems, in: European Conference on Parallel Processing, Springer. pp. 137–151.
[5] Besnard, J.B., Tarraf, A., Barthélemy, C., Cascajo, A., Jeannot, E., Shende, S., Wolf, F., 2023. Towards smarter schedulers: Molding jobs into the right shape via monitoring and modeling, in: International Conference on High Performance Computing, Springer. pp. 68–81.
[6] Besnard, J.B., Tarraf, A., Cascajo, A., Shende, S., . Introducing the metric proxy for holistic I/O measurements, in: Weiland, M., Neuwirth, S., Kruse, C., Weinzierl, T. (Eds.), High Performance Computing. ISC High Performance 2024 International Workshops, Springer Nature Switzerland. pp. 213–226. doi:10.1007/978-3-031-73716-9_15.
[7] Besnard, J.B., Tarraf, A., Cascajo, A., Shende, S., 2025. Introducing the metric proxy for holistic i/o measurements, in: International Conference on High Performance Computing, Springer. pp. 213–226.
[8] Boito, F., Pallez, G., Teylo, L., Vidal, N., 2023. IO-sets: Simple and efficient approaches for I/O bandwidth management. IEEE Transactions on Parallel and Distributed Systems 34, 2783–2796.
[9] Calotoiu, A., Hoefler, T., Poke, M., Wolf, F., 2013. Using automated performance modeling to find scalability bugs in complex codes, in: Proc. of the ACM/IEEE Conference on Supercomputing (SC13), Denver, CO, USA, ACM. pp. 1–12. doi:10.1145/2503210.2503277.
[10] Carretero, J., Exposito, D., Cascajo, A., Montella, R., 2022. Malleability techniques for hpc systems, in: International Conference on Parallel Processing and Applied Mathematics, Springer. pp. 77–88.
[11] Carretero, J., Garcia-Blas, J., Aldinucci, M., Besnard, J.B., Acquaviva, J.T., Brinkmann, A., Vef, M.A., Jeannot, E., Miranda, A., Nou, R., et al., 2023. Adaptive multi-tier intelligent data manager for exascale, in: Proceedings of the 20th ACM International Conference on Computing Frontiers, pp. 285–290.
[12] Carretero, J., Suarez, E., Schulz, M., 2024. Malleability techniques applications in high-performance computing.
[13] Pernice, S., Maglione, A., Tortarolo, D., Sirovich, R., Clerico, M., Rolla, S., Beccuti, M., Cordero, F., 2023. A new computational workflow to guide personalized drug therapy. Journal of Biomedical Informatics 148, 104546.
[14] Pleiter, D., Haus, U.U., Buttari, A., Anciaux-Sedrakian, A., Garcia, A., Ma, B., Samaey, G., Battiato, G., Houzeaux, G., Brandt, L., et al., 2025. ETP4HPC SRA 6 White Paper-Mathematical Methods and Algorithms. Technical Report. European Technological Platform for HPC.
[15] Ritter, M., Calotoiu, A., Rinke, S., Reimann, T., Hoefler, T., Wolf, F., 2020. Learning cost-effective sampling strategies for empirical performance modeling, in: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 884–895. doi:10.1109/IPDPS47924.2020.00095.
[16] Ritter, M., Geiß, A., Wehrstein, J., Calotoiu, A., Reimann, T., Hoefler, T., Wolf, F., 2021. Noise-resilient empirical performance modeling with deep neural networks, in: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), p. 23–34. doi:10.1109/IPDPS49936.2021.00012.
[17] Tarraf, A., Bandet, A., Boito, F., Pallez, G., Wolf, F., a. Capturing periodic I/O using frequency techniques, in: 2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 465–478. doi:10.1109/IPDPS57955.2024.00048.
[18] Tarraf, A., Muñoz, J.F., Singh, D.E., Özden, T., Carretero, J., Wolf, F., b. I/O Behind the Scenes: Bandwidth Requirements of HPC Applications With Asynchronous I/O, in: 2024 IEEE International Conference on Cluster Computing (CLUSTER), pp. 426–439. doi:10.1109/CLUSTER59578.2024.00044.
[19] Tarraf, A., Schreiber, M., Cascajo, A., Besnard, J.B., Vef, M.A., Huber, D., Happ, S., Brinkmann, A., Singh, D.E., Hoppe, H.C., Miranda, A., Peña, A.J., Machado, R., Gasulla, M.G., Schulz, M., Carpenter, P., Pickartz, S., Rotaru, T., Iserte, S., Lopez, V., Ejarque, J., Sirwani, H., Carretero, J., Wolf, F., c. Malleability in modern HPC systems: Current experiences, challenges, and future opportunities. IEEE Transactions on Parallel and Distributed Systems , 1–14doi:10.1109/TPDS.2024.3406764.
[20] Terrone, I., Volpatto, D., Pernice, S., Amparore, E., Sirovich, R., Cordero, F., Beccuti, M., 2025. Extension of the greatmod modeling framework to simulate non-markovian processes with general-distributed events, in: Computational Intelligence Methods for Bioinformatics and Biostatistics, Springer Nature Switzerland, Cham. pp. 177–191.
[21] Zanon Boito, F., Teylo, L., Popov, M., Jolivel, T., Tessier, F., Luettgau, J., Monniot, J., Tarraf, A., Carneiro, A., Osthoff, C., 2025. A deep look into the temporal I/O behavior of HPC applications -extended version. Technical Report RR-9577. Inria & Labri, Univ. Bordeaux. URL: https://inria.hal.science/hal-04978752.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 176–186

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Enhancing Financial NLP with Supercomputing: Spell Correction and Domain-Specific BERT Pretraining

Derya Uysal[a], Doğacan Toka[a], Elif Bozkurt[a], Osman Kumaş[a], Hasan Hüseyin Yılmaz[a]

[a] *Yapi Kredi Technology, Istanbul, Turkey*

Abstract

In this study, we present a two-stage approach to enhance natural language understanding in financial applications using deep learning techniques, supported by high-performance computing infrastructure. The first stage focuses on improving user input clarity through automatic spelling correction. We fine-tune the mT5 language model to detect and correct misspelled words in user messages, particularly in informal, user-generated financial texts. This preprocessing step is essential in financial chatbot systems, where even small textual errors can lead to critical misunderstandings or misclassifications. In the second stage, we pretrain a BERT-based language model on large-scale, open-source Turkish financial corpora to create a domain-adapted representation suitable for a variety of NLP tasks such as intent detection, entity recognition, and document classification. Unlike most prior work focusing on high-resource languages, our study addresses the significant challenge of developing domain-specific models for Turkish—a morphologically rich and under-resourced language. To accelerate model training and ensure scalability, all computations were carried out on supercomputers provided by the EuroHPC initiative. This infrastructure enabled faster training and higher model efficiency, reducing a task that took 19 hours on a local machine to just 30 minutes—achieving a 38× speedup. Furthermore, the resulting models can be fine-tuned and deployed on private, secure institutional servers, enabling use with sensitive banking data without compromising privacy. Our work demonstrates the critical role of supercomputing resources and domain adaptation in building practical, language-specific NLP systems for the financial industry.

Keywords: Spell correction; financial BERT; conversation with AI

1. Introduction

In recent years, natural language processing (NLP) has undergone a transformative evolution, driven by advances in deep learning and the increasing availability of large-scale datasets. These developments have enabled machines to generate, and interact with human language in ways that were previously unattainable. As a result, NLP technologies are now being integrated into a wide range of real-world applications, from digital assistants and customer service bots

E-mail address: derya.uysal@ykteknoloji.com.tr

1877-0509
10.1016/j.procs.2025.08.244

to document analysis and sentiment tracking. Among these applications, the financial domain presents both unique opportunities and significant challenges due to the domain's reliance on highly specific language, the sensitivity of its data, and the critical need for accuracy.

Financial institutions are increasingly leveraging NLP to automate customer interactions, extract insights from large volumes of textual data, and support decision-making processes. However, effective deployment of NLP in this context requires more than generic language models. Financial text often includes domain-specific jargon, abbreviations, and numerical expressions that are poorly understood by general-purpose models. Moreover, user-generated financial inputs—such as customer queries or transaction descriptions—tend to be informal and error-prone, frequently containing spelling mistakes, ambiguous phrases, or code-switching between languages. These challenges necessitate the development of robust, domain-adapted NLP systems that can handle noisy inputs while maintaining high accuracy and contextual understanding. Furthermore, most research in financial NLP has focused on high-resource languages such as English or Chinese. In contrast, Turkish presents unique challenges due to its agglutinative structure, rich morphology, and limited availability of domain-specific linguistic resources. These characteristics make both spelling correction and domain adaptation significantly more difficult and underexplored, thus amplifying the impact and novelty of our approach.

Another layer of complexity arises from the stringent data security and privacy requirements in financial services. Due to regulatory constraints and the sensitive nature of financial data, cloud-based NLP solutions may not be viable for many institutions. While our approach is compatible with private fine-tuning and on-premises deployment, we do not directly address the security protocols or infrastructure needed to achieve full regulatory compliance. Instead, we acknowledge this as a critical aspect for future exploration in real-world deployments.

To address these challenges, this study proposes a two-stage deep learning approach supported by high-performance computing infrastructure. The first stage aims to improve the clarity of user input through automatic spelling correction. By fine-tuning the multilingual T5 (mT5) model, we enable the detection and correction of misspelled words in user messages, thereby reducing noise in downstream tasks and improving the coherence of conversational agents. This step is critical in real-world financial settings where user-facing systems must respond accurately and promptly, even in the presence of imperfect input. While spelling correction is a general NLP task, its role is particularly crucial in financial chatbots. Users frequently input transaction details, abbreviations, or product names with typos or informal expressions. In this domain, even a minor misspelling can mislead intent recognition or entity extraction processes, leading to customer dissatisfaction or incorrect recommendations. Therefore, we approach this task not as a generic preprocessing step, but as an essential enabler for high-precision downstream tasks in Turkish financial applications.

In the second stage, we focus on domain adaptation for financial language understanding. We utilize a BERT-based transformer architecture and perform pretraining on large-scale, publicly available financial text corpora. Specifically, a ModernBERT [7] model is pretrained on Turkish financial data, benefiting from architectural enhancements tailored to encoder-only models. This pretraining allows the model to internalize the semantic and syntactic structures common in financial discourse, resulting in a representation that is more aligned with domain-specific NLP tasks such as named entity recognition, intent classification, and document summarization within financial contexts.

Crucially, the entire training pipeline is accelerated through the use of supercomputers provided by the EuroHPC initiative. Leveraging this infrastructure not only reduces training time but also allows for the exploration of larger model architectures and datasets, pushing the boundaries of what is feasible in academic and industrial NLP research. Furthermore, the trained models are designed to allow fine-tuning and deployment on private servers using institution-specific data. While this dual approach—combining public pretraining with private adaptation—has the potential to support compliance with data governance requirements, we treat this only as a deployment possibility rather than a fully implemented security solution. Ensuring production-level data protection remains an open research problem that falls outside the scope of this study.

In summary, this work demonstrates how the synergy between advanced deep learning techniques and high-performance computing can drive progress in domain-specific NLP. By addressing the twin challenges of input quality and domain adaptation, and by providing scalable, privacy-preserving deployment pathways, our approach lays the groundwork for robust NLP solutions in the financial industry. Importantly, this study represents one of the few attempts to build a domain-specific language model for the Turkish financial sector. By combining multilingual transformer models, Turkish financial corpora, and HPC infrastructure, we address the twin challenges of working

in a low-resource language and on a domain that requires high-precision downstream tasks. This dual-layered challenge—often overlooked in high-resource language studies—constitutes a core contribution of our work.

2. Related Works

In this section, we provide a comprehensive overview of the existing literature relevant to our study. The related works are categorized into three primary areas and each category is discussed in detail to highlight the methods, datasets, and contributions of prior studies, thereby positioning our approach within the current research landscape.

2.1. Spelling Correction in Natural Language Processing

Spelling correction is a foundational task in natural language processing, critical for ensuring input quality and preserving coherence in downstream applications—especially in sensitive domains like finance. Early approaches to spelling correction were mostly rule-based or relied on statistical models. However, with the advent of deep learning and transformer architectures, spelling correction has taken a significant leap forward. One such model is mT5 [13], a multilingual extension of the T5 transformer, pre-trained on the massive multilingual mC4 corpus. It has shown strong performance across over 100 languages, especially in grammatical and lexical error correction tasks [1].

Similarly, the ByT5 model approaches spelling correction from a character-level perspective, making it suitable for typographic and diacritic restoration tasks across 13 languages. This method significantly outperforms traditional dictionary-based spell checkers, particularly in morphologically rich languages [2]. These methods highlight the growing relevance of transformer-based models in solving orthographic and typographic inconsistencies in user-generated text—laying a solid foundation for domain-specific applications like financial conversational agents. However, few studies have explored multilingual transformer adaptation for spelling correction in domain-specific Turkish datasets, especially in financial contexts. Our work contributes to this body of research by evaluating and adapting mT5 for Turkish financial spelling correction, a setting that combines both low-resource and high-noise characteristics.

2.2. Domain Adaptation in Financial NLP

The financial domain poses unique challenges for NLP models, such as jargon, abbreviations, and significant requirements for semantic precision. General-purpose models like BERT may perform inadequately without domain-specific tuning. FinBERT was one of the pioneering efforts to address this. Trained on large-scale financial corpora such as Reuters TRC2, FinBERT demonstrated substantial gains in financial sentiment analysis and entity recognition tasks. It showed that even a general model like BERT could be significantly enhanced through domain-specific pretraining [3].

Further research compared models like FinBERT and domain-adapted BERTs and concluded that continual pretraining on in-domain data yields more consistent improvements than training from scratch [4]. This strategy also conserves computational resources while leveraging existing model knowledge. Non-English financial NLP has also seen promising progress. KR-FinBERT is a Korean BERT model trained on financial news and reports, achieving up to 96.3% accuracy in sentiment analysis tasks [5]. Likewise, research into Indonesian financial texts showed that domain-specific adaptation of multilingual BERT (mBERT) outperformed both standard and local models in classification and named entity recognition tasks [6]. Together, these efforts underscore the necessity of domain adaptation to meet the nuanced demands of financial language processing, particularly across multilingual contexts. Yet, existing financial BERT variants have largely focused on English or East Asian languages, leaving a gap for Turkish financial language modeling.

2.3. High-Performance Computing in NLP Training

Training large-scale transformer models is computationally intensive, making high-performance computing (HPC) infrastructures essential for timely and scalable model development. The EuroHPC initiative provides state-of-the-art supercomputers to support AI research across Europe. For example, the TrustLLM [26] project is leveraging this infrastructure to develop large, trustworthy language models tailored for European needs. This project illustrates the importance of scalable compute resources for both foundational model training and fine-tuning on sensitive data.

The Leonardo supercomputer has also been employed for NLP research, enabling institutions like RISE to experiment with large language models in secure, high-throughput environments. This access accelerates innovation while maintaining data privacy and training efficiency. By using HPC resources, researchers can experiment with larger architectures, conduct extensive hyperparameter tuning, and pretrain models on domain-specific corpora—tasks that would be prohibitive on conventional hardware. Our work aligns with this direction by combining EuroHPC infrastructure with domain-specific pretraining for Turkish, providing a concrete use case for resource-constrained, high-precision NLP in finance.

3. Methodology

This section outlines the methodological framework employed in our study to enhance natural language understanding in financial applications. The proposed approach is divided into two main components: a spell correction pipeline to improve the quality of user-generated inputs, and a domain-specific language model pretraining strategy tailored for financial text processing. We first describe the datasets used in both stages, including curated collections of financial documents and multilingual textual resources. Next, we elaborate on the preprocessing strategies that convert raw data into suitable formats—such as word, sentence, and paragraph structures—optimizing them for deep learning models. The spell correction module is implemented via fine-tuning the mT5 model, while the second stage involves continued pretraining of a BERT-based architecture on financial corpora to obtain a robust, domain-adapted representation. Each component is designed to be modular and scalable, ensuring compatibility with high-performance computing environments and future domain-specific fine-tuning efforts. The following subsections detail each stage of the methodology.

3.1. Datasets

To support the two-stage architecture proposed in this study, we utilized a diverse collection of open-source datasets tailored for distinct components of our methodology. The first group of datasets comprises financial documents used for domain-specific pretraining of the ModernBERT-based language model. These were compiled from a variety of publicly available PDF sources, including financial reports, regulatory filings, and market analysis documents, to ensure rich and representative coverage of financial language. The second group consists of three curated datasets structured at the word, sentence, and paragraph levels, respectively. These datasets were specifically designed to train and evaluate the spelling correction module, enabling it to learn contextual patterns of error and correction at varying linguistic granularities. Detailed descriptions of both dataset groups are provided in the following subsections.

3.1.1. Word, Sentence, Paragraph formats for Correction

A comprehensive review of existing resources for Turkish grammatical and orthographic error correction was conducted in order to construct a robust dataset suitable for the proposed multi-phase curriculum learning approach. During this survey, multiple publicly available datasets were evaluated based on their size, quality, and relevance to the objectives of the study. The key datasets considered include:

1. GECTurk (GGLAB-KU/gecturk) [8]: approximately 100,000 annotated sentences for grammatical error correction tasks, primarily focused on formal and academic text structures.
2. Annotated Tweets Dataset (atubakoksal/annotated tweets) [9]: a smaller corpus comprising approximately 2,000 annotated Turkish tweets, containing various informal linguistic phenomena.
3. Spellchecker Dataset (pnr-svc/spellchecker-dataset) [10]: a large-scale collection containing nearly 9 million words, targeting spelling correction across different Turkish domains.
4. Turkish Spelling Dictionary (asimokby/Turkish-Spelling-Dictionary) [11]: a manually curated list of approximately 100,000 base word pairs (expanded to 700,000 through data augmentation) intended for basic orthographic correction tasks.
5. Turkish GPT GEC Dataset (asimokby/Turkish-GPT-GEC) [12]: a dataset containing around 80,000 synthetically generated sentence pairs created to simulate realistic Turkish grammatical errors at the sentence level.

Following a detailed evaluation, forth and fifth datasets were selected as the primary sources for training and evaluation. The Turkish Spelling Dictionary was extensively utilized in both Phase 1 and Phase 2 to develop the model's capacity for correcting isolated word-level errors. In Phase 1, a random subset of 150,000 word pairs was selected to train the model on basic orthographic correction tasks. To prevent catastrophic forgetting and maintain word-level correction ability in Phase 2, an additional set of 20,000 previously unseen word pairs from the same source was incorporated into the training data.

To address discourse-level correction in Phase 3, an additional dataset was constructed manually. A corpus of 5,000 long user complaint texts was collected from şikayetvar.com, a Turkish online consumer complaint platform. These texts naturally contained a rich variety of spelling mistakes, grammatical inconsistencies, punctuation errors, and discourse-level incoherence. To create aligned corrected versions of these texts, a custom crawling and post-processing pipeline was developed. As part of this pipeline, texts were automatically corrected using GPT-4o-mini with tailored prompt engineering strategies designed to maximize correction quality while preserving the original semantic structure. This process resulted in a high-quality paired dataset consisting of noisy (corrupted) and corrected versions of real-world long-form Turkish texts.

Word-Level Training. In the initial phase, the model was trained on a dataset consisting of 150,000 isolated incorrect-correct word pairs. These examples contained no context and served as direct lexical mappings. The primary goal of this phase was to establish a foundational understanding of:

- Basic orthographic errors (e.g., character omissions, replacements)
- Common Turkish spelling patterns and suffixation rules
- Capitalization and diacritic inconsistencies

The training data was shuffled and split into training and evaluation sets using a 90/10 ratio. No sentences or contextual information were introduced at this stage. This low-context setup allowed the model to focus exclusively on token-level transformations without distraction from broader grammatical structures.

Sentence-Level Integration and Lexical Retention. In the second phase, the training regime was expanded to include both sentence-level examples and additional unseen word pairs. Specifically:

- 20,000 new word pairs were added from the source dataset, with care taken to ensure no overlap with the Phase 1 subset. This served a dual purpose: reinforcing low-level correction capabilities, preventing catastrophic forgetting of previously learned patterns
- 80,000 sentence pairs were added, each containing one erroneous sentence and its corrected version. These examples introduced more complex linguistic phenomena, such as: subject-verb agreement errors, word order misplacements, incorrect suffix usage or omission, over- or under- generation of function words.

By blending token-level and sentence-level training, this phase enabled the model to bridge the gap between local corrections and syntactic structure awareness. Sentence-based examples helped the model understand grammatical constraints that operate across word boundaries, which are common in agglutinative languages like Turkish.

Discourse-Level and Multi-Sentence Context Training. The final phase targeted higher-order linguistic errors spanning multiple sentences or full paragraphs. The dataset included:

- 5,000 new sentences not seen in Phase 2, again introduced to maintain word and sentence-level retention while offering fresh error examples.
- 5,000 short paragraph-style texts, each consisting of two to three sentences. These were largely derived from real-world, user-generated texts, particularly customer complaints.

In this phase, the model encountered errors such as, lack of sentence-ending punctuation, missing capitalization at the beginning of new sentences, pronoun resolution issues across sentence boundaries, misuse of discourse connectives and punctuation across clauses. The goal of this phase was to condition the model to perform context-aware correction

at the discourse level, including, long-distance dependency resolution, coherence and cohesion repair, inter-sentential normalization. This phase most closely reflects the type of input the model would receive in a production setting, where errors may span several sentences and exhibit nested, context-sensitive patterns.

3.1.2. Financial Documents for BERT

For the domain-specific pretraining of the ModernBERT model, we collected a corpus of publicly available financial documents in Turkish. The dataset spans multiple sectors, including finance, automotive, energy, telecommunications, and tourism, and comprises various document types such as financial reports, legal and regulatory texts, financial terminology dictionaries, and periodic financial publications. In total, the corpus consists of 82 carefully curated documents. To prepare the data for pretraining, all documents were converted into plain text format and preprocessed to create structured inputs suitable for masked language modeling (MLM) [7]. This preprocessing step ensured the consistency and quality of the training data, enabling the model to effectively learn domain-specific linguistic patterns and financial terminology.

3.2. Correction

In this section, we describe the multi-phase training strategy designed for the task of orthographic and grammatical error correction in Turkish. Our approach follows a curriculum learning paradigm, where the model is exposed to progressively more complex linguistic structures—from isolated words to fully formed paragraphs—across three distinct training phases. This methodology aims to emulate human language acquisition patterns, whereby learners first internalize lexical rules before acquiring syntactic and discourse-level competencies.

3.2.1. Model Architecture and Design

We utilized the google/mt5-xl model as the backbone architecture for all training phases. MT5 is a multilingual variant of the T5 encoder-decoder model, pre-trained on a massive multilingual corpus. It is particularly suitable for our task due to its strong performance on both low-resource languages and generation-based tasks such as machine translation and grammatical correction. To enable efficient fine-tuning on large-scale data with limited computational resources, we employed a parameter-efficient fine-tuning strategy based on Low-Rank Adaptation (LoRA), coupled with quantization via the bitsandbytes library. Specifically:

- The model weights were loaded in 4-bit precision using NF4 quantization, significantly reducing memory usage.
- A LoRA adapter was applied to the query (q) and value (v) projection layers in both self-attention and encoder-decoder attention submodules.
- The adapter dimension (r) was set to 64, with a scaling factor (lora alpha) of 32, and dropout of 0.1.
- LoRA was applied without altering bias terms, preserving the original model initialization.

3.2.2. Curriculum-Based Multi-Phase Training Strategy

The model was trained in three sequential phases, each designed to incrementally increase the complexity and contextual requirements of the correction task. Rather than mixing all examples together from the outset, we adopted a curriculum-style training scheme to guide the model from simpler to more complex examples, reducing the risk of overfitting on long-form text before mastering lower-level correction patterns.

3.3. Financial BERT

There have been various advancements on the BERT model architecture, training methods and context length. Our Financial BERT model is based on the ModernBERT architecture [7], which integrates several recently introduced methods including Rotary Positional Embedding (RoPE) [14], Local- Global Alternating Attention [15], and Flash Attention [16].

Architectural improvements in ModernBERT include the the Rotary Positional Embeddings (RoPE) [17] and GeGLU activation function [18]. RoPE introduces performance improvements in short- and long-context language models that enables easily extending context size. We adopt the GeGLU activation function, which is a combination of Gated-Linear Units (GLU)-based activation function [19] and BERT's GeLU activation function [25].

Additionally, ModernBERT also has efficiency improvements such as alternating attention, unpadding and Flash Attention. As presented in the ModernBERT model architecture, we apply global attention every 3 layers. Alternating between local and global attention makes it efficient to model long context models [15]. In global attention, every token within a sequence attends to every other token, whereas in local attention, tokens only attend to each other within a small sliding window. We use a window size of 128 tokens for local attention. We also employ unpadding [20] for both training and inference, which removes semantically empty tokens. Flash Attention is a core component of modern transformer-based models that made it possible computing efficient attention kernels. ModernBERT uses Flash Attention 3 [21] for global attention layers and Flash Attention 2 [22] for the local attention layers.

We follow the two-stage training approach of the ModernBERT model. In the first stage, the model is pretrained with a maximum context length of 1024 tokens. In the second stage, the context length is extended to 8192 tokens. We use the Masked Language Modeling (MLM) objective and set the mask rate to 30 %. We use the StableAdamW [23] optimizer that is based on the AdamW optimizer [24]. Our model is based on the ModernBERT-base model that has 22 hidden layers, with a total number of 149 million parameters. Additionally, the model has a hidden size of 768 with a GLU expansion of 2304. and 12 attention heads. We use a BPE-based tokenization approach, trained on financial data that has a vocabulary size of 50368, a multiple of 64, to ensure optimal GPU utilization .

4. Experiments

This section presents the experimental evaluation of the proposed two-stage framework, focusing on the effectiveness of the spelling correction module and the domain-specific BERT model trained on financial text. To assess the practicality and scalability of our approach, we conducted experiments in both local computing environments and high-performance computing (HPC) systems provided by the EuroHPC infrastructure. In Section 4.1, we report the performance of the word-level correction system under different runtime settings, including training times and model accuracy comparisons between local servers and supercomputer clusters. In Section 4.2, we evaluate the Financial BERT model pretrained on financial documents. The goal of these experiments is not only to measure model effectiveness, but also to assess the impact of HPC infrastructure on training efficiency and scalability.

4.1. Word Correction

4.1.1. Experimental Setup Overview

To evaluate the effectiveness of the proposed multi-phase training strategy for Turkish grammatical and orthographic error correction, a series of controlled experiments were conducted. All phases were trained on a single NVIDIA A100 GPU (40 GB memory), providing sufficient computational resources for large-scale fine-tuning using memory-efficient techniques.

All experiments utilized the mT5-XL model, a multilingual encoder-decoder transformer architecture, which was fine-tuned using a combination of Low-Rank Adaptation (LoRA) and 4-bit quantization. Specifically, LoRA modules were integrated into the query and value projection layers of both the self-attention and encoder-decoder attention mechanisms.

Training was performed using the paged AdamW optimizer with 32-bit weight precision. Across all phases, the initial learning rate was set to 5e-4, and a weight decay coefficient of 0.01 was applied to regularize the model parameters and prevent overfitting. Each phase was trained for 10 epochs. In every phase, a 90/10 split was used to separate the training and validation datasets. To facilitate smoother convergence and mitigate the risk of early overfitting, we applied a custom learning rate scheduler that multiplied the learning rate by a factor of 0.2 every two epochs. This gradual decay of the learning rate allowed the model to perform aggressive updates in the early stages of training while refining its parameters more delicately towards the end. All training phases utilized gradient checkpointing to further reduce memory consumption and employed bfloat16 mixed-precision training to accelerate computations without significant loss of numerical stability.

During training, model performance was evaluated at the end of each epoch using loss, Character Error Rate (CER), and Word Error Rate (WER) metrics. These evaluation metrics were chosen to provide a comprehensive view of the model's correction capabilities, both at the character level and at the semantic structure level. The following subsections present detailed descriptions of the training configurations and outcomes for each phase individually.

4.1.2. Phase 1 – Word Level Training

The first phase of training focused exclusively on isolated word-level error correction. A total of 150,000 corrupted-corrected word pairs were randomly sampled from the Turkish Spelling Dictionary dataset. These examples contained no sentence context, ensuring that the model concentrated solely on character-level transformations and morphological regularities typical of Turkish orthography.

Given the short length of individual words, the maximum input and output sequence length was set to 16 tokens, which allowed for highly efficient batch processing. The objective of this phase was to enable the model to acquire a robust foundation in correcting basic orthographic errors such as character omissions, substitutions, and common suffixation patterns, as well as resolving diacritic inconsistencies and capitalization errors.

4.1.3. Phase 2 – Sentence Level Augmentation

In the second phase of training, the model was exposed to more complex linguistic structures by extending the training data beyond isolated words to include full sentences. The dataset for this phase consisted of a mixture of 20,000 additional incorrect-correct word pairs and 80,000 corrupted-corrected sentence pairs. This augmentation aimed to enable the model to perform corrections not only at the token level but also within larger syntactic contexts, including subject–verb agreement, word order, suffix attachment, and function word usage.

Given the increased complexity and sequence length of the examples, the maximum tokenization length was raised to 32 tokens, allowing the model to capture broader contextual dependencies within each sentence. To accommodate this longer sequence structure without exceeding memory constraints, the effective batch size was adjusted to 128 tokens per device, and gradient accumulation was set to 8 steps to maintain an equivalent global batch size.

4.1.4. Phase 3 – Discourse Level and Multi-Sentence Context Training

In the third and final phase, the model was exposed to extended discourse-level structures, enabling it to learn corrections across multiple sentences and short paragraphs. The training data consisted of 5,000 additional corrupted-corrected sentence pairs (which were distinct from those used in Phase 2) and 5,000 corrupted-corrected short text passages, each containing two to three sentences. These examples included a variety of higher-order errors such as missing sentence boundaries, incorrect capitalization at the beginning of new sentences, and inconsistencies in pronoun use across adjacent clauses.

To adequately capture these extended linguistic contexts, the maximum sequence length during tokenization was increased to 512 tokens. Given the substantial growth in input size, the per-device training batch size was reduced to 12, and the per-device evaluation batch size to 16, while gradient accumulation was set to 16 steps to maintain an effective large batch size without exceeding memory constraints. Training in Phase 3 was conducted over 10 epochs, with the optimization and regularization strategies consistent with the previous phases. However, due to the significantly larger input sequences and the complexity of discourse-level corrections, convergence occurred more gradually.

In Phase 1, which involved isolated word-level correction, the model achieved a best validation loss of 0.1991, a best Word Error Rate (WER) of 0.2461, and a Character Error Rate (CER) of 0.0608. This phase primarily helped the model to internalize basic orthographic patterns, suffixation rules, and diacritic usage prevalent in Turkish. Upon integrating sentence-level examples in Phase 2, a substantial improvement was recorded. The best validation loss dropped to 0.1447, WER decreased to 0.1194, and CER to 0.0310. These results demonstrated that the model successfully adapted to syntactic structures, mastering not only token-level but also phrase- and clause-level correction tasks.

Finally, Phase 3, which introduced multi-sentence and paragraph-level contexts, led to the best overall performance. The model attained a minimum validation loss of 0.0223, a WER of 0.0619, and a CER of 0.0157. As also evidenced in Table 1, this phase confirmed the model's ability to handle discourse-level phenomena such as sentence boundary errors, inter-sentential coherence issues, and long-range dependency corrections.

The cumulative training time across all phases amounted to approximately 12 hours and 45 minutes. Despite the increase in input complexity and sequence length at each stage, the model maintained stable learning dynamics and achieved increasingly lower error rates, highlighting the effectiveness of the curriculum learning approach employed in this work.

As part of our high-performance experimentation, we leveraged the MeluXina supercomputer located in Luxembourg, under the EuroHPC initiative. The experiments were conducted on multi-GPU A100 servers, enabling efficient

Table 1. Evaluation results of spell correction training

Granularity Level	WER (%)	CER (%)	Training Time
Word (ours - 10 epoch)	0.2461	0.0608	1h 19m
Sentence (ours - 10 epoch)	0.1184	0.0299	2h 32m
Paragraph (ours - 10 epoch)	0.0619	0.0157	8h 54m
Word (meluxina - 20 epoch)	0.3922	0.0923	1h 5m 56s
Sentence (meluxina - 20 epoch)	0.3644	0.0665	13m 58s
Paragraph (meluxina - 20 epoch)	0.1917	0.1238	33m 25s

and scalable training of our spelling correction model across three levels of textual granularity: word, sentence, and paragraph.

To maintain consistency and comparability, we used the same hyperparameters as in our local experiments. The primary objective was to reduce training time through parallelization, while ensuring that the trained models could be seamlessly transferred and re-used on private infrastructure, particularly for applications involving sensitive financial data in banking environments. The results from the supercomputer-based training sessions are summarized in Table 1 below.

As shown in the table, sentence-level training yielded the lowest CER, while paragraph-level training achieved the best WER, suggesting that larger context windows help the model learn word relationships more effectively. On the other hand, word-level training offered faster runtime performance due to smaller input sizes and lower computational load. These results validate the suitability of MeluXina's HPC infrastructure for large-scale, multi-GPU training. The models trained in this setup can be efficiently ported to internal environments, allowing secure fine-tuning on institution-specific, proprietary datasets without compromising performance or data privacy.

Discussion of MeluXina Results.. Interestingly, the models trained on the MeluXina supercomputer exhibited higher Word Error Rate (WER) and Character Error Rate (CER) values compared to the locally trained models, despite using identical hyperparameters. For instance, sentence-level training yielded a WER of 11.84% on the local setup versus 36.44% on MeluXina. This discrepancy may be attributed to differences in data shuffling, tokenizer handling, or gradient synchronization behavior inherent in multi-GPU distributed training environments. Furthermore, the expected monotonic trend of improved performance from word- to paragraph-level training observed in the local setup was not consistently replicated on MeluXina. We hypothesize that this may result from increased noise accumulation or suboptimal convergence during distributed training, particularly when the effective batch size or communication efficiency is not fine-tuned per hardware topology.

These observations warrant further investigation and suggest that while HPC infrastructure enables rapid training, care must be taken to ensure reproducibility and stability across different hardware and parallelization regimes. Future work will explore more robust distributed training strategies and dynamic batch scaling to mitigate such discrepancies. Although we carefully ensured identical software stacks and hardware configurations between the local and HPC environments, the significant performance gap highlights the complexity of distributed training on large-scale HPC systems and the need for environment-specific optimization and validation procedures.

4.2. Financial BERT

For the domain-adapted language modeling stage, we trained a Turkish version of ModernBERT by continuing pretraining on our curated corpus of financial documents using the Masked Language Modeling (MLM) objective. We followed the ModernBERT methodology and adopted a masking rate of 30%, which has been shown to improve performance in domain adaptation scenarios by exposing the model to a wider variety of masked token combinations. Our training loss saturated after 80 epochs. Throughout the training process, we monitored the model's loss convergence and ensured stability through a combination of learning rate scheduling and gradient clipping. We also used a drop-out value of 0.1 to prevent overfitting with limited training data.

In our experiments, we used the Medium sized Turkish BERT model as the baseline [27]. The baseline model has 8 attention heads, 4 hidden layers and 42.2M parameters. Since the baseline model is trained on crawled data, it inher-

ently may contain various biases and may fail to capture domain-specific terminology. To evaluate the performance of our models, we selected the mask prediction task that evaluates a model's ability to decipher context. Our proposed ModernBERT-based model achieved 83.44 % accuracy on the masked set, whereas the baseline model achieved 61.32 % accuracy. The final pretrained ModernBERT model captures domain-specific terminology and syntactic structures common to financial documents, which we observed to improve downstream performance in tasks such as named entity recognition (NER), question answering (QA), and document classification (to be explored in future work).

This pretraining phase serves as the foundation for secure, institution-specific fine-tuning, allowing the model to be further adapted on proprietary banking data without the need to re-train from scratch. The resulting ModernBERT model is optimized for real-world financial language understanding tasks, offering both scalability and adaptability for private-sector deployment.

The training was carried out on the MeluXina supercomputer using A100 GPUs, allowing us to efficiently process large-scale financial texts. ModernBERT uses larger context size than BERT, where attention matrix calculation complexity increases with sequence length and attention head dimension. ModernBERT employs Flash Attention 2 and 3 for handling attention calculations for long sequences, whose effectiveness depends on the hardware it runs on. In our case, A100 GPUs were highly optimized for Flash Attention.

5. Conclusion

The experimental results demonstrate the effectiveness and scalability of our two-stage approach for enhancing natural language understanding in financial applications. Training on the local server provided strong baseline performance; however, the use of the MeluXina supercomputer significantly accelerated the training process, especially as the input granularity increased. Notably, while sentence- and paragraph-level datasets contain more tokens per sample compared to word-level datasets, the total number of steps required for convergence decreases, leading to shorter overall training times on high-performance computing (HPC) infrastructure. This behavior can be attributed to the richer contextual information available in larger text units, which enables faster and more efficient learning by the model. Furthermore, by following a curriculum learning strategy—from words to sentences to paragraphs—we effectively emulate natural language acquisition patterns, allowing the model to progressively internalize lexical, syntactic, and discourse-level knowledge.

In the second stage, our FinBERT model—trained using a Masked Language Modeling (MLM) setup with a 30% masking rate, consistent with the ModernBERT methodology—successfully adapted general BERT architectures to financial domain-specific tasks. The combination of domain-specific pretraining and supercomputing capabilities yielded models that are not only highly accurate but also efficiently trainable at scale.

Overall, these findings highlight the critical role of HPC resources in accelerating model development for specialized domains, and demonstrate that structured curriculum learning, combined with domain adaptation strategies, offers a powerful framework for building high-quality natural language processing systems tailored for real-world financial applications. The models developed through this study are ready for further fine-tuning on sensitive, institution-specific datasets, enabling secure, private deployment without compromising on performance.

However, we acknowledge that this work does not provide a comprehensive treatment of security and data privacy aspects associated with deploying NLP models in production financial environments. While the proposed architecture is compatible with local deployment and private fine-tuning, critical challenges such as model leakage, secure inference, and regulatory compliance remain unaddressed. We identify this as an important direction for future work, potentially involving techniques such as federated learning, differential privacy, or encrypted model serving to enable robust and compliant deployment pipelines.

Although our current pipeline focuses primarily on linguistic challenges such as spelling correction and domain adaptation, we acknowledge that quantitative reasoning remains an open challenge in financial NLP. Future extensions may integrate numerical reasoning modules or leverage hybrid architectures that combine language models with symbolic computation components [e.g., LILA, Toolformer, etc.], especially for tasks such as portfolio analysis or risk estimation.

Acknowledgements

The authors gratefully acknowledge the use of the MeluXina supercomputer hosted by LuxProvide, Luxembourg, as part of the European High Performance Computing Joint Undertaking (EuroHPC JU) infrastructure. Access to this HPC resource has been instrumental in conducting the large-scale experiments presented in this study.

References

[1] Pajak, K., Pajak, D. (2022). Multilingual fine-tuning for grammatical error correction *Expert Systems with Applications, 200*, 116948.
[2] Stankevičius, L., Lukoševičius, M., Kapočiūtė-Dzikienė, J., Briedienė, M., Krilavičius, T. (2022). Correcting diacritics and typos with a ByT5 transformer model. *Applied Sciences, 12(5)*, 2636.
[3] Yang, Y., Uy, M. C. S., Huang, A. (2020). Finbert: A pretrained language model for financial communications. *arXiv preprint* arXiv:2006.08097.
[4] Peng, B., Chersoni, E., Hsu, Y. Y., Huang, C. R. (2021). Is domain adaptation worth your investment? comparing BERT and FinBERT on financial tasks. Association for Computational Linguistics (ACL).
[5] Kim, E., Shin, H. (2022). KR-FinBert: Fine-tuning KR-FinBert for sentiment analysis. *Github. com*.
[6] Maharani, N. P. I., Purwarianti, A., Yustiawan, Y., Rochim, F. C. (2023, October). Domain-Specific Language Model Post-Training for Indonesian Financial NLP. In *2023 International Conference on Electrical Engineering and Informatics (ICEEI)* (pp. 1-6). IEEE.
[7] Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Cooper, N., Adams, G., Howard, J. & Poli, I. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. (2024), https://arxiv.org/abs/2412.13663
[8] Atakan Kara, Farrin Marouf Sofian, Andrew Bond, and Gözde Gül Şahin. (2023). GECTurk: Grammatical Error Correction and Detection Dataset for Turkish. arXiv:2309.11346
[9] Atuba Koksal. (2024). annotated tweets [Dataset]. GitHub. (Accessed: 28 April 2025).
[10] pnr-svc. (2024). spellchecker-dataset [Dataset]. Hugging Face. (Accessed: 28 April 2025).
[11] Asimokby. (2024). Turkish-Spelling-Dictionary [Dataset]. Hugging Face. (Accessed: 28 April 2025).
[12] Asimokby. (2024). Turkish-GPT-GEC [Dataset]. Hugging Face. (Accessed: 28 April 2025).
[13] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. (2021). mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 483–498, Online. Association for Computational Linguistics.
[14] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B. & Liu, Y. RoFormer: Enhanced Transformer with Rotary Position Embedding. (2023), https://arxiv.org/abs/2104.09864
[15] Team, G., Riviere, M., Pathak, S., Sessa, P., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al., Gemma 2: Improving Open Language Models at a Practical Size. (2024), https://arxiv.org/abs/2408.00118
[16] Dao, T., Fu, D., Ermon, S., Rudra, A. & Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. (2022), https://arxiv.org/abs/2205.14135
[17] Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U., Purohit, S., Reynolds, L., Tow, J., Wang, B. & Weinbach, S. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. (2022), https://arxiv.org/abs/2204.06745
[18] Shazeer, N. GLU Variants Improve Transformer. (2020), https://arxiv.org/abs/2002.05202
[19] Dauphin, Y., Fan, A., Auli, M. & Grangier, D. Language Modeling with Gated Convolutional Networks. (2017), https://arxiv.org/abs/1612.08083
[20] Zeng, J., Li, M., Wu, Z., Liu, J., Liu, Y., Yu, D. & Ma, Y. Boosting Distributed Training Performance of the Unpadded BERT Model. (2022), https://arxiv.org/abs/2208.08124
[21] Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P. & Dao, T. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. (2024), https://arxiv.org/abs/2407.08608
[22] Dao, T. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. (2023), https://arxiv.org/abs/2307.08691
[23] Wortsman, M., Dettmers, T., Zettlemoyer, L., Morcos, A., Farhadi, A. & Schmidt, L. Stable and low-precision training for large-scale vision-language models. (2023), https://arxiv.org/abs/2304.13013
[24] Loshchilov, I. & Hutter, F. Decoupled Weight Decay Regularization. (2019), https://arxiv.org/abs/1711.05101
[25] Hendrycks, D. & Gimpel, K. Gaussian Error Linear Units (GELUs). (2023), https://arxiv.org/abs/1606.08415
[26] European High-Performance Computing Joint Undertaking (EuroHPC). TrustLLM: Large Language Model Technology for Europe. Accessed: April 29, 2025.
[27] Kesgin, H., Yuce, M. & Amasyali, M. Developing and Evaluating Tiny to Medium-Sized Turkish BERT Models. *ArXiv Preprint ArXiv:2307.14134*. (2023)

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 187–196

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

An accelerated implementation of Extended Cellular Potts Model for tumor angiogenesis simulations

Luigi D'Onofrio[a], Pasquale De Luca[a,*], Anna Greco[a], Livia Marcellino[a]

[a]*Department of Science and Technology, Parthenope University of Naples, Naples, Centro Direzionale Isola C4, 80143*

Abstract

Modern distributed systems and neural computation face increasing demands in processing complex simulations, particularly in computational biology. While distributed computing offers powerful solutions for large-scale problems, certain computational challenges can be effectively addressed through GPU acceleration alone, providing a foundation for future distributed implementations. In this work, we present a GPU-parallel approach for simulating angiogenesis using the Cellular Potts Model (CPM). Here we deal with how a traditionally sequential biological simulation can be transformed into a parallel implementation, establishing a methodology that could be extended to distributed systems. By implementing the CPM on GPU using CUDA, and incorporating both chemotaxis and electric field effects, we extend the basic angiogenesis model by introducing a tumor-related chemical source, thus making a more detailed and biologically relevant simulation environment, we develop a framework that achieves significant performance improvements while maintaining biological accuracy. Experiments conducted on LEONARDO supercomputer, show up to 39× speedup compared to CPU implementations, particularly in handling large-scale matrices and complex energy calculations.

Keywords: Big-data simulations, angiogenesis, parallel algorithms, GPU computing, Leonardo Super-computer

1. Introduction

Modern computational challenges in neural networks and distributed systems often are related to the efficient processing of complex, large-scale simulations. As computational biology continues to advance, the demand for high-performance computing solutions has become increasingly critical, particularly in simulating complex biological phenomena [1, 2]. Among these phenomena, angiogenesis, the formation of new blood vessels from pre-existing ones, represents a compelling case study in computational complexity and resource requirements. The intersection of distributed computing, neural computation, and biological system modeling presents unique opportunities for advancing our understanding of complex physiological processes [3, 4]. While distributed systems have traditionally focused on partitioning computational loads across multiple nodes [5], certain biological simulations, such as angiogenesis

* Corresponding author. Tel.: +39-345-4150-455.
E-mail address: pasquale.deluca@uniparthenope.it

1877-0509
10.1016/j.procs.2025.08.245

modeling, can benefit significantly from computational accelerators [6, 10, 7, 8, 9, 29] as a foundational step toward future distributed implementations [11]. Angiogenesis plays an important role in both physiological and pathological conditions, particularly in cancer development, where it contributes to tumor growth and metastasis. The computational modeling of this process requires sophisticated mathematical frameworks capable of capturing multiple physical and chemical interactions [12]. The Cellular Potts Model (CPM), a lattice-based Monte Carlo method, has emerged as a powerful tool for simulating such cellular behaviors, but its computational intensity has historically limited its application to smaller-scale systems. Traditional CPU-based implementations of the CPM face significant performance limitations, particularly when simulating large-scale systems. As the lattice size increases, the computational requirements grow quadratically, making large-scale simulations impractical on conventional CPU architectures. In this work, we address these computational challenges through a GPU-parallel implementation of the CPM, focusing on efficient parallel processing of the core components of the model. Moreover, the mathematical framework of our implementation extends the traditional CPM by incorporating additional field terms that account for both chemical and electrical influences on cell behavior. These extensions allow for more realistic biological simulations by capturing the complex interplay between cells and their environment. Our model considers how cells respond to chemical gradients through chemotaxis and how electric fields influence cellular migration, providing a more comprehensive representation of biological reality.

The paper is organized as follows: Section 2 presents the mathematical framework and theoretical background of the problem, including the detailed formulation of the Cellular Potts Model and its sequential algorithmic implementation. Section 3 describes our GPU-parallel approach, detailing the optimization strategies and implementation techniques used to accelerate the simulation. Section 4 presents experimental results, providing a thorough analysis of the performance improvements achieved through GPU parallelization compared to the sequential implementation. Finally, Section 5 closes the paper with conclusions.

2. Mathematical framework

In this section, we present the mathematical framework of the extended Cellular Potts Model. Starting from the basic CPM formulation [13] and taking account the mathematical background presented in [14], we detail the addition of chemotaxis and electric field terms to capture complex cell behavior in tumor angiogenesis. Let $\Omega \subset \mathbb{Z}^d$ be a d-dimensional lattice representing the spatial domain where cells evolve, with d typically is equal to 2 or 3 for biological applications. At each lattice site $x \in \Omega$, we assign a cell index $\sigma(x) \in \{0, 1, \ldots, N\}$, where 0 denotes the extracellular medium and positive integers represent distinct cells. Each cell is characterized by its type $\tau(\sigma)$, allowing for multiple cell populations within the system.

The Cellular Potts Model describes the evolution of cellular configurations through the minimization of a Hamiltonian energy function H. In our extended formulation, the total energy of the system is given by:

$$H = H_{\text{ad}} + H_{\text{v}} + H_{\text{s}} + H_{\text{chemo}} + H_{\text{electric}}, \tag{1}$$

where H_{ad} quantifies adhesion interactions between neighboring cells, H_v represents volume constraint energy, H_s represents surface area constraint energy, H_{chemo} captures chemotactic response energy, and $H_{electric}$ models electric field influence energy. The adhesion term H_{ad} quantifies the interaction energy between neighboring cells:

$$H_{\text{ad}} = \sum_{x} \sum_{x' \in \mathcal{N}(x)} J\left(\tau(\sigma(x)), \tau(\sigma(x'))\right)\left(1 - \delta(\sigma(x), \sigma(x'))\right) \tag{2}$$

where $\mathcal{N}(x)$ represents the neighborhood of site x, $J(\tau_1, \tau_2)$ denotes the adhesion energy between cells of types τ_1 and τ_2, and δ is the Kronecker delta function. Cell volume and surface area are constrained through quadratic energy terms:

$$H_{\text{v}} = \lambda_V \sum_{\sigma>0} \left(V(\sigma) - V_{\text{target}}(\sigma)\right)^2, \quad H_{\text{s}} = \lambda_S \sum_{\sigma>0} \left(S(\sigma) - S_{\text{target}}(\sigma)\right)^2 \tag{3}$$

where $V(\sigma)$ and $S(\sigma)$ are the current volume and surface area of cell σ, $V_{target}(\sigma)$ and $S_{target}(\sigma)$ are their respective target values, and λ_V , λ_S are Lagrange multipliers controlling the strength of these constraints. The chemotactic response is incorporated through:

$$H_{\text{chemo}} = -\lambda_C \sum_x \sum_i \mu_i \left(\tau(\sigma(x))\right) c_i(x) \tag{4}$$

where λ_C is the chemotaxis strength parameter, $\mu i(\tau)$ represents the chemotactic sensitivity of cell type τ to chemical species i, and $c_i(x)$ denotes the concentration of chemical i at position x. The electric field contribution is modeled as:

$$H_{\text{electric}} = -\lambda_E \sum_x \mathbf{E}(x) \cdot \mathbf{x} \tag{5}$$

where λ_E controls the strength of the electrical response and $\boldsymbol{E}(x)$ is the electric field vector at position x. The electric field is derived from a potential ϕ satisfying the Poisson equation:

$$\nabla^2 \phi = -\rho \tag{6}$$

with charge density $\rho(x) = \alpha c(x)$, relating the electrical properties to chemical concentrations through the proportionality constant α. The system evolves through a Monte Carlo algorithm with Metropolis dynamics. For each attempted update, the probability of accepting a change in cell index from $\sigma(x)$ to $\sigma(x')$ is:

$$P(\sigma(x) \to \sigma(x')) = \min(1, e^{-\Delta H/T}) \tag{7}$$

in which ΔH is the total energy change and T is a temperature parameter controlling fluctuations. The chemical fields evolve according to reaction-diffusion dynamics:

$$\frac{\partial c_i}{\partial t} = D_i \nabla^2 c_i + R_i(c_1, \ldots, c_n, \sigma). \tag{8}$$

Here, D_i represents the diffusion coefficient of species i and R_i includes reaction terms. The electric field components are computed through central differences:

$$E_x(i,j) \approx -\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x}, \quad E_y(i,j) \approx -\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta x}. \tag{9}$$

In order to compute the electric field, we solve a linear system based on the current state of the cells and the chemical field. We assume that the electric potential ϕ satisfies a Poisson equation:

$$\nabla^2\phi = -\rho$$

where ρ is a charge density that we relate to the chemical concentration and cell distribution. For simplicity, we assume ρ is proportional to the chemical concentration c:

$$\rho(x) = \alpha c(x)$$

where α is a proportionality constant. On our lattice, we discretize the Laplacian operator using the standard 5-point finite difference stencil method:

$$\nabla^2\phi_{i,j} \approx \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}}{(\Delta x)^2}$$

where Δx is the lattice spacing. This leads to a system of linear equations:

$$\mathbf{A}\boldsymbol{\phi} = \mathbf{b}. \tag{10}$$

In detail, $\boldsymbol{A}$ is a sparse matrix representing the discretized Laplacian, ϕ is the vector of electric potential values at each lattice site, and $\boldsymbol{b}$ is the vector of charge density values (scaled by $-(\Delta x)^2$). Once the electric potential ϕ is obtained by solving the linear system in Eq. (10), we compute the electric field $\boldsymbol{E}$ as the negative gradient of ϕ:

$$\mathbf{E} = -\nabla\phi.$$

The electric field affects cell movement through the $H_{electric}$ term in the Hamiltonian. The energy change for a proposed cell index copy from site x to x' due to the electric field is: $\Delta H_{electric} = -\lambda_E[\boldsymbol{E}(x') \cdot x' - \boldsymbol{E}(x) \cdot x$.

This energy change is incorporated into the Metropolis algorithm along with the other energy terms, influencing the probability of accepting cell movements. The algorithm runs the system through a series of Monte Carlo steps, where each step involves updating the cell configuration based on the energy minimization principle. To simulate the evolution of the system over time, we employ a time-stepping loop in which the Metropolis algorithm is executed at each time step. Let t denote the current time, the system evolves iteratively over a sequence of time steps $t = t_0, t_1, \ldots, t_{max}$. At each time step t, the following steps are performed:

STEP 1 *Monte Carlo Steps*: A fixed number of Monte Carlo steps N_{MC} are executed to update the cell configuration. Each Monte Carlo step involves proposing a change to the cell configuration and accepting or rejecting it based on the Metropolis criterion:

$$P(\sigma(x) \rightarrow \sigma(x')) = min(1, e^{-\Delta H/T}), \tag{11}$$

where ΔH is the change in the Hamiltonian energy due to the proposed update. The total energy change ΔH is computed as the sum of the individual energy changes:

$$\Delta H = \Delta H_{\text{ad}} + \Delta H_{\text{v}} + \Delta H_{\text{s}} + \Delta H_{\text{chemo}} + \Delta H_{\text{electric}}, \tag{12}$$

STEP 2 *Chemical Dynamics*: The chemical concentrations $c_i(x)$ are updated using reaction-diffusion dynamics:

$$\frac{\partial c_i}{\partial t} = D_i \nabla^2 c_i + R_i(c_1, \ldots, c_n, \sigma).$$

STEP 3 *Electric Field Update*: The electric potential ϕ and electric field $\boldsymbol{E}$ are recomputed by solving (10).

This iterative process continues until the final time t_{max} is reached, allowing the system to evolve dynamically under the influence of mechanical, chemical, and electrical interactions. Previous procedures are compactly summarized in the following algorithm:

Algorithm 1 Extended Cellular Potts Model

Initialize lattice Ω, cell indices σ(x) and chemical concentrations $c_i(x)$
Compute ϕ and electric field $\boldsymbol{E}$ as in Eq. (10)
for $t = t_0, t_1, \ldots, t_{max}$ **do**
    for each Monte Carlo step $k = 1, 2, \ldots, N_{MC}$ **do**
        Select a random lattice site x and x'
        Compute ΔH % as in Eq. (12)
        Accept the change % as in Eq. (11)
    end for
    `update` chemical concentrations $c_i(x)$
    `solve` the linear system as in Eq. (10)
end for

The sequential implementation of the Algorithm 1 relies on several specialized libraries to handle different computational aspects of the simulation. For solving the linear systems arising from the electric field calculations, we employ both LAPACK and SuiteSparse libraries [16][17]. LAPACK (Linear Algebra Package) is utilized for dense matrix operations, particularly through the LAPACKE_dgesv function, which performs LU factorization with partial pivoting for solving systems of linear equations. However, as the lattice size increases, the matrix representing the discretized Laplacian becomes increasingly sparse, making dense matrix approaches computationally inefficient. To address this limitation, we integrate the SuiteSparse library, specifically its CSparse component, which provides efficient sparse matrix operations. The sparse matrix is initially constructed in triplet format using `cs_spalloc` and `cs_entry` functions, then converted to compressed-column format via `cs_compress` for optimal computational efficiency. The solution of the linear system involves symbolic factorization through `cs_sqr`, followed by numerical factorization using `cs_lu`, with the final solution obtained through a combination of `cs_ipvec`, `cs_lsolve`, and `cs_usolve` operations. In order to handle the reaction-diffusion dynamics of chemical fields, we implement finite difference methods using standard numerical libraries. The Monte Carlo steps, which form the core of the CPM evolution, are implemented using a sequential random number generator for the Metropolis acceptance criterion. The algorithm maintains separate data structures for the lattice configuration, chemical concentrations, and electric field components, with updates performed sequentially at each time step. This implementation, while functionally complete, faces performance limitations when dealing with large-scale simulations, particularly in the computation of energy terms and the resolution of field equations, motivating the need for parallel implementation strategies.

3. Parallel approach

The computational complexity of simulating angiogenesis using the Extended Cellular Potts Model presents significant challenges when executed sequentially, particularly for large-scale systems. For a standard simulation domain of 200×200 lattice sites, the algorithm processes 40,000 individual sites ($N = 200$) and manage complex energy calculations during each Monte Carlo step. The additional overhead of chemical field diffusion and electric field computations makes real-time simulation computationally prohibitive using traditional sequential approaches. In order to address these issues and enable practical large-scale simulations, we developed a parallel implementation leveraging the Single Instruction Multiple Data (SIMD) architecture of Graphics Processing Units (GPUs) by means of CUDA framework [18]. Our parallel implementation mimics the core Algorithm 1 while maintaining the biological accuracy of the simulation. The approach distributes computational tasks across thousands of concurrent threads, with each thread handling calculations for individual lattice sites. We identified several key components amenable to parallelization: energy calculations, chemical field updates, and electric field computations. The parallel architecture enables simultaneous processing of multiple lattice sites, significantly reducing the overall computation time. The first optimization involves restructuring data representations to align with GPU memory access patterns. The core simulation routine, for initializing cells, handles the parallel initialization of the lattice structure. This routine implements a transformation from the sequential CPM algorithm described by Glazier and Graner [13] to a parallel framework. The function operates by mapping thread blocks to lattice regions, with each thread processing an unique lattice site given by the transformation $\mathcal{T}: (i,j)7 \rightarrow z$, where $z = i \cdot N + j$ for a lattice of size $N \times N$. Following the initialization, the chemical updating procedure manages the parallel evolution of chemical fields. This kernel implements the discretized reaction-diffusion equations through a parallel finite difference scheme. The computational domain is partitioned into blocks of 256 threads per block threads, chosen to optimize occupancy on current GPU architectures as demonstrated in [15]. The parallel electric field computation is handled by building and solving the sparse linear system arising from the discretization of the Poisson equation. This routine leverages cuDSS library [21], employing techniques developed in [19] for efficient sparse matrix operations on GPUs. The system matrix $\boldsymbol{A} \in \mathbb{R}^{N^2 \times N^2}$ maintains its sparsity pattern throughout the simulation, allowing for optimized memory access patterns and computational workflows. The Monte Carlo step implementation maintains the statistical validity of the Metropolis algorithm while exploiting parallel computation. Following the approach of [20], the kernel utilizes the cuRAND library [23] for parallel random number generation, ensuring statistical independence across threads through the maintenance of separate random states $\{\xi_i\}_{i=1}^{N^2}$ for each lattice site. The chemical field normalization process is implemented through ad-hoc CUDA kernels. These functions employ a two-phase approach using parallel reduction operations from the CUB library [22]. The maximum chemical value c_{max} is computed through a parallel reduction operation $\mathcal{R} : \mathbb{R}^{N^2} \rightarrow \mathbb{R}$ defined by:

$$\mathcal{R}(\{c_i\}_{i=1}^{N^2}) = \max_{i \in [1,N^2]} c_i. \tag{13}$$

The parallel algorithm can be expressed formally as follows:

Algorithm 2 Parallel CPM Implementation

```
Initialize GPU memory and transfer initial lattice state
Configure CUDA streams and events for asynchronous operation
for each Monte Carlo step do
    compute parallel energy calculation kernel
    compute chemical field update kernel
    solve electric field system using cuDSS
    compute parallel field normalization
    if visualization step then
        Asynchronously transfer current state to host
    end if
end for
Finalize and release GPU resources
```

Table 1: Execution time comparison (seconds).

Implementation	$N = 200$	$N = 400$	$N = 800$
Sequential C	64.60	1101.09	18289.29
CUDA	6.11	47.64	437.95

Thread block configurations are optimized based on empirical testing by providing optimal occupancy across various problem sizes as follows: threads_per_block = 256.

For the chemical field update kernel, we employ a 2D block configuration that better matches the spatial locality of the computation. In detail, the grid configuration is adapted to the lattice dimensions:

$$\text{blocks\_per\_grid} = \left\lceil \frac{N^2}{\text{threads\_per\_block}} \right\rceil. \tag{14}$$

The electric field solver engages cuDSS for optimal performance, with the matrix structure:

$$\mathbf{A} = \begin{bmatrix} -4 & 1 & 0 & \cdots & 1 \\ 1 & -4 & 1 & \cdots & 0 \\ 0 & 1 & -4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & -4 \end{bmatrix}_{N^2 \times N^2}. \tag{15}$$

This structured sparsity pattern enables efficient parallel solution strategies. The solver pipeline consists of symbolic analysis, numerical factorization, and solution phases, all executed on the GPU to minimize data movement. Synchronization between different computational phases is managed through CUDA events and streams, enabling overlap of computation and communication where possible.

4. Results

The experimental evaluation of the proposed GPU-parallel implementation will be shown in this section. The test are conducted on the supercomputer LEONARDO (CINECA) equipped by: Intel Xeon Platinum 8358 CPU, 2.60GHz (Ice Lake) and NVIDIA custom Ampere A100 GPU 64GB HBM2e, NVLink 3.0 (200GB/s) and, CUDA toolkit version 12.3 [24].

We conducted performance analyses across three distinct simulation scenarios by increasing the computational complexity to confirm efficiency of the proposed approach. The experimental framework employed a set of parameters among three simulation configurations. The first simulation utilized a 200×200 lattice with 10 cells and 100 Monte Carlo steps, representing a baseline case for performance comparison. The second configuration expanded to a 400×400 lattice containing 400 cells and 200 Monte Carlo steps, while the third and most demanding scenario employed an 800×800 lattice with 80 cells and 400 Monte Carlo steps. Throughout all simulations, the temperature parameter remained constant at 20 units, ensuring consistent stochastic behavior in the Monte Carlo algorithm.

The model parameters were carefully calibrated to maintain biological relevance, with volume constraint coefficient $\lambda_V = 50$, surface constraint coefficient $\lambda_S = 0.5$, chemotaxis coefficient $\lambda_C = 100$, and electric field coefficient $\lambda_E = 50$. The target volume for all cells was set to 500 lattice sites, a value chosen to ensure realistic cell sizes relative to the lattice dimensions. Performance measurements highlight improvements achieved through GPU parallelization. For the baseline 200×200 lattice simulation, execution time decreased from 64.6 seconds in sequential C to just 6.1 seconds in our CUDA implementation. The improvements become more pronounced for the intermediate 400×400 case, with execution times reducing from 18 minutes and 21 seconds to only 47.6 seconds. The largest simulation (800×800) showed the best improvement, with execution time decreasing from 5 hours and 4 minutes to approximately 7 minutes and 18 seconds.

Table 2: Component-wise execution times simulation in seconds (N = 200).

Module	Sequential C	CUDA
Cell Initialization	0.23	0.45
Linear System Solution	52.40	4.76
Electric Field Calculation	0.04	0.001
Chemical Field Update	2.62	0.70
Chemical Field Normalization	0.02	0.02
Total MCS Time	55.91	6.11

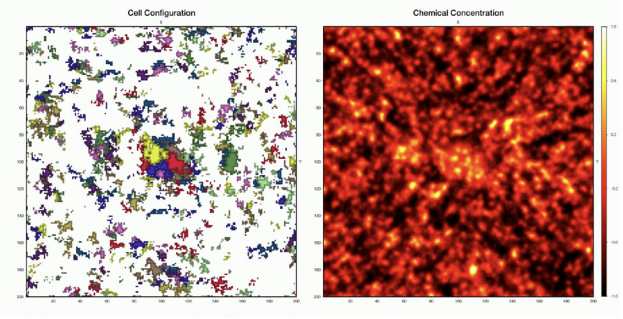


Fig. 1: Visualization of cell configuration (left) and chemical concentration field (right) from a CUDA simulation, demonstrating preserved biological accuracy in parallel implementation.

The baseline simulation achieved an increasing of performance in $9.145\times$, while the intermediate case demonstrated a $20.439\times$ improvement. The largest simulation exhibited the most significant gains with an acceleration factor of $39.338\times$. This scaling behavior with problem size highlights the effectiveness of our parallel implementation for large-scale simulations. Detailed analysis of individual computational components reveals varying degrees of performance improvement. For the baseline simulation, the linear system solution phase showed the most dramatic enhancement, reducing from 52.39 seconds in sequential C to 4.76 seconds in CUDA, representing an 11× speedup for this critical component. The chemical field update process similarly improved from 2.62 seconds to 0.70 seconds. Notably, the cell initialization phase showed marginally slower performance in CUDA (0.000455 seconds versus 0.000234 seconds in sequential C), because of the overhead of initial GPU memory allocation and transfer, though this cost becomes not useful in the context of total simulation time.

Let $\boldsymbol{s}^{CPU}(t) \in \mathbb{R}^m$ denote the complete system state vector computed by the CPU implementation at time step t, and $\boldsymbol{s}^{GPU}(t) \in \mathbb{R}^m$ denote the corresponding state vector from the GPU implementation, where m involves all system variables. More precisely, let q_i^{CPU} represent the i-th component of the CPU solution and q_i^{GPU} the corresponding GPU component. The numerical distance between implementations is quantified by the difference: $\delta_i = q_i^{GPU} - q_i^{CPU}$. We define error metrics as:

$$\| \boldsymbol{\delta} \|_\infty = \max_i |\delta_i|, \quad \| \boldsymbol{\epsilon}_{rel} \|_\infty = \max_i \left| \frac{\delta_i}{q_i^{CPU}} \right|.$$

Table 3 presents the complete numerical comparison for a representative simulation with lattice size $N = 200 \times 200$, $N_{cells} = 400$, and $t = 100$ Monte Carlo steps under deterministic conditions.

The numerical comparison in Table 3 shows that q_i^{CPU} and q_i^{GPU} exhibit differences δ_i, computed with same fixed seed, that are strictly bounded by machine precision limits. The observed differences arise from fundamental aspects of floating-point arithmetic in parallel computing environments. Here, $x_c^{(1)}$ and $x_c^{(2)}$ denote the $x-$coordinates the center of mass of cell, 1 and cell 2.

Table 3: CPU vs GPU comparison with exact floating-point values.

Quantity	**CPU Solution** q_i^{CPU}	**GPU Solution** q_i^{GPU}	**Difference** $\delta_i = q_i^{GPU} - q_i^{CPU}$
Cell center $x_c^{(1)}$ (lattice units)	127.4568	127.4568	0
Cell center $x_c^{(2)}$ (lattice units)	156.7890	156.7890	-1.1×10^{-15}
Cell volume $V^{(1)}$ (sites)	487	487	0
Cell surface area $S^{(1)}$ (sites)	94	94	0
Total energy H_{total}	−2027.4804	−2027.4804	-1.1×10^{-15}
Chemical conc. c(64, 64)	0.6789	0.6789	1.1×10^{-15}
Electric potential ϕ(128, 96)	-0.6789	-0.6789	2.2×10^{-16}
Global Error Metrics:			
Maximum absolute error: $\| \delta \|_\infty$			1.1×10^{-15}
Maximum relative error: $\| \epsilon_{rel} \|_\infty$			2.4×10^{-15}
Machine precision bound: $\epsilon_{machine}$			2.2×10^{-16}

The biological accuracy of our parallel implementation was validated through visual inspection of simulation results. Figure 1 exhibits a characteristic snapshot from a simulation, showing both cell configuration and chemical concentration fields. The cell patterns exhibit appropriate clustering and morphology consistent with biological expectations, while the chemical field shows the expected diffusion patterns and gradient formations. These qualitative observations confirm that the significant performance improvements were achieved without compromising the biological fidelity of the simulation. These results demonstrate that our GPU-parallel implementation not only accelerates existing feasible simulations but also enables the practical execution of previously intractable problem sizes. The achieved speedups, ranging from 9× to 39×, represent a significant advance in computational capability for CPM simulations, particularly beneficial for studying complex biological phenomena requiring larger domains or extended simulation times.

5. Conclusions

In this paper, we presented a GPU-parallel implementation of an Extended Cellular Potts Model for simulating angiogenesis, incorporating both chemotaxis and electric field effects. Our approach demonstrates how traditionally sequential biological simulations can be effectively transformed into parallel implementations. Through the use of CUDA, we achieved significant performance gains, showing up to 39× speedup compared to CPU implementations. The parallel implementation successfully handles the computational complexity of large-scale matrices and complex energy calculations inherent in the CPM simulation. Key improvements were realized through efficient thread block configurations, optimized memory access patterns, and the strategic use of cuDSS for sparse linear system solutions. The implementation maintains a clear separation between host and device operations, with careful management of data transfers and computational workflows. This architectural decision, combined with the use of specialized CUDA libraries, enables the simulation of previously intractable system sizes while preserving the stochastic nature of the Monte Carlo algorithm. These results open the possibility to predict and diagnosing tumor progression almost in a real time. Moreover, future directions include to integrate machine and deep learning models to further improve predictive capabilities by allowing more precise simulations.

Acknowledgment

We would like to thank EuroHPC and ISCRA for granting us access to the LEONARDO supercomputer, hosted by CINECA (Italy), which provided the computational resources used in this work. De Luca P. and Marcellino L. are member of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM). D'Onofrio L. author was supported by GNAMPA and by the European Union - NextGenerationEU within the framework of PNRR Mission 4 - "PRIN 2022" - grant number 2022BCFHN2 - Advanced theoretical aspects in PDEs and their applications CUP I53D23002300006.

References

[1] A. Ciaramella, E. Di Nardo, D. Terracciano, L. Conte, F. Febbraio, and A. Cimmino, "A new biomarker panel of ultraconserved long non-coding RNAs for bladder cancer prognosis by a machine learning based methodology," BMC Bioinformatics, vol. 23, 2023.

[2] P. De Luca and L. Marcellino, "Conservation Law Analysis in Numerical Schema for a Tumor Angiogenesis PDE System," Mathematics, vol. 13, no. 1, pp. 28, 2025.

[3] E. Di Nardo, A. Petrosino, and I. Ullah, "EmoP3D: A brain like pyramidal deep neural network for emotion recognition," Lecture Notes in Computer Science, vol. 11131 LNCS, pp. 607–616, 2019.

[4] A. La Ferlita, Y. Qi, E. Di Nardo, O. el Moctar, T. E. Schellin, and A. Ciaramella, "A Comparative Study to Estimate Fuel Consumption: A Simplified Physical Approach against a Data-Driven Model," Journal of Marine Science and Engineering, vol. 11, no. 4, 2023.

[5] A. Ciaramella, E. Di Nardo and G. Vettigli, "Recursive Learning Framework for Structured Data Agglomeration," 2024 International Joint Conference on Neural Networks (IJCNN), Yokohama, Japan, 2024, pp. 1-6.

[6] S. Cuomo, P. De Michele, E. Di Nardo, and L. Marcellino, "Parallel Implementation of a Machine Learning Algorithm on GPU," International Journal of Parallel Programming, vol. 46, no. 5, pp. 923–942, 2018.

[7] Conte, D., De Luca, P., Galletti, A., Giunta, G., Marcellino, L., Pagano, G., & Paternoster, B. (2022, July). First experiences on parallelizing peer methods for numerical solution of a vegetation model. In International Conference on Computational Science and Its Applications (pp. 384-394). Cham: Springer International Publishing.

[8] Cardone, A., De Luca, P., Galletti, A., & Marcellino, L. (2023). Solving Time-Fractional reaction–diffusion systems through a tensor-based parallel algorithm. Physica A: Statistical Mechanics and its Applications, 611, 128472.

[9] De Luca, P., Galletti, A., Ghehsareh, H. R., Marcellino, L., & Raei, M. (2020). A GPU-CUDA framework for solving a two-dimensional inverse anomalous diffusion problem. In Parallel Computing: Technology Trends (pp. 311-320). IOS Press.

[10] P. De Luca, A. Galletti, G. Giunta, and L. Marcellino, "Accelerated Gaussian convolution in a data assimilation scenario," in International Conference on Computational Science, pp. 199-211, 2020.

[11] V. Bevilacqua, A. Di Marino, E. Di Nardo, A. Ciaramella, I. De Falco, and G. Sannino, "Cross-domain Super-Resolution in Medical Imaging," 2024.

[12] Anderson, A.R.A., Chaplain, M.A.J., Rejniak, K.A., "Single-Cell-Based Models in Biology and Medicine", Springer, Basel, 2007.

[13] Glazier, J.A., Graner, F., "Simulation of the differential adhesion driven rearrangement of biological cells", Physical Review E, 47(3):2128-2154, 1993.

[14] De Luca, P., Galletti, A., Marcellino, L. (2025). Towards a Parallel Code for Cellular Behavior in Vitro Prediction. In: Sergeyev, Y.D., Kvasov, D.E., Astorino, A. (eds) Numerical Computations: Theory and Algorithms. NUMTA 2023. Lecture Notes in Computer Science, vol 14478. Springer, Cham. https://doi.org/10.1007/978-3-031-81247-7_6

[15] Nickolls, J., Buck, I., Garland, M., Skadron, K., "Scalable Parallel Programming with CUDA", Queue, 6(2):40-53, 2008.

[16] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., & Sorensen, D. (1999). LAPACK Users' Guide (3rd ed.). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898719604

[17] Davis, T. A. (2004). Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. ACM Transactions on Mathematical Software, 30(2), 196-199. https://doi.org/10.1145/992200.992206

[18] NVIDIA Corporation, "CUDA C++ Programming Guide", Version 12.3, 2023.

[19] Bell, N., Garland, M., "Implementing sparse matrix-vector multiplication on throughput-oriented processors", In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, ACM, 2009.

[20] Anderson, A.R.A., Weaver, A.M., Cummings, P.T., Quaranta, V., "Tumor morphology and phenotypic evolution driven by selective pressure from the microenvironment", Cell, 127(5):905-915, 2013.

[21] NVIDIA Corporation, "CUDA Direct Sparse Solver (cuDSS) Library Guide", Version 1.0.0, 2023.

[22] NVIDIA Corporation, "CUB Library Documentation", Version 2.1.0, 2023.

[23] NVIDIA Corporation, "cuRAND Library Programming Guide", Version 10.3, 2023.

[24] CINECA Supercomputing Centre, SuperComputing Applications and Innovation Department. (2024). "LEONARDO: A Pan-European Pre-Exascale Supercomputer for HPC and AI applications.", Journal of large-scale research facilities, 8, A186. https://doi.org/10.17815/jlsrf-8-186

[25] Kirk, D.B., Hwu, W.W., "Programming Massively Parallel Processors: A Hands-on Approach", Morgan Kaufmann, 2016.

[26] Harris, M., Luebke, D., "GPU Architecture and CUDA Programming Overview", In GPU Computing Gems Jade Edition, Morgan Kaufmann, 2018.

[27] Sanders, J., Kandrot, E., "CUDA by Example: An Introduction to General-Purpose GPU Programming", Addison-Wesley Professional, 2010.

[28] Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C., "GPU Computing", Proceedings of the IEEE, 96(5):879-899, 2008.

[29] Cuomo, S., Farina, R., Galletti, A., & Marcellino, L. (2014, September). An error estimate of Gaussian Recursive Filter in 3Dvar problem. In 2014 Federated Conference on Computer Science and Information Systems (pp. 587-595). IEEE.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 197–206

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Investigation of novel epigenetic signals through combinatorial crosstalk between histone isoforms

Hatice Döşeme[a,b], Seyit Kale[a,c,*]

[a]*Izmir Biomedicine and Genome Center, Dokuz Eylül University Health Campus, Balçova, Izmir 35330, Türkiye*

[b]*Izmir International Biomedicine and Genome Institute, Dokuz Eylül University Health Campus, Balçova, Izmir 35330, Türkiye*

[c]*Faculty of Medicine, Department of Biophysics, Izmir Katip Çelebi University, Çiğli, Izmir 35620, Türkiye*

Abstract

Our genomic material is packed into a fiber known as chromatin. The smallest repeating architectural unit of chromatin is a disc-shaped nucleoprotein called the nucleosome. Nucleosomes are dynamic entities whose conformation, structural stability, and DNA binding properties can be altered through epigenetic mechanisms, i.e., chemical alterations that do not involve changes in DNA sequence. These alterations involve post-translational modifications and full replacement of structural components of the protein components of the nucleosome, also known as histones. While the crosstalk between histone post-translational modifications has been extensively studied as key epigenetic regulators, the potential crosstalk between histone variants, in other words isoforms, remains largely unexplored. Some of these variants exhibit significant sequence differences from their canonical counterparts, and their potential crosstalk could be implicated in novel functional purposes. We employ atomistic molecular simulations to examine how different combinations of these variants could come together and alter the structural and dynamic properties of chromatin. This *in silico* approach offers a cost-effective and rapid alternative to traditional wet-lab experiments, which is particularly crucial in the field of chromatin and epigenetics due to the challenging nature of stably reconstituting nucleosomes. We report unique variant combinations that induce large-scale alterations in the structure, dynamics, and the DNA affinity of nucleosomes, suggestive of potential roles in the context of DNA-templated processes in health and in disease.

Keywords: nucleosome; molecular dynamics; computational biophysics

* Corresponding author. ORCID: https://orcid.org/0000-0001-7903-8543
E-mail address: seyit.kale@ibg.edu.tr

10.1016/j.procs.2025.08.246

1. Introduction

Our genomic material is packed into a fiber known as chromatin whose smallest repeating architectural unit is a disc-shaped nucleoprotein known as the nucleosome [1]. Nucleosomes are dynamic entities whose conformation, stability, and DNA-binding properties can be altered through multiple mechanisms, such as incorporating different histone variants and post-translational modifications (PTMs). A nucleosome is composed of ~147bp of DNA wrapped around a core octamer of protein components known as histones (Fig. 1A). These histones are known as H3, H4, H2A, and H2B [1].

While the intricate interplay between the nucleosome and its interactors orchestrates all DNA-templated processes, a key question remains: how do specific histone isoforms contribute to this complex choreography? Our current understanding on nucleosomes based on three-decades of structural studies points to the fact that while the average architecture of nucleosomes is largely the same, their functional properties could be very diverse. This diversity could be imprinted in dynamics which could be fundamental to various complex functions such as transcription, replication, DNA-repair, and cell fate alteration [2].

Histone diversity is most pronounced in the two evolutionarily youngest histones, H2A and H2B [3] where certain pairs show preferential pairing, resulting in a wide range of alternative combinations and raising the diversity of potential nucleosome configurations (Fig. 1B). Furthermore, the H2A:H2B dimer becomes a unique functional unit that can be recognized and replaced by chaperones and chromatin remodelers [3]. DNA can be compacted and arranged in the nucleus with the help of the H2A:H2B dimers which anchor DNA around the nucleosome core [26]. The dimer unit can assemble independently, and it can remain stable even while the nucleosome unwinds along the DNA termini. This complex interaction highlights how flexible and adaptive chromatin architecture is, offering a dynamic foundation for controlling gene expression and cellular identity in a variety of biological situations [4]. Furthermore, the H2A:H2B dimer is stable during nucleosome unwinding, can assemble independently, and can be identified and replaced by chaperones or remodelers as a single unit. The main H2A variants are H2A.X, H2A.Z, macroH2A, H2A.B, H2A.J, TSH2A.1 and H2A.L. H2B variants are TSH2B, H2B.W1, subH2B and H2B.E [27].

Recent work using cutting-edge techniques like atomistic simulations and detailed structural analyses is revealing a more dynamic and potentially asymmetric crosstalk between DNA and histone proteins [5]. We have observed H2A variants acting as intrinsic DNA unwinders, and this behavior is further amplified when combined with CENP-A, the centromere-specific isoform of histone H3. CENP-A presence confers the nucleosome a more flexible and open conformation which is critical for the mitotic fidelity [28] and the assembly of the inner kinetochore [30]. Thus, the simpler interaction of various kinetochore protein components, especially the CCAN (Constitutive Centromere-Associated Network), facilitates the proper establishment of the inner kinetochore [28]. Among these isoforms, H2A.Bbd and H2A.B are particularly intriguing because the DNA unwinding can also be coupled with opening of nucleosome gyres away from each other. These dynamic behaviors can be understandably difficult to capture via structural studies alone, highlighting the importance of high-performance computational approaches in fields such as chromatin and epigenetics where wet-lab experiments can be challenging and expensive.

1.1 Widom 601 and Widom 601L DNA Sequences

The 147-base pair long DNA sequence known as Widom 601 was developed through plasmid selection with the goal of finding the sequence with the highest affinity for the histone core octamer [6]. This property ensures that nucleosomes can be stably reconstituted in various settings, making it a frequently used sequence in the structural studies of chromatin. The Widom 601 sequence is non-palindromic and one half of this sequence binds the nucleosome more strongly than the other half as inferred from single molecule pulling experiments [31,32]. With respect to the sense strand of the sequence, the “strong” half precedes the “weak” half of the Widom 601 sequence. More recently, a palindromic version of this sequence, known as Widom 601L, has been developed where both halves carry the same sequence of the original Widom 601 sequence’s “strong” half. This new sequence has been shown to confer structural and dynamic implications in the way linker histone H1 binds the nucleosome [28].

1.2. Histone-DNA interactions

There are several hundred water-mediated interactions and more than 120 direct protein-DNA interactions in the nucleosome. Direct protein-DNA interactions are found at specific locations on the octamer surface, particularly near regions facing the DNA minor grooves. Histone interfaces with DNA are formed by salt linkages, hydrogen bonds between sidechain basically charged groups and hydroxyls, and main-chain amides with the DNA backbone phosphates

[7]. Different combinations of these local interactions contribute to sequence and pattern specific recognition of DNA by histone core octamers. Despite their propensity to favor certain DNA sequences over others, nucleosomes can bind to almost any sequence, in part thanks to the flexibility of water-mediated connections. The DNA inside the nucleosome core is distorted by the strength and placement of DNA-binding sites around the octamer surface [8].

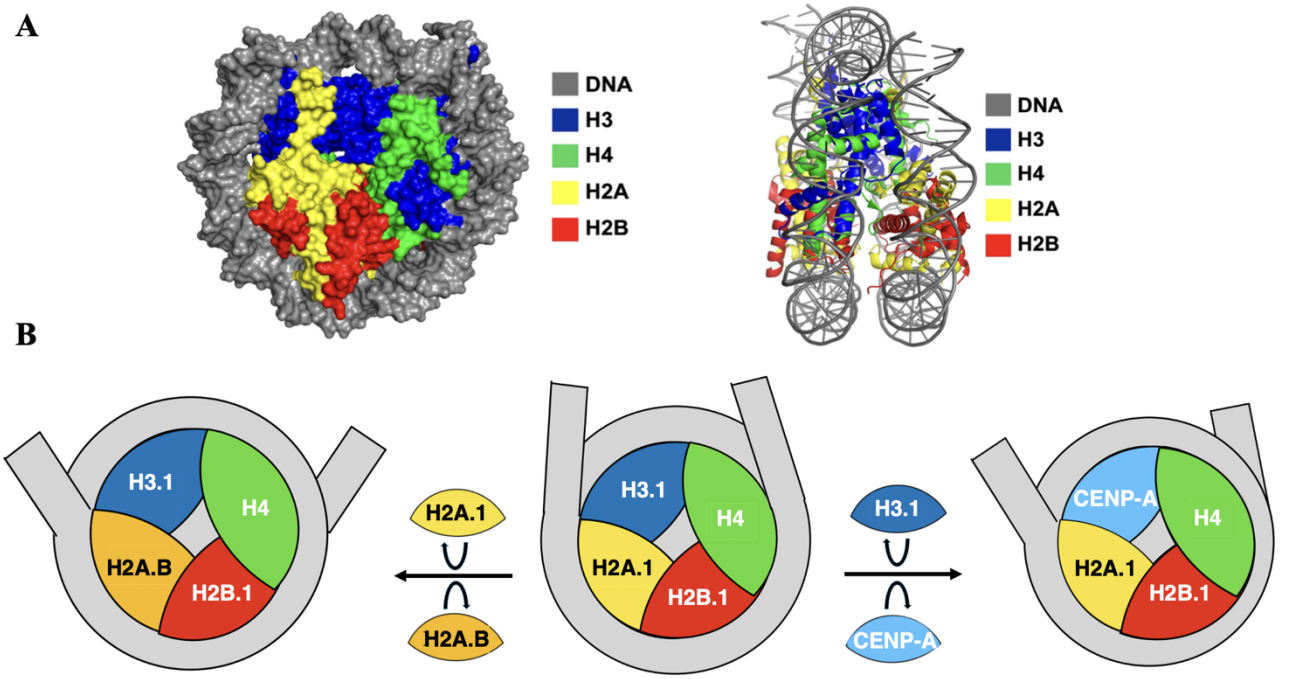


Fig. 1. A. Front (left) and side (right) views of the crystal structure of the nucleosome core particle consisting of H2A, H2B, H3, and H4 core histones, and DNA. **B.** Schematic representation of behavior alteration by the exchange of histone isoform pairs. Note the schematic representation of the conformational changes in DNA termini.

1.3. Histone variants

The macromolecular construction known as eukaryotic chromatin is extremely dynamic. Chromatin remodeling complexes that introduce PTMs to the core histones can alter the location, composition, and structure of nucleosome core particles. Histone variants can also be incorporated to alter the higher order three-dimensional form of the chromatin fiber [10][11]. Sequence alignment of between CENP-A and the canonical H2A suggests approximately 50% sequence conservation. In comparison, H2A isoforms H2A.B and H2A.Bbd differ from their canonical counterparts in 3% and 9%, respectively.

1.3.1. Isoforms of H2A

The highest variety among core histone isoforms is found in the H2A histone family. These variations are particularly pronounced in their distinct C-terminal sequences and in the way they are positioned on chromosomes. Examples of these include the globally dispersed H2A.Z and H2A.X, as well as the vertebrate-specific H2A.B, and H2A.Bbd, and macroH2A. H2A.Z contributes to gene activation by preventing the spread of yeast gene silencing. The truncated C-terminal tail of H2A.Bbd, which is linked to active genes and lacking from inactive X chromosomes, causes nucleosomes to become unstable. The precise process of silencing macroH2A, which is concentrated on inactive X chromosomes, is unknown, however, it has a sizable non-histone domain that may have enzymatic activity. At DNA breaks, phosphorylation of H2A.X, identified by a particular C-terminal pattern, facilitates the recruitment of DNA repair proteins [10].

- **H2A.Bbd**
  Evidence points to the promotion of elevated gene activity by H2A.Bbd-containing nucleosomes. Compared to canonical nucleosomes, these particles are less stable and wrap lesser DNA at their termini, which may increase DNA accessibility. Supporting this behavior, H2A.Bbd nucleosome arrays exhibit increased transcription rates in *in vitro* studies. Finally, H2A.Bbd is shown to have quicker nucleosome exchange in living cells, an indicator of active transcription [14].

- **H2A.B**
  The mammalian-specific histone variant H2A.B is particularly prevalent in testes and is implicated in the production of sperm and embryos as well as development of various tumors. This variant's presence at replication forks of actively transcribed genes, and DNA repair sites suggests that it could be involved in DNA damage response, RNA processing, gene regulation, and cell division. Therefore, in contrast to normal H2A nucleosomes, H2A.B-containing nucleosomes probably use novel regulatory mechanisms [13].

1.3.2. Isoforms of H3

Histone H3, a crucial part of eukaryotic chromatin, aids in the synthesis of nucleosomes through its long N-terminal tail and globular domain. Among all histone proteins, H3 notably experiences the greatest post-translational modifications. In humans and other eukaryotes, the CENP-A protein, encoded by the CENP-A gene, plays a crucial role in defining kinetochore location on chromosomes by acting as a variant of histone H3 [15].

- **CENP-A**
  CENP-A is essential for centromere identification because it identifies the precise spot on each chromosome where the kinetochore assembles. Where sister chromatids join during mitosis is ultimately determined by this core particle [28]. CENP-A containing nucleosomes constitute a constant part of centromeres and is frequently linked to changes in normal histone proteins. CENPA is essentially a modified form of histone H3 that takes the place of normal H3. With extensive sequence diversity and no common histone modification sites on its N-terminal tail, it differs greatly from conventional H3 [16].

2. Materials and Methods

We retrieved the atomic coordinates of five nucleosome structures from the Protein Data Bank (PDB) and AlphaFold [9]. These structures are: 1- Canonical Nucleosome (PDB ID: 3LZ0 [17]), 2- H2A.Bbd:CENP-A-containing nucleosome (AlphaFold and PDB ID: 6TEM [16]), 3-H2A.B:CENP-A-containing nucleosome (PDB IDs: 6V2K [12], 6TEM [16]), 4-H2A.B:CENP-A-containing nucleosome and 5-Widom601L-H2A.B:CENP-A nucleosome (PDB IDs: 8AAG [29], 6M4H [13], and 6TEM [16]) (**Fig. 2**). Each nucleosome structure was simulated as three independent replicates. In each complex, we elongated the nucleosomal DNA ends such that in the final constructs there are symmetrically positioned 149 base pairs of DNA. To ensure that every complex has the same original Widom 601 DNA sequence, we modified mutated single base pairs using the Web 3DNA web server. We used PyMOL and SuperLooper2 to correct missing flexible protein loops. Each structure was solvated in a cubic water box in the presence of 161.5 mM NaCl. The physiological salt concentrations that are replicated in wet-lab investigations using PBS solution are reflected in the selection of the NaCl content. Subroutines of the GROMACS software, version 2024.2 [18], were used to introduce the ions. We described the biological material and the solvent environment using CHARMM36m [19][20] force field and the OPC water model [21], respectively. Experimental agreement in our previous research utilizing the nucleosome core particle supports the choice of this force field and water model combination. We collected production trajectories for each complex using GROMACS version 2024.2 [18]. Each system was first energy minimized by combining the conjugate gradient and steepest descent methods. Following energy minimization, the systems were equilibrated for 1 ns at 100 K and 310 K in the NVT ensemble, followed by the NPT ensemble at 310K and 1 atm production trajectories phase for all four nucleosomal systems using an integration timestep of 2 fs and to a total of 500 ns. Three combinations of histone variants were chosen for statistical validation, and they were rerun in two copies, each 500 ns long, using different initial random number seeds. The Parrinello-Rahman barostat [22][24] and the velocity-rescaling thermostat [23] are used to maintain constant pressure and temperature, respectively. To predict protein-protein binding affinities, we used a heuristic regression model known as Prodigy which uses a contact-based model that examines surface characteristics and intermolecular interactions [25]. While maintaining the original physical model of Prodigy, we modified the computation pipeline such that affinity calculation is computed for all recorded frames of a trajectory. This allowed us to improve the statistical reliability of the prediction while preserving the underlying model. To supplement computed affinities, we also counted the number of proximal non-hydrogen atom pairs to assess relative binding strengths. A distance threshold of 5 Å between non-hydrogen atoms is considered a proximal contact. PyMOL, VMD, and Python scripts are used to perform all analyses. To assess gaping motion within the nucleosomes inter-gyre separation analysis is performed by calculating distance between the mass centers of opposing nucleosomal local DNA segments. Each DNA segment is 10 or 20 base pairs long. The computation of the posterior, i.e., bottom, gyre opening involved the distance between the mass center of base pairs –45

to −35 with the mass center of base pairs +35 to +45. The computation of the strong side opening involved the distance between the mass center of base pairs −65 to −45 with the mass center of base pairs +15 to +35. The computation of the weak side opening involved the distance between the mass center of base pairs −35 to −15 with the mass center of base pairs +45 to +65. Base pair indices follow the sense strand and span from −74 to +74.

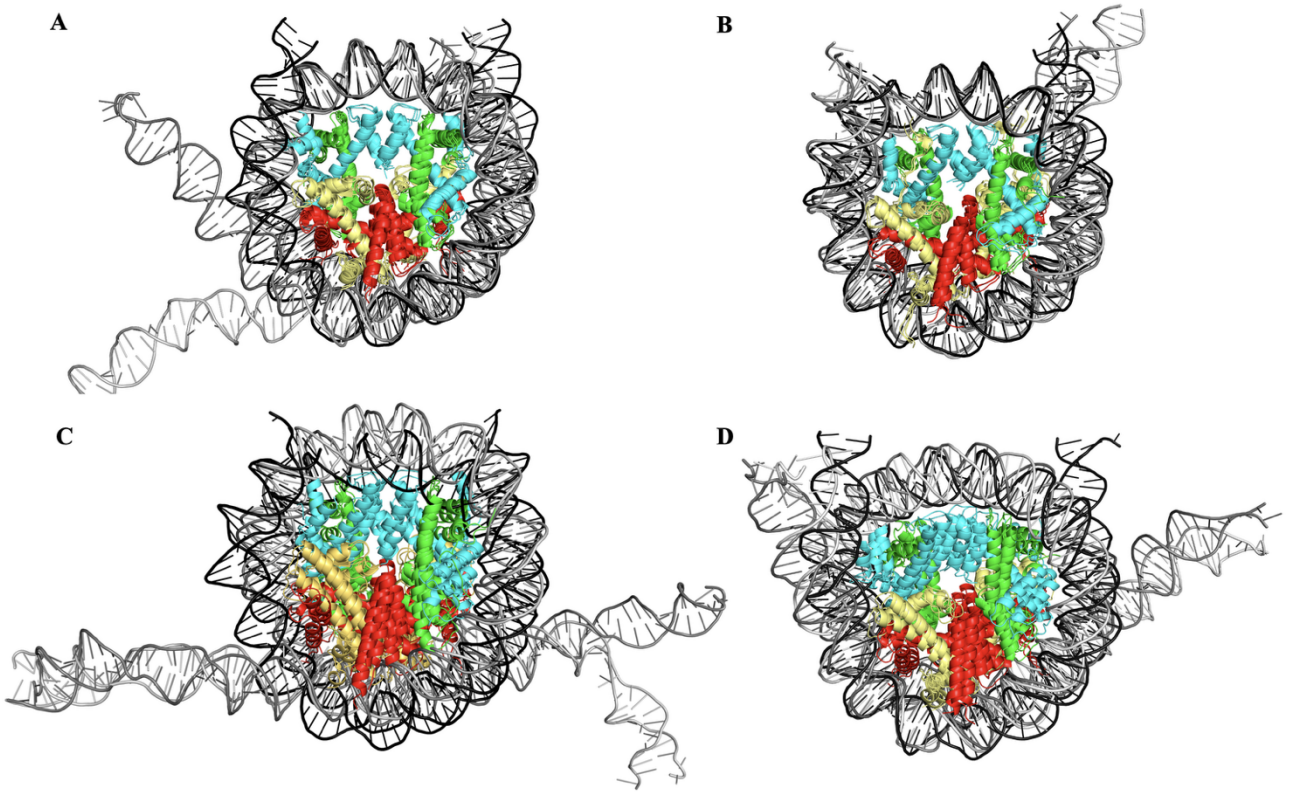


Fig. 2. Representative snapshots of three simulations with initial (time = 0, black DNA), midpoint (time = 250 nanoseconds, dark gray DNA), and final (time = 500 nanoseconds, light gray DNA) states of the H2A.B:CENP-A-containing nucleosome (**A**, starting PDB IDs: 6V2K [12], 6TEM [16]), H2A.B:CENP-A-containing nucleosome (**B**, starting PDB IDs: 6M4H [13], 6TEM [16]), and H2A.Bbd:CENP-A-containing nucleosome (**C**, AlphaFold and starting PDB ID: 6TEM [16]). These structures have a Widom601 DNA sequence (**A**, **B**, and **C**). **D.** Representative snapshots of the H2A.B:CENP-A containing nucleosome with Widom601L DNA sequence. Starting PDB IDs for this setup are 8AAG, 6M4H, and 6TEM.

3. RESULTS

3.1. H2A.B:CENP-A-containing nucleosome exhibits strong tetramer-dimer stability

As a measure of core particle stability, we investigated the interactions between the core tetramer units (i.e., H3 and H4, or their isoforms) and the core dimer units (i.e., H2A and H2B, or their isoforms). Contact analysis of the nucleosome structures containing canonical histone and histone variant combinations, along with their replicas, revealed consistent structural results. Root Mean Square Deviation (RMSD) analysis for the canonical nucleosome replicates (both histone core and DNA) suggested that the simulations reach equilibrium sufficiently well over the course of the simulation timescales (**Supplementary Fig. 1**).

We found that the strongest interaction was observed between the tetramer-dimer proteins within the H2A.B:CENP-A nucleosome structure (**Fig. 3A**). This observation is supported by distance analysis, where the H2A.B:CENP-A combination displayed the shortest distances, reflecting a strong interaction between the protein components (**Fig. 3B**). Binding affinity calculations showed that the average binding affinity for the canonical nucleosome was approximately –25 kcal/mol. The H2A.B:CENP-A nucleosome exhibited the closest binding affinity to this value, further supporting a robust interaction between the tetramer-dimer histones (**Fig. 3C**). These findings suggest that there is minimal separation between the histone proteins in the H2A.B:CENP-A nucleosome, indicating that the protein-protein interactions within this combination are strong and stable (**Supplementary Fig. 2**).

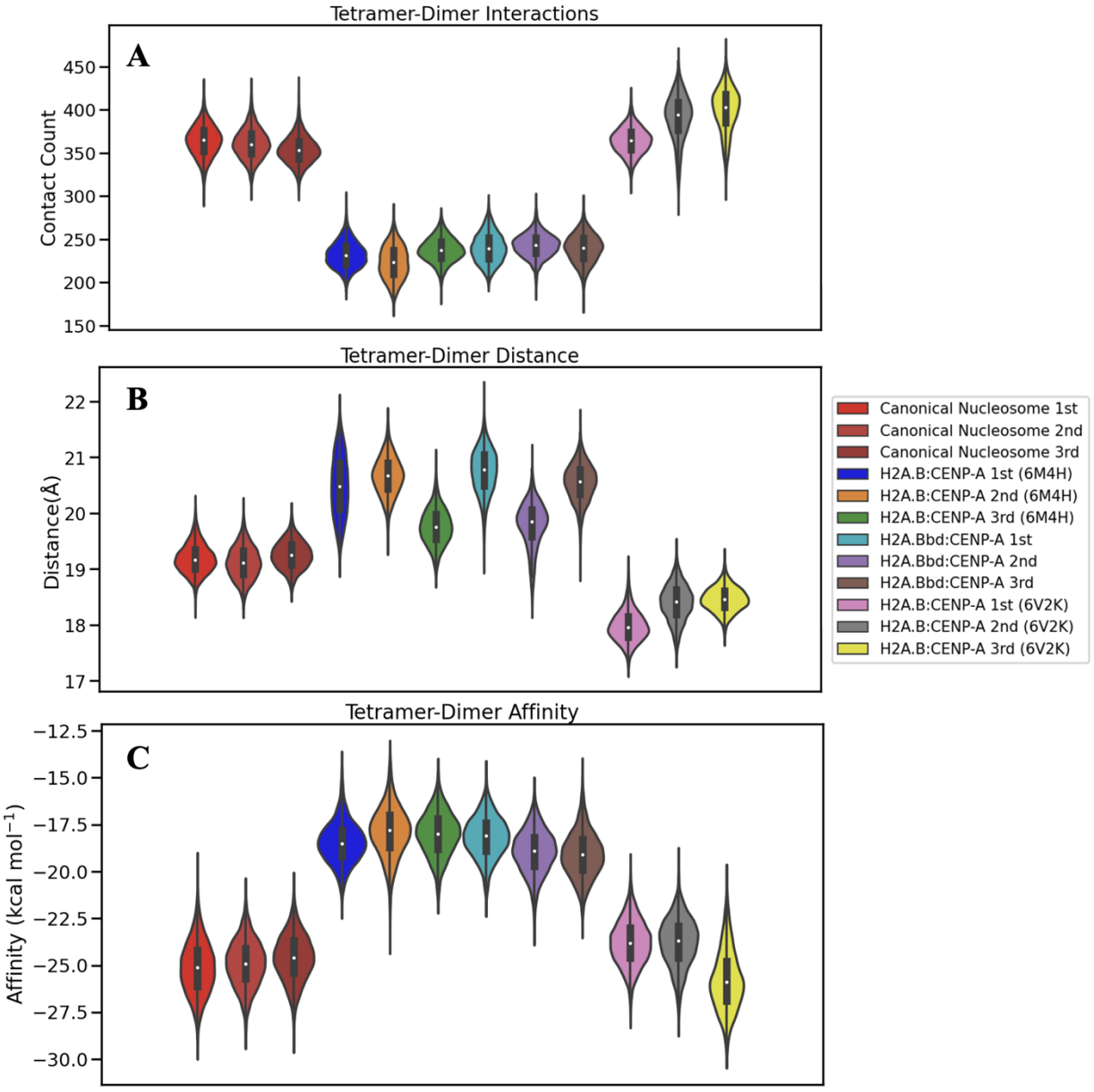


Fig. 3. Average contacts (**A**) and distances (**B**) between the heavy atoms of the tetramer and dimer structures. **C.** Computed tetramer-dimer affinities of H2A:H3 histone variant combinations.

3.2. Chimeric nucleosomes exhibit inter-gyre gaping motion

Chimeric nucleosomes containing H2A.B or H2A.Bbd in conjunction with CENP-A can undergo large-scale conformational changes as inferred from our structural analyses of the molecular dynamics trajectories. While it is well-known that DNA termini can spontaneously open and close within multi-millisecond timescales [33] there is evidence of a slower collective motion within nucleosomes whereby DNA halves, or "gyres", can spontaneously separate from each other and later return back [34]. These motions are known as breathing and gaping, respectively, and they are generally not exhibited by the canonical nucleosome in significant amplitudes [2]. The chimeric nucleosomes considered in this study exhibit both types of motions in noticeable and varying degrees.

Our findings demonstrate that the DNA gyres in these chimeric nucleosomes exhibit noticeable gaping motion, either sideways (on the "weak" side or the "strong" side of the Widom 601 nucleosome) or posterior (from the bottom, opposite the dyad). This behavior, which is virtually absent in the canonical nucleosomes, suggests that the histone variant combinations can introduce a special type of flexibility to the global nucleosome. These conformational changes imply that these isoforms may support less understood chromatin functions compared to the traditional physiological roles of chromatin such as transcriptional activation, silencing, or DNA repair (**Fig. 4**).

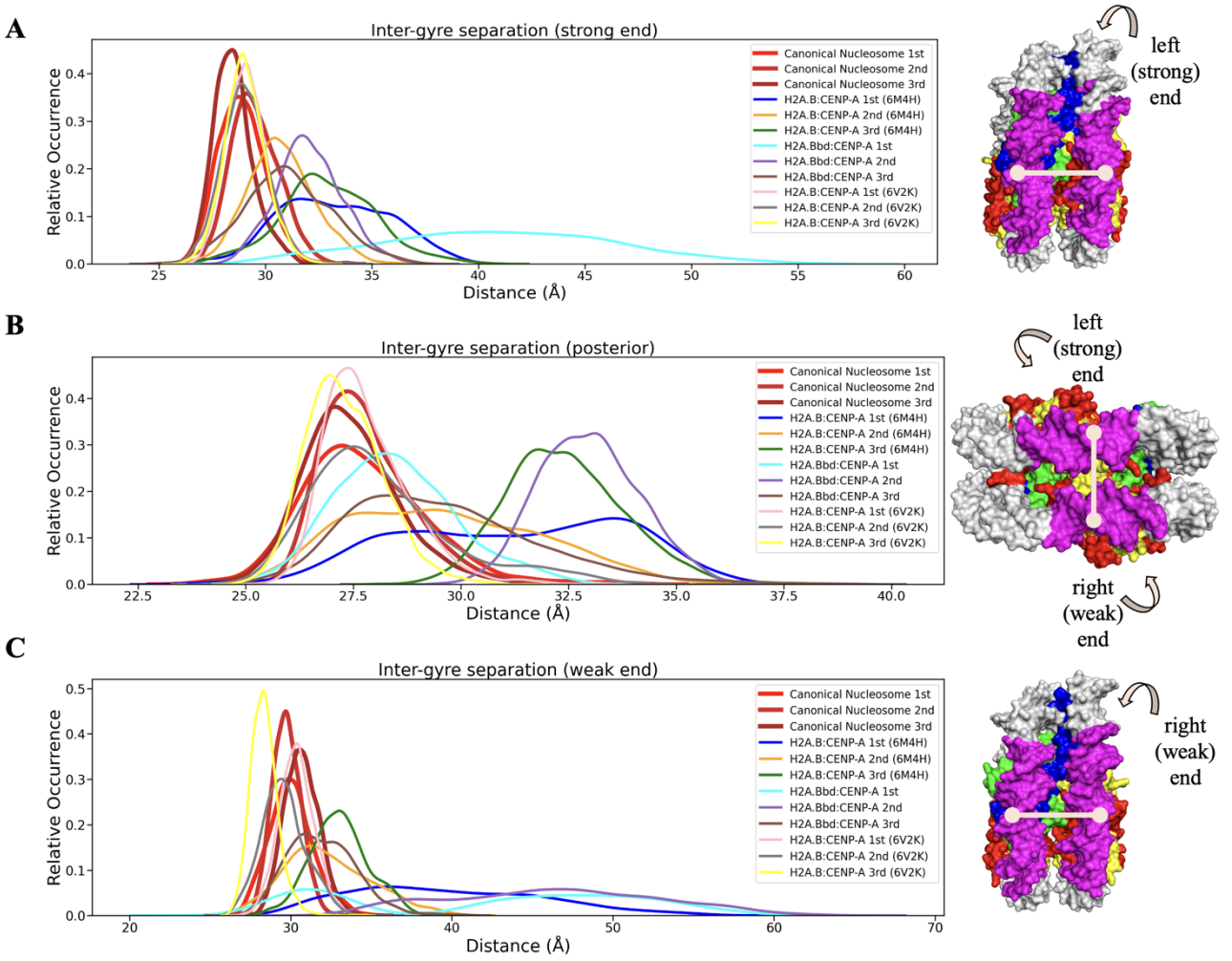


Fig. 4. Nucleosome inter-gyre separations at three sides: left side (**A**), from below (**B**), and right side (**C**) in nucleosome combinations containing various H2A:H3 histone variant combinations.

3.3. Chimeric nucleosomes exhibit symmetrical and asymmetrical opening of DNA termini

The chimeric nucleosomes containing H2A isoforms in combination with the CENP-A nucleosome also exhibit both symmetrical and asymmetrical opening of DNA ends of the Widom sequence. When examining the canonical nucleosome structures formed by various H2A:CENP-A combinations, the greatest degree of DNA arm opening was observed in the H2A.Bbd:CENPA complex in the form of large-scale symmetrical opening to the extent of several superhelical turns (**Fig. 5A-B-C**). This suggests that this core octamer has the lowest DNA affinity of the chimeric combinations we studied here. In contrast, H2A.B presence leads to a similar degree of DNA end opening albeit asymmetrically, i.e., on one end only (**Fig. 5D-E-F**). This degree of DNA termini opening is not seen in the control canonical nucleosome simulation. Chimeric nucleosomes containing CENP-A and H2A.B were simulated using the palindromic Widom 601L DNA sequence. Our findings were further supported by the observation of both symmetrical and asymmetrical unwrapping of the DNA arms in these nucleosomes, which is consistent with our previous result using the canonical Widom 601 DNA sequence (**Supplementary Fig. 3**).

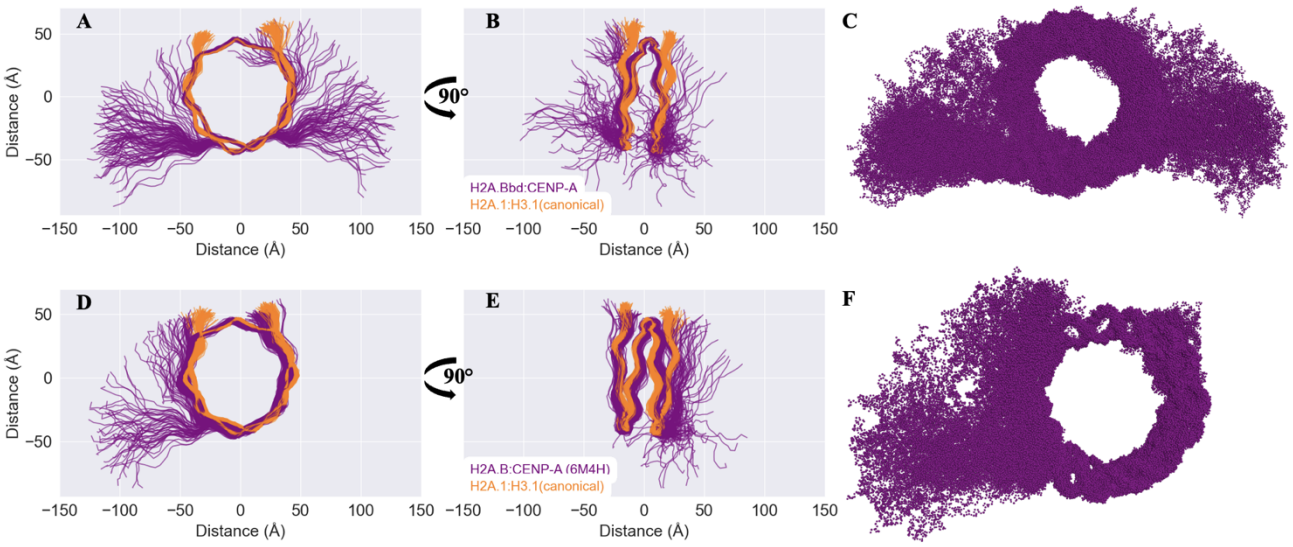


Fig. 5. Traces of unfolding DNA in different nucleosomes. **A-B.** H2A.Bbd:CENP-A-containing (purple) versus the canonical nucleosome (orange). **C.** H2A.Bbd:CENP-A-containing (purple) nucleosome with phosphate groups represented as spheres. **D-E.** H2A.B:CENP-A-containing versus the canonical nucleosome. **F.** H2A.B:CENP-A-containing nucleosome with phosphate groups represented as spheres. Traces are computed over base-pair mass centers.

4. Discussion, Future Remarks and Conclusion

In this study, we used atomistic molecular dynamics simulations to study the potential effects of combining select isoforms of H2A and the centromere-specific H3 isoform, CENP-A, on nucleosome structure and DNA accessibility. Some of these histone isoform combinations improve chromatin accessibility or core octamer stability in pronounced ways which could be implicated in fundamental physiological processes such as transcription, DNA repair, and chromatin remodeling. These findings imply that these specific combinations of histone variants might be an understudied topic of chromatin regulation. It is yet to be elucidated whether such histone isoform combinations could arise spontaneously in nature. Finding allosteric variant combinations and examining their effects on chromatin dynamics and cellular functions should be the focus of future research. Furthermore, research should look at how these combinations respond to different physiological conditions and whether they serve as chromatin-associated protein regulatory switches. Gaining knowledge of the entire range of these variant combinations and their functional roles could deepen our understanding of epigenetic mechanisms and offer fresh perspectives on the dynamic and structural regulation of the genome.

Acknowledgments

This work was supported by the EMBO Installation Grant no 5056 (awarded to S.K., available by partial funds provided by TÜBİTAK). MD trajectories are generated and analyzed using the high-performance computing resources provided by the Izmir Biomedicine and Genome Center, TÜBİTAK ULAKBIM High Performance and Grid Computing Center (TRUBA resources), and the Spanish National Supercomputer MareNostrum5. Access to MareNostrum5 was provided by the EuroHPC-AI and Data-Intensive Applications Access Call (EHPC-AI-2024A01-088, awarded to S.K.). The numerical analyses reported in this paper were partially performed at Izmir Biomedicine and Genome Center. Financial support by TÜBİTAK does not mean that the content of the publication has been approved in a scientific sense by TÜBİTAK.

Data Availability

Supplementary Figures and the data utilized in this study, including molecular dynamics input, topology and parameter files can be accessed on Zenodo (under https://doi.org/10.5281/zenodo.15809079, doi: 10.5281/zenodo.15809079).

Supplementary Figure 1: RMSDs of histone core proteins **(A),** and nucleosomal DNA **(B)** over the 500 ns long trajectories.

Supplementary Figure 2: Average contacts (**A**) and distances (**B**) between the heavy atoms of the tetramer and dimer structures. (**C)** Computed tetramer-dimer affinities of histone variant combinations.

Supplementary Figure 3: Traces of unfolding DNA in different nucleosomes. **A-B**. H2A.B:CENP-A-containing (purple) with Widom601L DNA versus the canonical nucleosome (orange). **C**. H2A.B:CENP-A-containing (purple) nucleosome with phosphate groups represented as spheres. **D-E**. H2A.B:CENP-A-containing versus the canonical nucleosome. **F**. H2A.B:CENP-A-containing nucleosome with phosphate groups represented as spheres. Traces are computed over base-pair mass centers.

References

[1] Luger, K., Mä Der, A. W., Richmond, R. K., Sargent, D. F., & Richmond, T. J. (1997). Crystal structure of the nucleosome core particle at 2.8 A ˚ resolution. In Nature © Macmillan Publishers Ltd (Vol. 389).
[2] Shaytan, A. K., Armeev, G. A., Goncearenco, A., Zhurkin, V. B., Landsman, D., & Panchenko, A. R. (2016). Coupling between Histone Conformations and DNA Geometry in Nucleosomes on a Microsecond Timescale: Atomistic Insights into Nucleosome Functions. Journal of Molecular Biology, 428(1), 221–237. https://doi.org/10.1016/j.jmb.2015.12.004
[3] Shaytan, A. K., Landsman, D., & Panchenko, A. R. (2015). Nucleosome adaptability conferred by sequence and structural variations in histone H2A-H2B dimers. In Current Opinion in Structural Biology (Vol. 32, pp. 48–57). Elsevier Ltd. https://doi.org/10.1016/j.sbi.2015.02.004
[4] Luger, K., Dechassa, M. L., & Tremethick, D. J. (2012). New insights into nucleosome and chromatin structure: An ordered state or a disordered affair? In Nature Reviews Molecular Cell Biology (Vol. 13, Issue 7, pp. 436–447). https://doi.org/10.1038/nrm3382
[5] Boopathi, R., Danev, R., Khoshouei, M., Kale, S., Nahata, S., Ramos, L., Angelov, D., Dimitrov, S., Hamiche, A., Petosa, C., & Bednar, J. (2020). Phase-plate cryo-EM structure of the Widom 601 CENP-A nucleosome core particle reveals differential flexibility of the DNA ends. Nucleic Acids Research, 48(10), 5735–5748. https://doi.org/10.1093/NAR/GKAA246
[6] Lowary, P. T., & Widom, J. (n.d.). New DNA Sequence Rules for High Affinity Binding to Histone Octamer and Sequence-directed Nucleosome Positioning.
[7] Segal, E., Fondufe-Mittendorf, Y., Chen, L., Thåström, A., Field, Y., Moore, I. K., Wang, J. P. Z., & Widom, J. (2006). A genomic code for nucleosome positioning. Nature, 442(7104), 772–778. https://doi.org/10.1038/nature04979
[8] Davey, C. A., Sargent, D. F., Luger, K., Maeder, A. W., & Richmond, T. J. (2002). Solvent mediated interactions in the structure of the nucleosome core particle at 1.9 Å resolution. *Journal of Molecular Biology*, *319*(5), 1097–1113. https://doi.org/10.1016/S0022-2836(02)00386-8
[9] Jumper, J., Evans, R., Pritzel, A. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* 596, 583–589 (2021). https://doi.org/10.1038/s41586-021-03819-2
[10] Mariño-Ramírez, L., Kann, M. G., Shoemaker, B. A., & Landsman, D. (2007). Histone structure and nucleosome stability NIH Public Access.
[11] Zhou, B. R., Feng, H., Kale, S., Fox, T., Khant, H., de Val, N., Ghirlando, R., Panchenko, A. R., & Bai, Y. (2021). Distinct Structures and Dynamics of Chromatosomes with Different Human Linker Histone Isoforms. *Molecular cell*, *81*(1), 166–182.e6. https://doi.org/10.1016/j.molcel.2020.10.038
[12] Zhou, M., Dai, L., Li, C., Shi, L., Huang, Y., Guo, Z., Wu, F., Zhu, P., & Zhou, Z. (2021). Structural basis of nucleosome dynamics modulation by histone variants H2A.B and H2A.Z.2.2. The EMBO Journal, 40(1). https://doi.org/10.15252/embj.2020105907
[13] Hirano, R., Arimura, Y., Kujirai, T., Shibata, M., Okuda, A., Morishima, K., Inoue, R., Sugiyama, M., & Kurumizaka, H. (2021). Histone variant H2A.B-H2B dimers are spontaneously exchanged with canonical H2A-H2B in the nucleosome. Communications Biology, 4(1). https://doi.org/10.1038/s42003-021-01707-z

[14] Bao, Y., Konesky, K., Park, Y. J., Rosu, S., Dyer, P. N., Rangasamy, D., Tremethick, D. J., Laybourn, P. J., & Luger, K. (2004). Nucleosomes containing the histone variant H2A.Bbd organize only 118 base pairs of DNA. EMBO Journal, 23(16), 3314–3324. https://doi.org/10.1038/sj.emboj.7600316
[15] Rosenfeld, J. A., Wang, Z., Schones, D. E., Zhao, K., DeSalle, R., & Zhang, M. Q. (2009). Determination of enriched histone modifications in non-genic portions of the human genome. BMC Genomics, 10. https://doi.org/10.1186/1471-2164-10-143
[16] Boopathi, R., Danev, R., Khoshouei, M., Kale, S., Nahata, S., Ramos, L., Angelov, D., Dimitrov, S., Hamiche, A., Petosa, C., & Bednar, J. (2021). Phase-plate cryo-EM structure of the Widom 601 CENP-A nucleosome core particle reveals differential flexibility of the DNA ends. Nucleic Acids Research, 48(10), 5735–5748. https://doi.org/10.1093/NAR/GKAA246
[17] Vasudevan, D., Chua, E. Y. D., & Davey, C. A. (2010). Crystal Structures of Nucleosome Core Particles Containing the "601" Strong Positioning Sequence. *Journal of Molecular Biology*, *403*(1), 1–10. https://doi.org/10.1016/j.jmb.2010.08.039
[18] Van der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A.E., Berendsen, H.J.C., (2005). GROMACS: fast, flexible, and free. J. Comput. Chem., 26, 1701–1718.
[19] Best, R.B., Zhu, X., Shim, J., Lopes, P.E.M., Mittal, J., Feig, M., et al., (2012). Optimization of the additive CHARMM allatom protein force field targeting improved sampling of the backbone phi, psi and side-chain chi(1) and chi(2) dihedral angles. J. Chem Theory Comput., 8, 3257 -3273.
[20] Huang, J., Rauscher, S., Nawrocki, G., Ran, T., Feig, M., de Groot, B.L., et al., (2017). CHARMM36m: an improved force field for folded and intrinsically disordered proteins. Nature Methods, 14, 71–73.
[21] Izadi, S., Anandakrishnan, R., Onufriev, A.V., (2014). Building water models: a different approach. J. Phys. Chem. Letters, 5, 3863–3871.
[22] Doğan, D., Arslan, M., Uluçay, T., Kalyoncu, S., Dimitrov, S., and Kale, S. (2021). CENP-A Nucleosome is a Sensitive Allosteric Scaffold for DNA and Chromatin Factors. J Mol Biol 433. 10.1016/j.jmb.2020.166789.
[23] Bussi G, Donadio D, Parrinello M. Canonical sampling through velocity rescaling. *J Chem Phys*. 2007;126(1):014101. doi:10.1063/1.2408420
[24] Parrinello, M., Rahman, A., (1981). Polymorphic transitions in single-crystals - a new molecular-dynamics method. J.Appl. Phys., 52, 7182–7190.
[25] Xue LC, Rodrigues JP, Kastritis PL, Bonvin AM, Vangone A. PRODIGY: a web server for predicting the binding affinity of protein-protein complexes. *Bioinformatics*. 2016;32(23):3676-3678. doi:10.1093/bioinformatics/btw5
[26] Hirano, R., Arimura, Y., Kujirai, T., Shibata, M., Okuda, A., Morishima, K., Inoue, R., Sugiyama, M., & Kurumizaka, H. (2021). Histone variant H2A.B-H2B dimers are spontaneously exchanged with canonical H2A-H2B in the nucleosome. *Communications biology*, *4*(1), 191. https://doi.org/10.1038/s42003-021-01707-z
[27] Shaytan, A. K., Landsman, D., & Panchenko, A. R. (2015). Nucleosome adaptability conferred by sequence and structural variations in histone H2A-H2B dimers. *Current opinion in structural biology*, *32*, 48–57. https://doi.org/10.1016/j.sbi.2015.02.004
[28] Kale, S., Boopathi, R., Belotti, E., Lone, I. N., Graies, M., Schröder, M., Petrova, M., Papin, C., Bednar, J., Ugrinova, I., Hamiche, A., & Dimitrov, S. (2023). The CENP-A nucleosome: where and when it happens during the inner kinetochore's assembly. *Trends in biochemical sciences*, *48*(10), 849–859.https://doi.org/10.1016/j.tibs.2023.07.010
[29] Louro, J. A., Boopathi, R., Beinsteiner, B., Mohideen Patel, A. K., Cheng, T. C., Angelov, D., Hamiche, A., Bendar, J., Kale, S., Klaholz, B. P., & Dimitrov,S. (2023). Nucleosome dyad determines the H1 C-terminus collapse on distinct DNA arms. *Structure (London, England : 1993)*, *31*(2), 201–212.e5. https://doi.org/10.1016/j.str.2022.12.005
[30] Roulland, Y., Ouararhni, K., Naidenov, M., Ramos, L., Shuaib, M., Syed, S. H., Lone, I. N., Boopathi, R., Fontaine, E., Papai, G., Tachiwana, H.,Gautier,T., Skoufias, D., Padmanabhan, K., Bednar, J., Kurumizaka, H., Schultz, P., Angelov, D., Hamiche, A., & Dimitrov, S. (2016). The Flexible Ends of CENP-A Nucleosome Are Required for Mitotic Fidelity. *Molecular cell*, *63*(4), 674–685. https://doi.org/10.1016/j.molcel.2016.06.023
[31] Ngo, T. T., Zhang, Q., Zhou, R., Yodh, J. G., & Ha, T. (2015). Asymmetric unwrapping of nucleosomes under tension directed by DNA local flexibility. *Cell*, *160*(6), 1135–1144. https://doi.org/10.1016/j.cell.2015.02.001
[32] Gansen, A., Felekyan, S., Kühnemuth, R., Lehmann, K., Tóth, K., Seidel, C. A. M., & Langowski, J. (2018). High precision FRET studies reveal reversible transitions in nucleosomes between microseconds and minutes. *Nature communications*, *9*(1), 4628. https://doi.org/10.1038/s41467-018-06758-1
[33] Li, G., Levitus, M., Bustamante, C., & Widom, J. (2005). Rapid spontaneous accessibility of nucleosomal DNA. *Nature structural & molecular biology*, *12*(1), 46–53. https://doi.org/10.1038/nsmb869
[34] Ngo, T. T., & Ha, T. (2015). Nucleosomes undergo slow spontaneous gaping. *Nucleic acids research*, *43*(8), 3964–3971. https://doi.org/10.1093/nar/gkv276

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 207–217

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Transforming Computational Power into Environmental Solutions: HPC-driven Research on Urethanase-mediated Plastic Degradation

Pedro Paiva[a,b], Luís M. C. Teixeira[a,b], Pedro Ferreira[a,], Daniel E. Otzen[b,c], Pedro A. Fernandes[a,b], Maria J. Ramos[a,b,*]

[a]*LAQV/REQUIMTE, Departamento de Química e Bioquímica, Faculdade de Ciências Universidade do Porto, Rua do Campo Alegre, s/n, 4169-007 Porto, Portugal*

[b]*EnZync Center for Enzymatic Deconstruction of Thermoset Plastics*

[c]*Interdisciplinary Nanoscience Centre (iNANO), Department of Molecular Biology and Genetics, Aarhus University, Gustav Wieds Vej 14, DK-8000, Aarhus, Denmark*

Abstract

The extensive accumulation of persistent synthetic polymers has made plastics a rapidly expanding global concern. In this context, enzymatic degradation offers a viable and environmentally friendly alternative to the conventional plastic recycling methods. Here, we discuss how multi-microsecond Molecular Dynamics (MD) simulations and hybrid Quantum Mechanics/Molecular Mechanics (QM/MM) methods have allowed us to shed light on the conformational dynamics and catalytic mechanisms underlying the activity of the polyurethane-degrading enzymes UMG-SP2 and UMG-SP3. Our findings emphasize the role of these computational approaches for the identification of the rate-limiting step, stationary states' geometries, and residue-specific electrostatic effects, enabling subsequent mutagenesis studies to enhance the efficiency of these urethanases. The access to EuroHPC's computing resources was essential to this research, as it allowed the extensive sampling of enzyme:substrate dynamics over large simulation timescales and the QM/MM-level resolution of the free-energy profile associated with the cleavage of the substrate's urethane bond. This work demonstrates the relevant role of HPC-driven research in advancing mechanistic insights and highlights how the synergy between large-scale computational resources and enzymatic knowledge can enhance our understanding of plastic biodegradation.

* Corresponding author. Tel.: +351 220 402 000
E-mail address: mjramos@fc.up.pt

1877-0509
10.1016/j.procs.2025.08.247

Keywords: Plastic degradation; Polyurethane; Enzyme catalysis; Urethanase; Molecular Dynamics; High-Performance Computing

1. Introduction

1.1. Plastics

Plastics are important polymeric materials used in a huge range of applications, including boat hulls, decks, tanks, pipes, gratings, and bottles. However, their massively widespread use has resulted in significant global plastic pollution. Moreover, plastic-derived microplastics are building up in our bodies, and we have no knowledge on the long-term effects of this accumulation.[1-4]

Current plastic degradation methods pose significant environmental problems. Landfilling requires extensive space and high costs for land acquisition and maintenance worldwide, while also posing long-term risks of soil and groundwater contamination due to persistent organic pollutants released from plastic breakdown. Incineration generates harmful by-products such as polychlorinated biphenyls, mercury, furans, and dioxins, releasing highly toxic and carcinogenic fumes, which can cause severe threats to both the human health and the environment.[5-8]

Unsaturated polyesters add up to a significant portion of present plastic waste, and since plastics are generally resistant to total degradation under natural conditions, the finding of plastic-degrading microorganisms offers a promising solution. While enzymes capable of the depolymerisation of some plastics, such as Polyethylene Terephthalate (PET), have been identified and extensively studied, those that efficiently depolymerize polyurethanes (PU) have not yet been as fully researched or characterized.

Unfortunately, the breakdown of synthetic polymers by various microbes occurs at a relatively slow rate, which makes the biodegradation process impractical for real-time industrial applications. Additionally, most studies have focused on the biodegradability of individual bacterial strains, whereas in nature, bacteria typically work synergistically in consortia, as has been observed in various studies.[9]

One promising approach to enhance plastic biodegradability is by increasing the efficiency of the enzymes responsible for polymer breakdown. Recent advances in computational biology and bioinformatics have played a key role in unravelling the molecular mechanisms involved in microbial plastic degradation.[10, 11]

1.2. Objectives

Certain microorganisms produce enzymes, such as hydrolases, lipases, and proteases among others, capable of hydrolysing the structural backbone of plastic polymers, and enzyme engineering has become a valuable strategy, providing a modern and targeted method to accelerate plastic degradation and address the escalating problem of plastic waste.[12, 13]

Performing successful enzymatic mechanistic studies, which reveal the transition state structures of the rate-limiting steps is highly impactful because understanding how to enhance enzyme efficiency relies on the detailed structural knowledge of those critical rate-limiting steps. We have developed a method capable of predicting mutations that enhance enzyme activity, that will be used when we have the transition-state structures of the rate-limiting steps of the enzymatic mechanisms.[14]

Therefore, our end goal was to perform enzymatic catalysis studies conducive to the elucidation of the then-unknown catalytic mechanisms of two newly discovered urethanases, *i.e.* thermoset unsaturated polyurethane (PU)-degrading enzymes UMG-SP2 and UMG-SP3, the latter of which has since been characterised and published in the PDB and the literature by the consortium En'Zync, of which we are members, funded by the Novo Nordisk Foundation (https://inano.au.dk/about/research-centers-and-projects/enzync).[15]

Over the past three decades, theoretical calculations have gained increasing importance, mainly due to the remarkable advancements in computational technology. The fundamental knowledge accumulated in establishing an enzymatic mechanism holds the potential to contribute uniquely to experiments being conducted to outcompete classical chemical recycling methods.

1.3. Choice of computational methods

Hybrid Quantum Mechanics/Molecular Mechanics (QM/MM) molecular dynamics (MD) with enhanced sampling is presently the most reliable approach for establishing enzymatic mechanisms.[14, 16, 17]

However, achieving accurate results requires a high-level theoretical QM method. Density functional theory (DFT) is preferred due to its favourable balance between accuracy and computational cost. Higher-level methods would be ideal, but they are impractical due to their excessive computational demands. PBE is one of the affordable density functionals, in this context, used with a double-zeta polarized basis set; theoretical levels lower than this do not yield robust results.[18]

The rest of the enzyme is described at the molecular mechanics level. An effective way to determine a reaction mechanism is by using umbrella sampling to compute free energy profiles for the elementary steps of different mechanistic hypotheses along the relevant reaction coordinates. We have recently shown that this approach is highly effective for various enzyme reactions, provided that at least 10 ps per umbrella sampling window is performed.[17, 19] However, achieving this necessary level of calculations, requires approximately 0.5 million core hours per free energy profile. Given that a complete mechanistic study typically involves two to three reaction steps with two possible mechanistic pathways each, the total computational cost quickly escalates to several million core hours.

To begin with, the initial enzyme structure with adequate protonation states, solvated in water, with the substrate correctly positioned, must be relaxed using classical MD. These simulations were performed using the open-source software GROMACS[20, 21] due to its excellent computational efficiency and parallelization performance. Each classical MD simulation lasted 400 ns with multiple replicas to ensure a thorough sampling of the conformational space. Additionally, we used the CP2K[22] software, equally known for its high computational efficiency and strong parallelization capabilities, in our QM/MM MD calculations.

1.4. Advances achieved through HPC access

The access to HPC resources is important to drive computational work. Nowadays, computational fields have been established as relevant due to their capability and versatility. Moreover, the technological and scientific development of computational methodologies promoted their reliability. Therefore, there is an increased demand for employing these methods in several fields. The study of enzymes is no stranger to this fact. Indeed, the scientific community recognized the importance of using computational methods to study these complex biological macromolecules. This is particularly true for studying their dynamical behaviour, as simulations allow for a more in-depth atomistic analysis. Nevertheless, this type of analysis requires statistical significance. To achieve this, there is a need for considerable sampling, allowing the system to explore conformational space. Consequently, computational resources are typically the bottleneck of these works. Thus, access to HPC resources can significantly aid in overcoming this hurdle, allowing for greater exploration and, consequently, a deeper understanding of the given system.

On the other hand, the knowledge of enzymatic mechanisms with atomic detail is one of the most relevant targets in biochemistry. However, in the literature, reaction mechanisms are often described incorrectly by hypothesis or by similarity to other established reaction mechanisms. Hybrid QM/MM simulations, using DFT at the QM level as well as good functionals and basis sets, can achieve excellent levels of sampling when running the corresponding calculations in an HPC, resulting in very accurate and detailed reaction mechanisms.

Herein, we demonstrate the importance of HPC resources and their impact on enzymatic studies. For that, we showcased how computational methodologies aided in understanding UMG-SP2 and UMG-SP3 dynamic behaviour and catalytic mechanism.

2. Understanding the Dynamical Behaviours of UMG-SP2 and UMG-SP3

The usefulness of HPC resources was highlighted throughout the studies that we conducted on UMG-SP2 and UMG-SP3. Studying structural features and stability requires simulating the respective systems for a significant amount of time. By leveraging the petascale capabilities of the EuroHPC supercomputer MeluXina, we successfully

carried out MD simulations for UMG-SP2 and UMG-SP3, collectively exceeding 56 μs of simulation time. These calculations utilized over 1.2 million CPU core-hours on MeluXina, whose nodes are equipped with dual EPYC 7H12 AMD CPUs, each featuring 64 cores running at 2.6 GHz for a total of 128 cores per node, and 512 GB of RAM. Multithreading was disabled for all GROMACS runs, as our benchmarks indicated that using one thread per physical core provided the most efficient configuration for our system. Our simulations confirmed dynamical behaviours observed experimentally. In addition, they allowed us to sample the conformational space of the enzyme:substrate complexes, revealing important interactions, and validating the stability of the modelled systems.

Concerning UMG-SP2, several structures have been resolved by X-ray crystallography.[23] This allowed us to study the stability of structural features exhibited by each state. In all of them, we identified a loop (named L3 loop), which was present in all structures and played an important role as a gating mechanism for substrate recognition. The loop encompassed Lys224, which changed its orientation upon substrate binding. In the absence of a ligand, Lys224 adopted an inward arrangement, with its side chain facing the active site. On the other hand, when a ligand was bound to the enzyme, Lys224 opted for an outward orientation, with its side chain facing the solvent. Interestingly, the same observations were made regarding UMG-SP3. Once again, multiple structures have been resolved by X-ray crystallography.[15] The striking difference between these structures was identified as the orientation of Arg209 (equivalent to UMG-SP2's Lys224), which changed based on the absence or presence of a ligand. In line with UMG-SP2, the absence of a ligand rendered an inward arrangement (Figure 1A), while its presence promoted an outward arrangement (Figure 1B). Our simulations confirmed the stability of these structural features and revealed interactions that stabilized the orientation of Lys224 (UMG-SP2) and Arg209 (UMG-SP3). In the inward orientation, the ligand established several hydrogen bond interactions with the backbone of the enzyme and nearby residues. In addition, this orientation resulted in the residue being inserted into a pocket that contained bulky residues (such as Trp and His), reducing its freedom of movement. The outward orientation was stabilized by several electrostatic and π-cation interactions. Based on this, it was proposed that L3 was responsible for substrate recognition, with the transition to an outward arrangement flagging the binding of a substrate. To confirm this, we transferred the inhibitor, present in the crystallographic UMG-SP3-inhibitor complex, to the ligand-free structure. Accordingly, we observed that Arg209 reoriented to the outward arrangement, confirming that the switch in orientation was triggered by the presence of a ligand (Figure 1C).

To better understand the role of L3 loop and its impact on catalysis, both enzymes were subjected to a mutagenesis study, combining both experimental and computational methodologies. Regarding UMG-SP2, four residues of L3 were targeted (Ala141, Lys224, Asp226, and Gln399) to study its influence on the hydrolytic activity of UMG-SP2. Our MD simulations revealed that the tested mutations should favour catalysis through different contributions: i) increase of the L3 loop flexibility due to size reduction and polarity change; ii) rearrangement of L3, promoting a stable interaction with L4, providing direct access to the active site; and iii) increase of the L3 flexibility due to charge removal. Ultimately, we expected that a more mobile L3 would improve catalysis. The same conclusions were withdrawn regarding the UMG-SP3 study. Once again, four L3 residues were targeted (Arg204, Arg209, Gly210, and Glu211). Our simulations revealed that most mutations promoted a more flexible L3. This was either due to a loss of stabilizing interactions because of charge removal or size decrease, or due to repulsion caused by charge replacement. Based on the substrate recognition mechanism, we proposed that a more flexible L3 should render an easily activated UMG-SP3, enhancing catalysis. Indeed, experimental activity assays revealed that most of these tested mutants exhibited higher activity when compared to the *wild-type*. Through the simulation of each mutated system, we were able to predict the productivity of a given mutation and provide an explanation. Therefore, in line with the UMG-SP2 study, we proved the utility of employing computational methods and combining them with experimental assays.

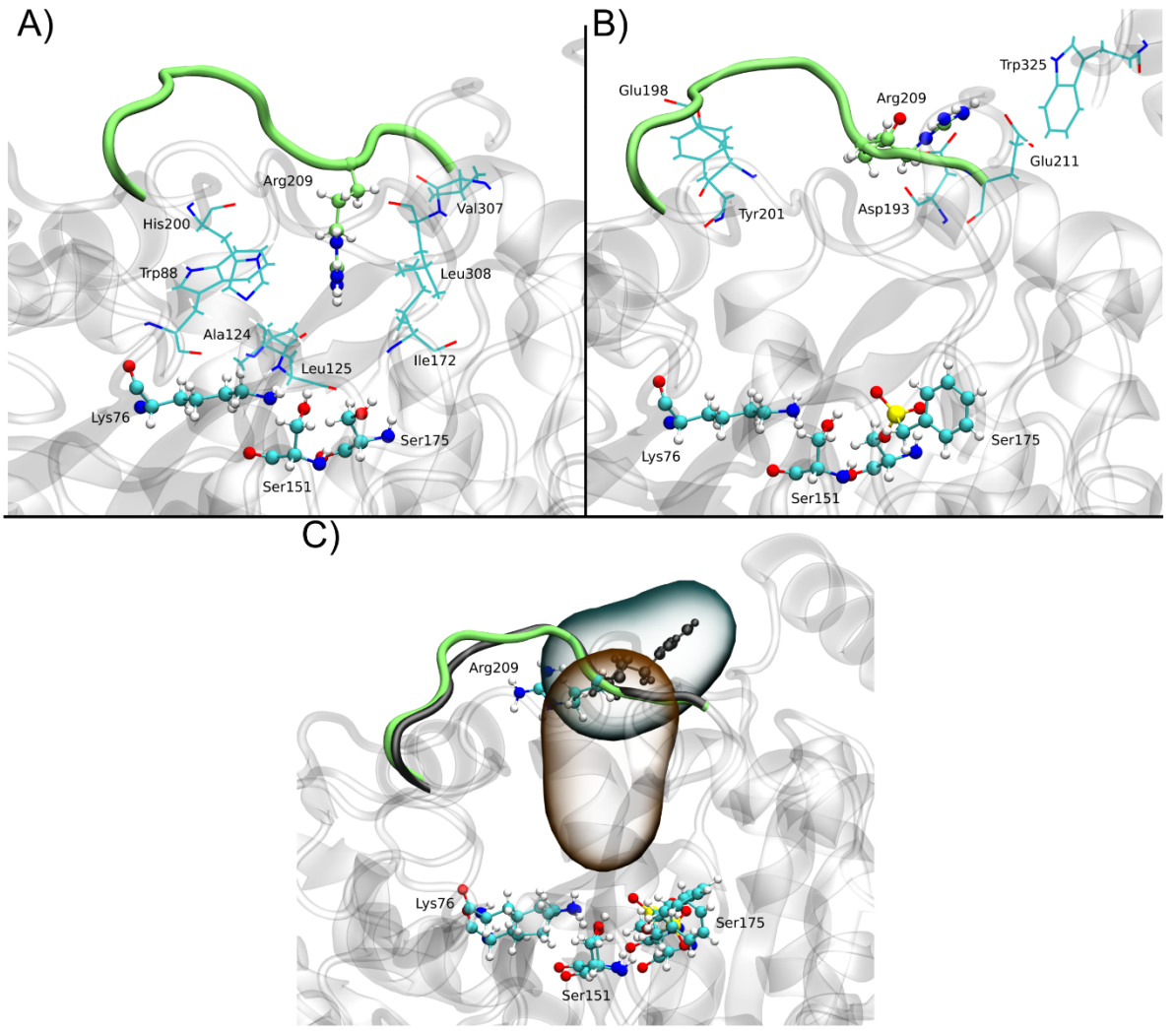


Fig. 1. UMG-SP3 active site structural arrangement of the ligand-free (A), inhibitor-bound (B), and altered (C) structure. In all three structures, the catalytic residues (Lys76, Ser151 and Ser175), Arg209 and L3 loop (lime) are highlighted. Additionally, the interacting residues that stabilize the inward (A) and outward (B) arrangement of Arg209 are also shown. For the altered structure (C), the positioning of the inward (orange sphere) and of the outward (blue sphere) arrangement are highlighted. Furthermore, the conformation of Arg209 and L3 at the inhibitor-bound structure (black) is shown.

Besides studying the dynamical behaviour of structural features, computational simulations are also relevant for validating computationally modelled complexes. This is particularly relevant when we want to study the catalytic mechanism of a given enzyme and there is no resolved enzyme:substrate structure. Accordingly, to study the mechanism of UMG-SP2, we had to model the substrate (DUE-MDA), within the active site, using Molecular Docking.[24] With the use of well-established distance restrictions, the search algorithm was able to correctly position the reaction's key atoms in a favourable orientation. Next, we simulated the modelled UMG-SP2:DUE-MDA complex system to allow the enzyme and the substrate to adapt to their surrounding environment, and to validate the stability of the model. The simulation confirmed that the enzyme's overall fold remained stable. As expected, we observed that DUE-MDA's termini exhibited considerable flexibility, although the substrate remained in the active site throughout

the simulation. Moreover, the reaction's key atoms maintained a catalytically favourable orientation, validating our model.

3. Mechanistic insights into UMG-SP2 urethanase

Our previous MD simulations were instrumental in characterizing the dynamic behaviours of the UMG-SP2 and UMG-SP3 enzymes. These simulations also provided valuable insights into the enzyme-PU substrate interactions and offered a deeper understanding of the enhanced depolymerization activity observed in mutant UMG-SP2 and UMG-SP3 structures. Leveraging this knowledge, we constructed a QM/MM model of the UMG-SP2:DUE-MDA complex (Figure 2) to investigate the catalytic mechanism of UMG-SP2 involved in urethane/carbamate bond degradation.[25] While UMG-SP2 shares the same catalytic triad as other enzymes in the Amidase Signature family,[26-28] and despite previous research on the Ser-Ser$_{cis}$-Lys mechanism,[29] the atomistic and energetic details of this process vary across family members. To our knowledge, this was the first study to disclose the catalytic mechanism of urethane bond cleavage by amidase signature enzymes, as no prior work had addressed this reaction pathway in atomic detail.

We selected four complementary properties to study the catalysis: 1.the free-energy profiles of each step obtained from QM/MM calculations, which allow to identify the rate-limiting step of the urethane bond cleavage catalysed by UMG-SP2; 2.the optimised transition-state structures, which possess a single imaginary frequency associated with the bond-breaking and bond-forming events in each step; 3.the stabilised, by Ile187/Gly188, classical oxyanion-hole; 4. the electrostatic energy contribution of individual residues to the energy barrier upon systematic in-silico deletion (149 mutations). Together, these properties capture the thermodynamics, structure and electrostatics of bond-breaking and bond-forming events, enabling a complete depiction of both acylation and deacylation stages.

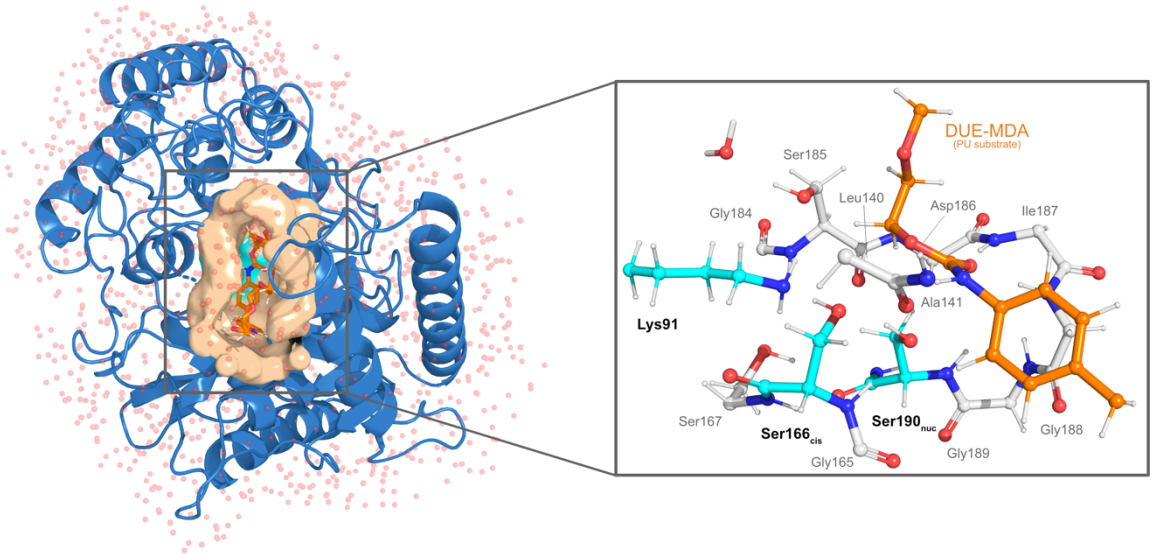


Fig. 2. (Left) The QM/MM model used to study the catalytic machinery of UMG-SP2 (9929 atoms, including solvent). The active site cavity is shown as a bright orange surface (the catalytic triad residues are colored in light blue). The DUE-MDA substrate is shown as orange sticks. Water molecules are represented in red transparent spheres (hydrogen atoms are not shown). (Right) QM region, composed of 126 atoms, shown in ball-and-stick representation. The DUE-MDA substrate is colored in orange. The Ser190$_{nuc}$-Ser166$_{cis}$-Lys91 catalytic triad's carbon atoms are colored in light blue.

Our results revealed that the enzymatic cleavage of the urethane bond by UMG-SP2 proceeded through a two-stage mechanism typical of serine hydrolases: Stage 1 – acylation, and Stage 2 – deacylation.

In the acylation stage, the Ser190$_{nuc}$–Ser166$_{cis}$–Lys91 catalytic triad adopted a geometry that facilitated an efficient proton relay, activating Ser190$_{nuc}$ for nucleophilic attack. The activated serine then concertedly attacked the carbonyl carbon of the substrate's target urethane bond, forming a tetrahedral intermediate that collapsed via cleavage of the ester side of the urethane bond. This reaction released an alcohol-leaving group and yielded an acyl-enzyme complex. We identified this bond cleavage as the rate-limiting step of the reaction, with an associated activation Gibbs energy of 20.8 kcal·mol^{-1} (Figure 3).

In Stage 2, a water molecule (Wat$_{cat}$) that occupied the vacant space left by the alcohol-leaving group hydrolysed the acyl-enzyme intermediate, releasing a highly unstable carbamic acid product, which is known to spontaneously decompose into an amine and CO_2.[30, 31] This final step regenerated the enzyme's catalytic triad and completed the reaction cycle. Unlike in Stage 1, we did not observe a stable tetrahedral intermediate during deacylation. Instead, this geometry appeared only transiently, rapidly decaying into the final product. The hydrolysis step itself involved a concerted reaction combining multiple asynchronous elementary steps, including the activation of Wat$_{cat}$ by Ser166$_{cis}$ and the nucleophilic attack by the activated Wat$_{cat}$ on the carbonyl carbon of the acylated Ser190$_{nuc}$.

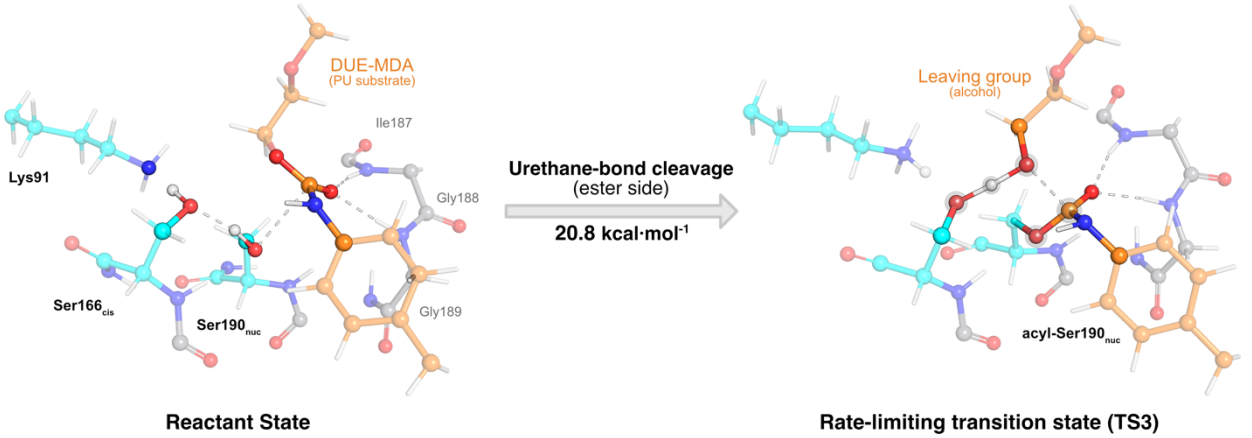


Fig. 3. Representation of the reactant state (left) and the transition state (TS3, right) of the rate-limiting step of the urethane bond cleavage reaction. The DUE-MDA substrate is colored in orange, and the Ser190$_{nuc}$-Ser166$_{cis}$-Lys91 catalytic triad's carbon atoms are colored in light blue.

We computed the full Gibbs free-energy profile of each step at the B3LYP/6-311+G(2d,2p)-D3(BJ):ff14SB//B3LYP/6-31G(d)-D3(BJ):ff14SB level of theory, identifying all key stationary states. The breakdown of the tetrahedral intermediate formed during the acylation stage (*i.e.*, the cleavage of the urethane bond by its ester moiety) emerged as the highest energy barrier in the cycle, suggesting it as the rate-determining step (Figure 3). While it appears to govern the turnover rate, we acknowledge that real-world factors such as product release and solvent diffusion could also influence the overall reaction energetics. However, the calculated energy barriers fell within the experimental range reported for other hydrolases,[32] corroborating the robustness of our model and supporting the enzyme's potential for industrial use under mild conditions.

Throughout the reaction, the oxyanion hole, formed by the backbone amides of Ile187 and Gly188, played an important stabilizing role, particularly in stationary states exhibiting a tetrahedral geometry. We also found that the catalytic Ser166$_{cis}$ not only participated in proton shuttling processes, directly contributing to the activation of the nucleophilic Ser190$_{nuc}$, but also appeared to selectively stabilize the ester moiety of the urethane bond, pointing to a possible role in substrate recognition. These interactions were conserved across catalytically competent conformations observed in MD simulations, which underscores their functional relevance in enabling UMG-SP2 to form efficient enzyme-substrate complexes.

Building on this mechanistic understanding, we next sought to identify which residues most strongly influenced the rate-limiting transition state (TS3, Figure 3). To do so, we systematically removed 149 individual residues from the MM layer and evaluated their impact on the activation barrier through single-point energy calculations. It is noteworthy that this approach did not account for possible conformational rearrangements following residue removal and, as such, operated under the assumption that these modifications do not induce significant structural rearrangements that could compromise the enzyme's catalytic efficiency. Despite this limitation, this method proved highly informative in finding residues with significant electrostatic influence over the rate-limiting TS3.

Our results identified several detrimental residues, *i.e.*, those that raised the activation energy when present. These were primarily positively charged residues located near the nucleophilic $Ser190_{nuc}$, which negatively impacted the necessary redistribution of electron density during catalysis. In contrast, nearby negatively charged residues tended to stabilize TS3, lowering the activation energy. Despite being far from the active site, some destabilizing residues exerted a strong electrostatic influence, emphasizing the important role of long-range interactions in enzyme catalysis. By demonstrating that eliminating or replacing destabilizing residues close to the active site could improve catalytic efficiency—more precisely, by reducing the energetic cost of the rate-limiting step—these results made a strong basis for further rational mutagenesis research. On the other hand, the identified stabilizing residues could serve as anchor points to introduce mutations aimed at enhancing active site polarization.

We developed a mechanistic basis that extends beyond this single enzyme to other members of the amidase signature superfamily by unveiling the complete catalytic cycle of UMG-SP2 at the atomic level. Our work provided detailed insights into the transition-state energetics, substrate stabilization, and active site electrostatics, which are essential for guiding future enzyme engineering efforts. Furthermore, the gathered results provided the groundwork for ongoing QM/MM MD studies into the catalytic mechanism of UMG-SP3, a close homolog of UMG-SP2. These simulations, which require accurate quantum-level assessments of the active site region in a dynamic enzyme environment, are computationally demanding. They have so far consumed over 3.5 million CPU core-hours on the petascale EuroHPC supercomputer MeluXina, evidencing the significant resources required to portray the phenomena behind enzyme catalysis at this resolution. These follow-up studies will help determine whether the structural differences observed in the MD simulations of UMG-SP3 translate into distinct mechanistic or energetic features.

In summary, our research elucidated the catalytic mechanism of UMG-SP2 and provided a roadmap for improving its performance through structure-based enzyme engineering. These insights advance the current understanding of enzymatic urethane bond cleavage and offer concrete strategies for enhancing urethanase activity in biotechnological applications.

4. Conclusion

Here we present our investigation of the conformational dynamics and catalytic capabilities of the recently discovered urethanases UMG-SP2 and UMG-SP3. Using multi-microsecond MD simulations across multiple replicas in combination with hybrid QM/MM calculations, we extensively sampled the enzyme–substrate conformational landscape.

We built the QM/MM models from high-resolution crystal structures of active urethanases (UMG-SP2, PDB code: 8WDW; UMG-SP3, PDB code: 9FW1) in which the substrate analogue DUE-MDA, a substrate analog that constituted a fraction analog of a typical polyether-polyurethane polymer, was docked into the active site. The QM region comprised the catalytic triad, the oxyanion hole stabilizing residues, and the urethane moiety of the substrate up to four covalent bonds from the scissile urethane bond. Multiple replicas of classical MD confirmed that the enzyme–substrate complex remained stable and catalytically competent, with a persistent nucleophile–carbonyl distance around 3.2 ± 0.2 Å. [25] These criteria ensured that all structural and electronic variables relevant to urethane bond cleavage were explicitly represented in the model.

This allowed us to characterize, for the first time, the catalytic mechanism of urethane-bond hydrolysis by an amidase-signature enzyme, identify the key transition states that govern turnover, and map residue-specific contributions to catalytic efficiency.

Our findings demonstrate that hydrolysis of the urethane bond by UMG-SP2 proceeds through the canonical two-step serine-hydrolase cycle, with cleavage of the ester side of the urethane bond constituting the rate-limiting barrier. Electrostatic analysis of the corresponding transition state revealed a small cluster of destabilizing cationic residues near the nucleophilic Ser190, suggesting straightforward mutational routes to lower the energy barrier. These insights now boost ongoing QM/MM-MD studies on UMG-SP3 that once again exploit the computational power of EuroHPC's petascale supercomputer MeluXina. Our extensive MD data also uncovered a concerted gating motion centered on the L3 loop. Arg209 (UMG-SP3) and Lys224 (UMG-SP2) switch between inward and outward orientations in response to ligand binding, thereby steering the substrate towards the catalytic triad. *In silico* guided mutagenesis of residues of this loop produced variants with up to a three-fold increase in catalytic efficiency, emphasizing the practical value of deep conformational sampling.

The quality of our results relied on the extensive conformational sampling performed: a total of 56 μs of classical MD yielded converged active-site fluctuations. We also assessed the sensitivity of the QM/MM values to the choice of the electronic structure method. Specifically, the QM energies of the stationary points in the UMG-SP2 mechanism were calculated with four methods (B3LYP, PWPB95, DSD_PBEB95 and SCS-MP2). The rate-limiting activation free energy varied by no more than 2.3 kcal mol^{-1} (ranging from 20.8–23.1 kcal mol^{-1}), which falls well within the expected intrinsic error of hybrid QM/MM approaches. This variation reflects a systematic error inherent to the choice of functional, as QM calculations yield deterministic values without statistical uncertainties. The computed activation barrier lies within the typical range for hydrolases, and the reaction proceeded to produce the experimentally observed products. All intermediates and transition states exhibited a single imaginary frequency, and continuous intrinsic reaction coordinate paths confirmed their identities. Together, these factors underscore the statistical robustness and reliability of our results.

Achieving this level of detail was attainable only through access to EuroHPC resources. Simulating complex enzyme–substrate systems over microsecond timescales and computing DFT-level free-energy profiles required hundreds of parallel CPU cores and millions of core-hours. Without these resources, we could neither achieve statistically meaningful sampling nor describe the chemistry at a quantum-mechanical level robustly. EuroHPC access proved indispensable not merely for speeding up calculations but for making this research possible at all.

Our results illustrate how the synergy between advanced simulation methodologies and state-of-the-art HPC infrastructure is improving enzyme research.[15, 23, 25, 33] HPC-powered computational enzymology is no longer just a supporting tool but can be a driver of discovery by guiding experiments, elucidating mechanistic pathways, and delivering tangible biotechnological solutions.

The forthcoming EuroHPC exascale platforms will permit explicit sampling of polymer matrices decorated with multiple enzymes, routine use of higher-level electronic-structure methods, and seamless integration with more robust AI and data-driven algorithms. These capabilities will further accelerate the design, optimization, and deployment of enzymatic solutions to global challenges like plastic pollution.

Acknowledgments

The authors would like to thank the EnZync consortium and the financial support received by the Novo Nordisk Fonden, under project NNF22OC0072891 (Challenge Programme 2022 - Recycling or a Sustainable Society). PP, LMCT, PF, PAF, and MJR also acknowledge the financial support of FCT/MCTES (LA/P/0008/2020 DOI 10.54499/LA/P/0008/2020, UIDP/50006/2020 DOI 10.54499/UIDP/50006/2020 and UIDB/50006/2020 DOI 10.54499/UIDB/50006/2020), through national funds. PP, LMCT, PF, PAF, and MJR would further like to thank the European High-Performance Computing Joint Undertaking (EuroHPC JU) that granted access to MeluXina, the petascale EuroHPC supercomputer located in Bissen, Luxembourg, under proposal EHPC-REG2023R03-163. PF thanks FCT for funding (Ref. CEECINST/00136/2021/CP2820/CT0002 DOI 10.54499/CEECINST/00136/2021/CP2820/CT0002. LMCT would also like to thank FCT/MECI for funding his PhD project through the grant 2024.05090.BD.

References

[1] Geyer R, Jambeck JR, Law KL. 2017. Production, use, and fate of all plastics ever made. Sci Adv.3(7):e1700782.
[2] Hartmann NB, Huffer T, Thompson RC, Hassellov M, Verschoor A, Daugaard AE, Rist S, Karlsson T, Brennholt N, Cole M, et al. 2019. Are We Speaking the Same Language? Recommendations for a Definition and Categorization Framework for Plastic Debris. Environ Sci Technol.53(3):1039-1047.
[3] Lamb JB, Willis BL, Fiorenza EA, Couch CS, Howard R, Rader DN, True JD, Kelly LA, Ahmad A, Jompa J, et al. 2018. Plastic waste associated with disease on coral reefs. Science.359(6374):460-462.
[4] Worm B, Lotze HK, Jubinville I, Wilcox C, Jambeck J. 2017. Plastic as a Persistent Marine Pollutant. Annual Review of Environment and Resources.42(Volume 42, 2017):1-26.
[5] Hopewell J, Dvorak R, Kosior E. 2009. Plastics recycling: challenges and opportunities. Philosophical Transactions of the Royal Society of London Series B: Biological Sciences.364(1526):2115-2126.
[6] Kawaguchi H, Ogino C, Kondo A. 2017. Microbial conversion of biomass into bio-based polymers. Bioresource Technology.245(Pt B):1664-1673.
[7] Nagarajan R, Thirumalaisamy S, Lakshumanan E. 2012. Impact of leachate on groundwater pollution due to non-engineered municipal solid waste landfill sites of erode city, Tamil Nadu, India. Iranian J Environ Health Sci Eng.9(1):35.
[8] Roy PK, Titus S, Surekha P, Tulsi E, Deshmukh C, Rajagopal C. 2008. Degradation of abiotically aged LDPE films containing pro-oxidant by bacterial consortium. Polymer Degradation and Stability.93(10):1917-1922.
[9] Huerta Lwanga E, Thapa B, Yang X, Gertsen H, Salanki T, Geissen V, Garbeva P. 2018. Decay of low-density polyethylene by bacteria extracted from earthworm's guts: A potential for soil restoration. Science of the Total Environment.624:753-757.
[10] Mohanan N, Montazer Z, Sharma PK, Levin DB. 2020. Microbial and Enzymatic Degradation of Synthetic Plastics. Frontiers in Microbiology.11:580709.
[11] Skariyachan S, Taskeen N, Kishore AP, Krishna BV. 2022. Recent advances in plastic degradation - From microbial consortia-based methods to data sciences and computational biology driven approaches. Journal of Hazardous materials.426:128086.
[12] Krueger MC, Harms H, Schlosser D. 2015. Prospects for microbiological solutions to environmental pollution with plastics. Appl Microbiol Biotechnol.99(21):8857-8874.
[13] Tokiwa Y, Calabia BP, Ugwu CU, Aiba S. 2009. Biodegradability of plastics. International Journal of Molecular Sciences.10(9):3722-3742.
[14] Pinto AV, Ferreira P, Neves RPP, Fernandes PA, Ramos MJ, Magalhães AL. 2021. Reaction Mechanism of MHETase, a PET Degrading Enzyme. ACS Catalysis.11(16):10416-10428.
[15] Rotilio L, Bayer T, Meinert H, Teixeira LMC, Johansen MB, Sommerfeldt A, Petersen AR, Sandahl A, Keller MB, Holck J, et al. 2025. Structural and Functional Characterization of an Amidase Targeting a Polyurethane for Sustainable Recycling. Angewandte Chemie International Edition.64(7):e202419535.
[16] Jerves C, Neves RPP, Ramos MJ, da Silva S, Fernandes PA. 2021. Reaction Mechanism of the PET Degrading Enzyme PETase Studied with DFT/MM Molecular Dynamics Simulations. ACS Catalysis.11(18):11626-11638.
[17] Neves RPP, Cunha AV, Fernandes PA, Ramos MJ. 2022. Towards the Accurate Thermodynamic Characterization of Enzyme Reaction Mechanisms. Chemphyschem.23(13):e202200159.
[18] Sousa SF, Ribeiro AJM, Neves RPP, Brás NF, Cerqueira NMFSA, Fernandes PA, Ramos MJ. 2017. Application of quantum mechanics/molecular mechanics methods in the study of enzymatic reaction mechanisms. Wiley interdisciplinary reviews: Computational molecular science.7(2):e1281-n/a.
[19] Coimbra JTS, Neves RPP, Cunha AV, Ramos MJ, Fernandes PA. 2022. Different Enzyme Conformations Induce Different Mechanistic Traits in HIV-1 Protease. Chemistry.28(42):e202201066.
[20] Abraham MJ, Murtola T, Schulz R, Páll S, Smith JC, Hess B, Lindahl E. 2015. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX.1-2:19-25.
[21] Van Der Spoel D, Lindahl E, Hess B, Groenhof G, Mark AE, Berendsen HJ. 2005. GROMACS: fast, flexible, and free. Journal of Computational Chemistry.26(16):1701-1718.
[22] Kuhne TD, Iannuzzi M, Del Ben M, Rybkin VV, Seewald P, Stein F, Laino T, Khaliullin RZ, Schutt O, Schiffmann F, et al. 2020. CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations. Journal of Chemical Physics.152(19):194103.
[23] Li Z, Han X, Cong L, Singh P, Paiva P, Branson Y, Li W, Chen Y, Jaradat DsMM, Lennartz F, et al. 2025. Structure-Guided Engineering of a Versatile Urethanase Improves Its Polyurethane Depolymerization Activity. Advanced Science.n/a(n/a):2416019.
[24] Jones G, Willett P, Glen RC, Leach AR, Taylor R. 1997. Development and validation of a genetic algorithm for flexible docking. Journal of Molecular Biology.267(3):727-748.
[25] Paiva P, Teixeira LMC, Wei R, Liu W, Weber G, Morth JP, Westh P, Petersen AR, Johansen MB, Sommerfeldt A, et al. 2025. Unveiling the enzymatic pathway of UMG-SP2 urethanase: insights into polyurethane degradation at the atomic level. Chemical Science (Royal Society of Chemistry: 2010).16(5):2437-2452.
[26] Neu D, Lehmann T, Elleuche S, Pollmann S. 2007. Arabidopsis amidase 1, a member of the amidase signature family. The FEBS Journal.274(13):3440-3451.
[27] Neumann S, Granzin J, Kula MR, Labahn J. 2002. Crystallization and preliminary X-ray data of the recombinant peptide amidase from Stenotrophomonas maltophilia. Acta Crystallographica Section D: Biological Crystallography.58(Pt 2):333-335.
[28] Shin S, Lee TH, Ha NC, Koo HM, Kim SY, Lee HS, Kim YS, Oh BH. 2002. Structure of malonamidase E2 reveals a novel Ser-cisSer-Lys catalytic triad in a new serine hydrolase fold that is prevalent in nature. EMBO Journal.21(11):2509-2516.
[29] Cerqueira NMFSA, Moorthy H, Fernandes PA, Ramos MJ. 2017. The mechanism of the Ser-(cis)Ser-Lys catalytic triad of peptide amidases. Physical Chemistry Chemical Physics.19(19):12343-12354.
[30] Ouellette RJ, Rawn JD. 15 - Synthetic Polymers. In: Ouellette RJ, Rawn JD, editors. Principles of Organic Chemistry. Boston: Elsevier; 2015. p. 397-419.
[31] Remko M, Rode BM. 1995. Ab initio study of decomposition of carbamic acid and its thio and sila derivatives. Journal of Molecular Structure:

THEOCHEM.339(1):125-131.
[32] Sousa SF, Calixto AR, Ferreira P, Ramos MJ, Lim C, Fernandes PA. 2020. Activation Free Energy, Substrate Binding Free Energy, and Enzyme Efficiency Fall in a Very Narrow Range of Values for Most Enzymes. ACS Catalysis.10(15):8444-8453.
[33] Teixeira LMC, Paiva P, Ferreira P, Rotilio L, Morth JP, Otzen DE, Fernandes PA, Ramos MJ. 2025. Bridging Experiment and Theory: A Computational Exploration of UMG-SP3 Dynamics. Pure and Applied Chemistry (Submitted to an IUPAC special issue).

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 218–226

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Quantum Emulators: CPU, single GPU and multiple GPUs performance comparison

Mathilde Chenu[a,*]

[a]*Joint Research Centre of the European Commission*

Abstract

Digital quantum computing promises new pathways for problem solving. However, currently available gate-based quantum computers still suffer from short decoherence timings and noisy results. In the meantime, quantum emulators can offer a way to test and refine quantum algorithms. They reproduce classically the behavior of quantum computers at the cost of large memory requirements: the memory needed doubles with each additional qubits emulated. Emulators can be set-up on a large range of machines, from laptops to supercomputers. In this paper, we compare the capabilities of quantum emulators built on the EuroHPC Karolina server using one CPU, one GPU or multiple GPUs in a single node. In terms of the number of qubits emulated, we show that while only 20+ qubits can be emulated using a single CPU, a single GPU brings this number to 30+. We also compare the timing performances, both for randomly generated quantum circuits and for Shor's factorization algorithm. For randomly generated circuits with same depth and width, the emulators using a GPU and eight GPUs respectively are up to 20 and 180 times faster than the CPU-based emulator. For Shor's algorithm which can have more than a million gates, the emulators using a GPU and eight GPUs respectively are up to 11 and 20 times faster than the CPU-based emulator. This benchmark of different types of quantum emulators on the Karolina server helps to choose the right parameters and make the most of the Karolina resources for future scientific work involving quantum emulation.

Keywords: Quantum computing ; Quantum emulators ; Factorization ; Shor's algorithm ; Cryptography ;

1. Introduction

Quantum technologies are rapidly developing and in particular quantum computers. Quantum computers can be of different types:

- Analog quantum simulators are measurable quantum systems used to simulate quantum physics phenomena;
- Quantum annealers are quantum systems used to find local minimums in optimization problems;

* Corresponding author.
E-mail address: mathilde.chenu@ec.europa.eu

10.1016/j.procs.2025.08.248

- Digital quantum computers are quantum systems used to perform mathematical operations on qubits through gates [1].

We will focus on *digital quantum computers* for the rest of the paper.

Digital quantum computers are rapidly developing with quantum machines having an increasing number of usable qubits. For example, the IBM machines provided 20 superconducting qubits in 2019, 133 in 2024 and plan to offer 200 by 2029 [2]. However, these qubits are noisy, meaning that they lose the information they carry over time. Gates are also imperfect meaning that the operation is implemented with a certain error. When the error rate is sufficiently low, the correct output can still be approximated from the noise-affected result, but after a certain amount of error the information is lost. Research is ongoing to find more stable qubit types and include error correction and mitigation measures during the computation to increase the accuracy of the results [3] [4].

In the meantime, quantum emulators are *classical* tools used to mimic the states of a digital quantum computer on a classical machine, at the cost of large memory and timing requirements. The memory needed to emulate qubits doubles with each additional qubit. However, these emulators provide perfect qubits representation, which won't be affected by the noise. Such emulators are very useful to test quantum algorithms at a small scale, and observe their behavior in a noise-free environment or tweak their parameters. Noise-models can also be added to test the efficiency of error-correcting codes for qubits [5].

Note that emulators are sometimes also called simulators, particularly by computer scientists. For physicists however, the term quantum simulator refers to a type of quantum computers simulating quantum systems. This is the reason why they usually use the term 'emulators' to describe the classical software tools mimicking the behavior of quantum computers. To avoid confusion, we will systematically use the term emulator for classical machines, reserving the term simulator for analog quantum simulators.

The capabilities of emulators can rapidly reach their limits, even with very large resources. For example, in [6] the authors showed that 2048 GPUs from the Julich supercomputer where necessary to emulate 40 qubits. However, emulators can be built with less resources. They won't provide as many simulated qubits as 2048 GPUs, but can nonetheless be useful to develop and improve quantum algorithms at a small scale.

In this work we realize a performance comparison between different emulators types, using a single CPU, a single GPU, and up to the 8 GPUs of one node. To compare the performances, we test the emulators on two types of quantum circuits: first on randomly generated circuits with number of qubits and depth equal, then using purposeful and longer algorithm, the Shor's factorization algorithm in our case [7]. Shor's algorithm is particularly relevant in the context of quantum computing. It can solve two of the most prominent cryptographic problems in polynomial time using a quantum computer, namely the factorization and discrete logarithm problem, when the best classical algorithms require subexponential and exponential time respectively. It is therefore a particularly studied algorithm.

We answer the following questions:

- How many qubits can be emulated using a CPU, a GPU, and up to 8 GPUs in one node?
- How are the timing performances comparing for these three emulators for randomly generated circuits?
- How are the timing performances comparing for these three emulators for Shor's algorithm?

The paper is organized as follows. We start by describing how the emulators were set-up, before presenting the performance results for randomly generated circuits. We then introduce the Shor's algorithm, describe the performances of the emulator on Shor's circuit, and conclude.

2. Setting-up the emulator

To benchmark the capabilities of a quantum emulator, we used the European High-Performance Computing Joint Undertaking server Karolina hosted in Czechia, using one node equipped with eight NVIDIA A100 GPUs and with

320GB HBM2 memory. To set-up the emulator, we used Python 3.14, along with Qiskit 1.4[1], Qiskit AerSimulator[2] and qiskit-aer-gpu module instead.

An important parameter to define when preparing the emulator is the value given to the `blocking-qubits` in the AerSimulator, which sets the largest size of the memory chunk used when parallelizing. This value should be set such that $sizeof(complex) \times 2^{(blocking\_qubits+4)}$ is smaller than the size of the smallest memory space in bytes. In our case for the Karolina's compute nodes with GPU accelerators, the size of a complex number is 32 bytes, and the size of the smallest memory space is 40960 MiB. This means that the maximal value for blocking qubits should be 26. However, the maximal value is not necessarily the optimal one.

Figure 1 illustrates what happens when the value is set too low, at 15 on the left, or too high, at 26 on the right. When the value is too low, the cost of parallelization (synchronization and communication between the processes) is greater than the benefit, making the computation with eight GPUs slower than with only one. When the value is too high, the benefit is delayed to a higher number of qubits emulated.

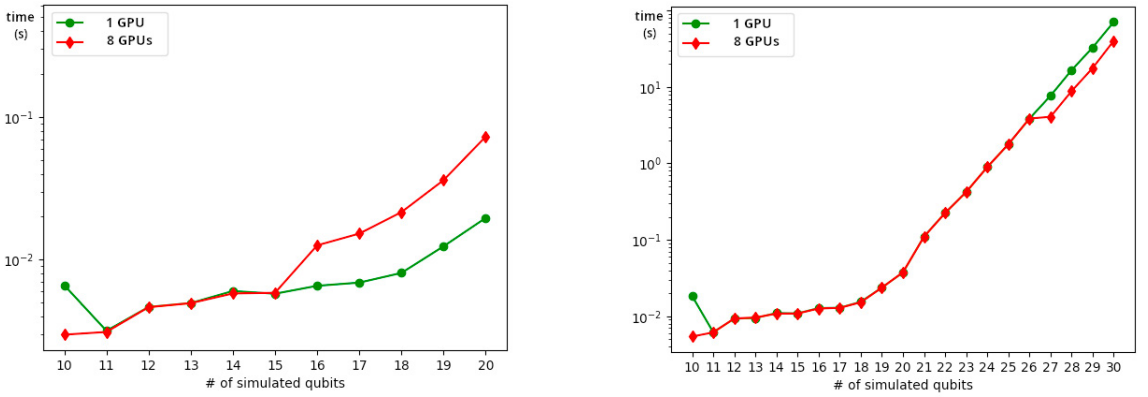


Fig. 1. Comparison of blocking qubits values: Time in seconds spent to run randomly generated quantum circuits of depth n for n qubits with blocking qubit parameter set to 15 (left), and 26 (right), with a quantum emulator on Karolina server using one GPU (ThrustGPU, NVIDIA A100) and 8 GPUs (NVIDIA A100 with 320GB HBM2 memory).

By evaluating randomly generated circuits with decreasing `blocking qubits` value starting from 26 and comparing the timing performances with one GPU and eight GPUS, we find that the optimal value for `blocking qubits` is in fact 23. In this case, the use of the eight GPUs of one node is optimal, meaning that the use of eight GPUs does not slow down the computation compared with one GPU when emulating less than 23 qubits, but does speed-up the computations when emulating circuits on 23 qubits and more.

3. Performances comparison with randomly generated square circuits

3.1. Methodology

For our performance comparison experiments on the quantum emulators, we start by studying the performances on randomly generated circuits using the Quantum Volume function. The Quantum Volume function takes a integer n as input, and outputs randomly generated circuits of depth n using n qubits. Since the circuits generated through the Quantum Volume function are relatively short, the limiting factor in this case will be the memory available to store the information of each qubits, and not the timing needed to run the circuit. Nonetheless, studying the emulation timing is useful to compare different hardware.

[1] Note that Qiskit 2.0 was released in March 2025, after our experiments.
[2] AerSimulator is an emulator and not a quantum simulator, despite its name. Computer scientists often use the term simulator for emulators.

We compare the timing results to run the Quantum Volume function for increasing n until reaching the hardware memory capacities limits using one CPU (AMD EPYC 7763, 64-core, 2.45 GHz), one GPU (NVIDIA A100) and one node of eight GPUs (NVIDIA A100 with 320GB HBM2 memory).

3.2. Results

Number of qubits. The first step of our experiment with randomly generated circuits is to determine how many qubits can be emulated with the emulators based on one CPU, one GPU and one node with 8 GPUs. When using one CPU we have been able to emulate circuits having 25 qubits. By using a single GPU, our experiment shows that up to 31 qubits can be emulated before the limit of the memory capacity is reached. When using one node with 8 GPUs and 320GB of HBM2 memory, up to 35 qubits can be emulated.

Timing comparisons. The second step is to compare the timing performances of the emulators when emulating randomly generated circuits of the same size with one CPU, one GPU and eight GPUs. It appears, unsurprisingly, that the time needed to emulate a circuit grows exponentially with the number of qubits and the size of the circuit. We observe that the computation with one GPU is 5 times faster than the computation with one CPU for 15 qubits, but the ratio grows to 20 for 25 qubits. Besides, using eight GPUs can be as much as 8 times faster than using a single GPU when emulating 31 qubits. In both cases, it appears that the gain in terms of timings grows with the number of qubits emulated.

The results are summarized on Figure 2 and Figure 3, where the number of qubits and depth of the circuits are on the x-axis, and the time spent in quantum computing is represented in seconds in logarithmic scale on the y-axis. For the results on Figure 2, the `blocking qubits` value was set to 26, the maximal possible value, which explains why one GPU and eight GPUs produce the same timing results. For the results on Figure 3, the `blocking qubits` value was set to 23, the optimal value found, which is why one GPU and eight GPUs produce the same timing results until emulation for 23 qubits and afterwards the eight GPUs start producing faster timings.

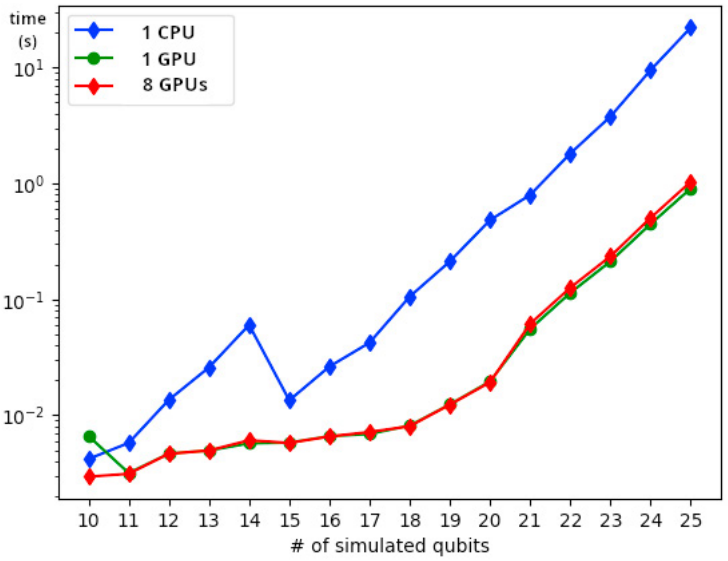


Fig. 2. Comparison CPU, GPU and GPUs: Time in seconds spent to run a randomly generated quantum circuits of n qubits and depth n with `blocking_qubits` set at 26, with a quantum emulator on Karolina server using one CPU (AMD EPYC 7763, 64-core, 2.45 GHz), one GPU (ThrustGPU, NVIDIA A100) or on node of eight GPUs (NVIDIA A100 with 320GB HBM2 memory).

These results show that 30+ qubits can already be emulated with one GPU, meaning that emulators can be built even with limited resources and without the need for a supercomputer. It also highlights the benefits of using several high quality GPUs for quantum emulation, which leads to both faster computation, but also to emulate a higher

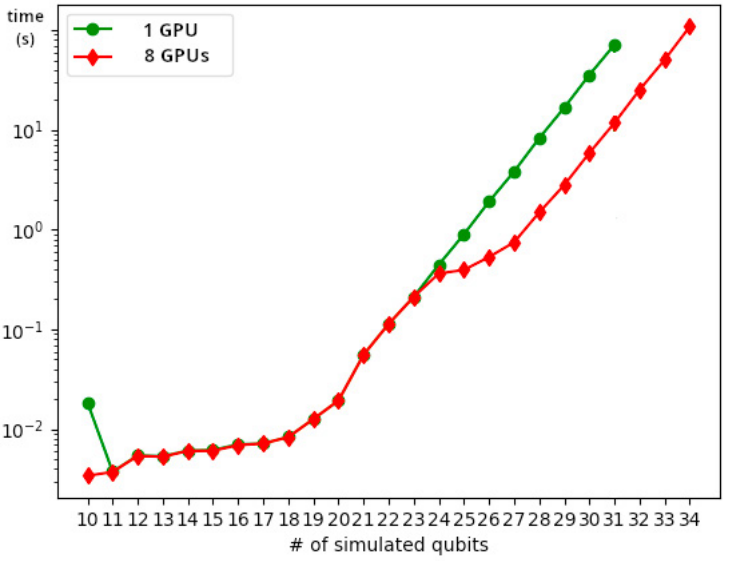


Fig. 3. Comparison GPU and GPUs: Time in seconds spent to run randomly generated quantum circuits of depth n with n qubits, `blocking_qubits` set at 23, with a quantum emulator on Karolina server one GPU (ThrustGPU, NVIDIA A100) and 8 GPUs (NVIDIA A100 with 320GB HBM2 memory).

number of qubits. Our results also provide a benchmark of the capabilities of the GPU accelerated compute nodes of the Karolina server for quantum emulation.

4. Performances comparison with Shor's algorithm

4.1. Description of Shor's algorithm

Cryptography is particularly impacted by the arrival of quantum computers. Shor's algorithm is one of the most important quantum attacks, as it can solve the factorization problem, which is the corner stone of the security of RSA algorithm [9]. In the context of RSA, the factorization problem is the following: given a known integer N such that $N = pq$, with p and q two unknown primes, find p or q. This factorization problem can be solved efficiently with Shor's algorithm as described in Algorithm 1. Note that only the *QuantumPeriodFinding* part requires a quantum computer, the rest of the algorithm is computed classically. Several easy cases can be discarded immediately. As even numbers are easily recognizable from their last digit, we can easily discard the case where p or q is equal to 2. In addition, there exist a classical algorithm to determine in linear time if N is a prime power, that is if $N = p^r$ with r an integer [8]. For these reasons, we assume from now on that $p \neq q$ and that p and q are odd primes.

We provide a brief explanation of the soundness of the algorithm. When $\gcd(a, N) = 1$, then a belongs to the multiplicative group of integers modulo N, and there exist a smallest integer r such that $a^r \equiv 1 \mod N$. Let us assume that r is even (otherwise we simply start the procedure again with a different a). Then $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$. Besides, by definition, N divides $a^r - 1$. It follows that N divides $(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$. Since $N = pq$ with p and q primes, three cases are possible:

- N divides $(a^{\frac{r}{2}} - 1)$,
- or N divides $(a^{\frac{r}{2}} + 1)$,
- or p divides $(a^{\frac{r}{2}} + 1)$ and q divides $(a^{\frac{r}{2}} - 1)$.

Algorithm 1: Shor's algorithm

Data: Integer N such that $N = pq$, $p \neq q$, with p and q odd primes
Result: p and q
found $\leftarrow$ *False* ;
while *found* = *False* **do**
  find a random a coprime with N;
  if $\gcd(a, N) \neq 1$ **then**
    found $\leftarrow$ True ;
    $p \leftarrow \gcd(a, N)$;
    $q \leftarrow \frac{N}{p}$;
    return *(p, q)*;
  else
    $r \leftarrow$ *QuantumPeriodFinding*(a, N);
    if *r is even* **then**
      $d \leftarrow \gcd(a^{\frac{r}{2}} + 1, N)$;
      if $d \neq N$ **then**
        found $\leftarrow$ True ;
        $p \leftarrow d$;
        $q \leftarrow \frac{N}{p}$;
        return *(p, q)*;

The first option contradicts the fact that the order of a is r, hence we can discard it. With the second option, $\gcd(a^{\frac{r}{2}} + 1, N) = N$, but this does not give us a factor of N, meaning that the algorithm needs to restart with another a. In the third case, $\gcd(a^{\frac{r}{2}} + 1, N) = p$ and $\gcd(a^{\frac{r}{2}} - 1, N) = q$, and we have found the factors of N.

We now briefly describe the *QuantumPeriodFinding* quantum algorithm. It uses the following unitary operator:

$$U|k\rangle = \begin{cases} |ak \mod N\rangle & 0 \leq k < N, \\ |k\rangle & N \leq k < 2^n \end{cases}$$

Let r be the order of $a \mod N$. Then U^r is the identity, which means that the eigenstates and associated eigenvalues of U are respectively $|\psi_j\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2i\pi jk/r} |a^k \bmod N\rangle$ and $\omega_j = e^{-2i\pi j/r}$ for $0 < j < r - 1$.

Note that the sum of the eigenstates is equal to $|1\rangle$.

$$\begin{aligned} \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |\psi_j\rangle &= \frac{1}{r} \sum_{j=0}^{r-1} \sum_{k=0}^{r-1} e^{-2\pi i(jk/r)} |a^k \bmod N\rangle \\ &= \frac{1}{r} \sum_{k=0}^{r-1} \left(\sum_{j=0}^{r-1} e^{-2\pi i(jk/r)} \right) |a^k \bmod N\rangle \\ &= \frac{1}{r} \sum_{j=0}^{r-1} |1\rangle + \frac{1}{r} \sum_{k=1}^{r-1} \left(\sum_{j=0}^{r-1} e^{-2\pi i(jk/r)} \right) |a^k \bmod N\rangle \\ &= |1\rangle \end{aligned}$$

We now use the quantum phase estimation as a subroutine. Let U be a unitary operator and ψ an eigenstate such that $U|\psi\rangle = \omega|\psi\rangle$, with $\omega = e^{2i\pi\theta}$. The phase estimation algorithm takes as input U and ψ and returns with high-probability the phase θ of the eigenvalue ω.

We cannot run the quantum phase estimation on the eigenstates directly as r is unknown. However, we can apply it on $|1\rangle$, which is equivalent to applying it to $\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}|\psi_j\rangle$. We can then obtain $\frac{j}{r}$ for one of the j, all j being equiprobable. Using the (classical) continued-fraction decomposition algorithm, we can retrieve the value r from the output of the phase estimation algorithm.

More details about the Shor's algorithm, and in particular its quantum part, can be found in [10].

4.2. Methodology

We choose to implement the Shor's algorithm for any integer, meaning that our circuit construction is generic and adaptable to any integer size. While this naturally leads to longer timings, it is also closer to how Shor's algorithm would be used in practice when attacking cryptography public keys. To build our code, we updated and refined the implementations of Ignazio Pedone `https://github.com/ignaziopedone/shor-analysis/blob/main/RSA/Shor_library.py` and `https://github.com/ttlion/ShorAlgQiskit`. The code is available upon request.

Two approaches are possible when building generic implementations of the Shor's algorithm. The first approach is a more concise circuit requiring $4n + 1$ qubits to factorize a n-bit integer. A sequential version has more gates, but requires only $2n + 2$ qubits. As the limitation in the case of quantum emulators is the number of qubits that can be emulated, we favor the second sequential version of the algorithm for our experiment. The algorithm is run entirely, both classical and quantum parts, to verify the correctness of the result. However we only measure the time spent in quantum computing when running on emulators based on a CPU, one GPU and one node of eight GPUS. We reuse the optimal parameter found when setting-up the emulator, that is, we set `blocking qubits` to 23. We run the quantum part of the generic Shor's factorization algorithm, that is the quantum period finding algorithm, for increasing integers N selected such that N is the product of two primes.

4.3. Results

The detailed results of the experiment are presented on Table 1, with the integers factorized N, their factorization, their bit-length $n = \log_2 N$, the number of qubits necessary to run Shor's algorithm which is $(2n + 2)$, and the time needed to run ten shots of the quantum algorithm using one CPU (AMD EPYC 7763, 64-core, 2.45 GHz), one GPU (NVIDIA A100), and 8 GPUs (NVIDIA A100 with 320GB HBM2 memory).

For efficiency reasons, we used only the experiments running in less than one hour and a half, or 5400 seconds. This limit was reached after factorizing $N = 707$ for CPU experiments, and after factorizing $N = 3649$ for the experiment with one and eight GPUs. In theory, larger integers could be factorized using the CPU, GPU and GPUs-based emulators.

All circuits provided the correct answer with high probability in the noise-free environment with 10 shots for each experiment.[3]

From these results we can conclude that:

- The three quantum emulators (CPU, GPU and GPUs) can successfully run quantum circuits with depth larger than one million gates, and provide the correct results. In comparison, quantum computers running circuits with the similar depth are very likely to return the wrong output due to the decoherence noise. For this reason, the emulator remains useful to study the behavior of quantum circuits in an ideal noise-free environment, despite the availability of quantum computers.
- The time needed to factorize integer grows exponentially with the bit-length of the integer factorized on a quantum emulator. It is the time rather than the memory that limits the usefulness of the quantum emulator in the case of Shor's algorithm. In theory, 34 emulated qubits would be able to factorize a sixteen-bit number, but in at least eight hours.

[3] Shor's algorithm is probabilistic and needs to be run several times as several parameters a might need to be tried.

Integer and factorization	Length (in bits)	Number of qubits needed	Circuit depth	Timing CPU (in secondes)	Timing GPU (in secondes)	Timing GPUs (in secondes)
$15 = 3 \times 5$	4	10	6608	0.34468	0.79939	1.4782
$21 = 3 \times 7$	5	12	12944	1.1485	0.80006	0.72214
$55 = 5 \times 11$	6	14	24439	2.8531	1.7354	1.5057
$111 = 3 \times 37$	7	16	48803	14.248	3.7407	3.3093
$221 = 13 \times 17$	8	18	114576	70.323	11.679	8.8438
$391 = 17 \times 23$	9	20	333348	416.43	44.002	31.723
$707 = 7 \times 101$	10	22	1148566	2428.4	206.20	131.35
$1763 = 41 \times 43$	11	24	4330884	-	967.47	583.79
$3649 = 41 \times 89$	12	26	16959474	-	4761.0	2645.5

Table 1. Shor's algorithm performances: integers factorized and their bit-length, the number of qubits and circuit-depth needed to run Shor's factorization algorithm, and the time in seconds spent in quantum computing with a quantum emulator on Karolina server using CPU (AMD EPYC 7763, 64-core, 2.45 GHz), one GPU (ThrustGPU, NVIDIA A100), and 8 GPUs (NVIDIA A100 with 320GB HBM2 memory)

- Using one GPU is up to 11 times faster compared to using one CPU. Using eight GPUs is up to 18 times faster than using a CPU, and almost up to twice as fast as one GPU. The larger the number of qubits used, the greater the gain. The speed-up obtained for random circuits short circuits is higher than for longer specific circuits like Shor's.

To highlight the exponential growth of the time needed to factories an integer with respect to its bit-length, Figure 4 summaries the results of Table 1 with the integers factorized on the x-axis, and the time needed to run ten shots of the quantum period finding algorithm on the y-axis with a logarithmic scale.

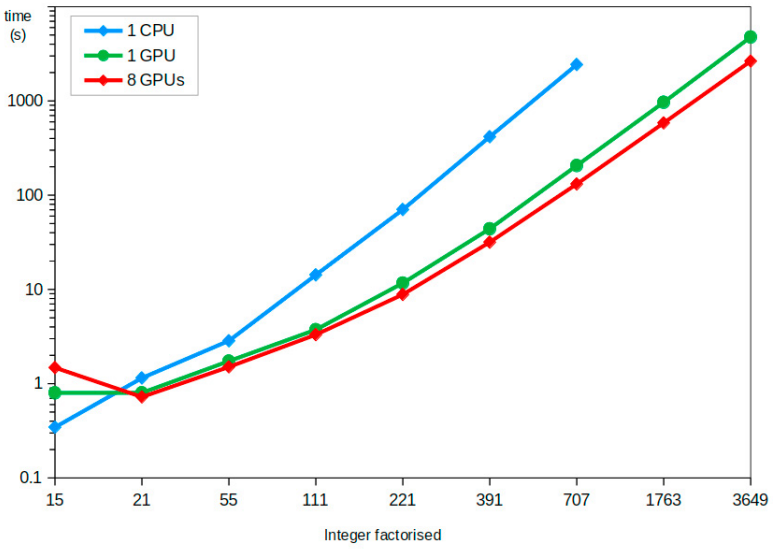


Fig. 4. Shor's algorithm performances: Time in seconds spent to run the quantum part of Shor's factorization algorithm (10 shots) for increasing integers with a quantum emulator on Karolina server using CPU (AMD EPYC 7763, 64-core, 2.45 GHz), one GPU (ThrustGPU, NVIDIA A100), and 8 GPUs (NVIDIA A100 with 320GB HBM2 memory)

5. Conclusion

In this work, we successfully set up quantum emulators using one CPU, one GPU or one node with eight GPUs on EuroHPC's Karolina server and compared their performances.

While a CPU is sufficient to emulate 25+ qubits, using a single GPU allows to emulate 30+ qubits, and a node of eight GPUs reaches 35 qubits.

When tested on randomly generated circuits of depth n using n qubits for increasing integers n, a single NVIDIA A100 GPU produces an emulator already up to 20 times faster than the CPU-based emulator. An emulator using eight NVIDIA A100 GPUs within one node is 8 times faster than a single GPU emulator, and 160 times faster than the CPU-based emulator.

The benefit of using GPUs also translates from randomly generated quantum algorithms to the specific case of Shor's algorithm, which can have millions of gates, although in smaller proportions. We observe that the emulator with eight GPUs is up to 18 times faster than the CPU-based emulator, and almost twice as fast as the GPU-based emulator.

Our experiment quantify the benefit in term of speed and number of qubits emulated when using one or eight high quality GPUs compared to one CPU, highlighting the benefits of high quality GPUs hardware when emulating quantum computers with EuroHPC's Karolina server. This benchmark of different types of quantum emulators on the Karolina server will help to make the right choice of parameters and make the most of the computing resources for future scientific work involving quantum emulation.

Acknowledgements

We would like to thank the EuroHPC Joint Undertaking for providing us with access to the Karolina server, and for their supportive team. We are also grateful to Dr. Mario Chizzini from the Joint Research Centre for his generous assistance in clarifying numerous paradoxes in quantum physics. Last but not least, we would like to thank the anonymous reviewers for their valuable comments and feedback.

References

[1] Dowling, Jonathan, and Milburn, Gerard. (2003) "Quantum technology: the second quantum revolution." *Phil. Trans. R. Soc. A.*

[2] IBM. (2024) "Quantum roadmap to anticipate the future of quantum-centric supercomputing." https://www.ibm.com/quantum/blog/ibm-quantum-roadmap-2025

[3] Ryutaroh Matsumoto, and Manabu Hagiwara. (2021) " A Survey of Quantum Error Correction." *IEICE TRANSACTIONS on Fundamentals*

[4] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. OBrien.(2023) "Quantum error mitigation." *Rev. Mod. Phys. 95*

[5] Alessio Cicero, Mohammad Ali Maleki, Muhammad Waqar Azhar, Anton Frisk Kockum, and Pedro Trancoso. (2024) "Simulation of Quantum Computers: Review and Acceleration Opportunities." *Arxiv* https://arxiv.org/abs/2410.12660

[6] Dennis Willsch, Madita Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. (2023) " Large-Scale Simulation of Shors Quantum Factoring Algorithm." *Mathematical Perspectives on Quantum Computing and Communication*

[7] Peter Shor. (1994) "Polynomial time algorithms for discrete logarithms and factoring on a quantum computer." *Algorithmic Number Theory, ANTS*

[8] Daniel Bernstein. (1998) "Detecting perfect powers in essentially linear time." *Math. Comput.*

[9] Ronald Rivest, Adi Shamir, and Leonard Adleman. (1978) "A Method for Obtaining Digital Signatures and Public-key Cryptosystems." *Commun. ACM*

[10] Ray LaPierre. (2021) "Introduction to Quantum Computing." *Springer*

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 227–236

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Improving the scalability of a high-order atmospheric dynamics solver based on the `deal.II` library

Giuseppe Orlando[a,*], Tommaso Benacchio[b,*], Luca Bonaventura[c]

[a] *CMAP, CNRS, École polytechnique, Institut Polytechnique de Paris, Route de Saclay, 91120 Palaiseau, France*
[b] *Weather Research, Danish Meteorological Institute, Sankt Kjelds Plads 11, 2100 Copenhagen, Denmark*
[c] *Dipartimento di Matematica, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy*

Abstract

We present recent advances on the massively parallel performance of a numerical scheme for atmosphere dynamics applications based on the `deal.II` library. The implicit-explicit discontinuous finite element scheme is based on a matrix-free approach, meaning that no global sparse matrix is built and only the action of the linear operators on a vector is actually implemented. Following a profiling analysis, we focus on the performance optimization of the numerical method and describe the impact of different preconditioning and solving techniques in this framework. Moreover, we show how the use of the latest version of the `deal.II` library and of suitable execution flags can improve the parallel performance.

Keywords: Numerical Weather Prediction; Non-conforming meshes; Flows over orography; IMEX schemes; `deal.II`

1. Introduction

Efficient numerical simulations of atmospheric flows are key to viable weather and climate forecasts and several related practical applications. Medium-range global numerical weather predictions (NWP) up to ten days ahead are typically required to complete within a one-hour operational threshold in the forecast cycles of meteorological centres. In addition, the computational meshes employed for these forecasts are being refined to head towards the km-scale, while experimental limited-area suites already reach the hectometric scale [10]. Hence, in the current context of increasing demand of computational resources driven by increasingly high spatial resolutions, massively parallel model scalability is a fundamental requirement.

In this work, we present recent advances in the parallel performance of a Implicit-Explicit Discontinuous Galerkin (IMEX-DG) solver [13] employed for atmosphere dynamics applications [14, 15, 16], following up on earlier analyses by the authors [17]. The implementation is carried out in the framework of the `deal.II` library [1, 2, 3]. First, we

* Corresponding author.
E-mail address: giuseppe.orlando@polytechnique.edu, tbo@dmi.dk

1877-0509
10.1016/j.procs.2025.08.249

show how the use of suitable preconditioning techniques can significantly improve the performance of the elliptic pressure solver. More specifically, we measure the impact of a novel preconditioning technique for the linear system associated with the pressure field that exploits the numerical formulation of the problem. Next, we assess the impact of activating execution flags and of using the most recent version of the library (9.6.2, [1]). Finally, we present the result of employing a different strategy for the solution of a linear system.

The paper is structured as follows. In Section 2, we briefly review the model equations and the relevant details of the numerical methodology used in this work. The profiling and parallel performance analysis in numerical experiments with a three-dimensional benchmark of atmospheric flow are presented in Section 3. Finally, some conclusions are reported in Section 4.

2. The model equations and some details of the numerical method

The compressible Euler equations of gas dynamics represent the most comprehensive mathematical model for atmospheric flows [20]. Considering for simplicity the dry, non-rotating case, the mathematical model reads as follows:

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\,\mathbf{u}) = 0 \\
&\frac{\partial(\rho\,\mathbf{u})}{\partial t} + \nabla\cdot(\rho\,\mathbf{u}\otimes\mathbf{u}) + \nabla\, p = \rho\mathbf{g} \\
&\frac{\partial(\rho E)}{\partial t} + \nabla\cdot\left[(\rho h + \rho k)\,\mathbf{u}\right] = \rho\mathbf{g}\cdot\mathbf{u},
\end{aligned} \tag{1}$$

where $t \in (0, T_f]$ is the temporal coordinate, supplied with suitable initial and boundary conditions on the computational domain. Here $\otimes$ denotes the tensor product, T_f is the final time, ρ is the density, $\mathbf{u}$ is the fluid velocity, and p is the pressure. Moreover, $\mathbf{g} = -g\mathbf{k}$ is the acceleration of gravity, with $g = 9.81\,\mathrm{m\,s^{-2}}$ and $\mathbf{k}$ being the upward pointing unit vector in the standard Cartesian frame of reference. Finally, E denotes the total energy per unit of mass, h is the specific enthalpy and $k = 1/2\,|\mathbf{u}|^2$ is the specific kinetic energy. System (1) is complemented by the equation of state of ideal gases, given by $\rho e = \frac{1}{\gamma-1} p$, where e denotes the specific internal energy $e = E - k$ and the isentropic exponent γ is taken equal to 1.4.

The time discretization is based on an Implicit-Explicit (IMEX) Runge-Kutta (RK) method [6], which is a widely employed approach for ODE systems that include both stiff and non-stiff components. IMEX-RK schemes are represented compactly by the companion Butcher tableaux:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^\top \end{array} \qquad \begin{array}{c|c} \tilde{\mathbf{c}} & \tilde{\mathbf{A}} \\ \hline & \tilde{\mathbf{b}}^\top \end{array}$$

with $\mathbf{A} = \{a_{lm}\}, \mathbf{b} = \{b_l\}, \mathbf{c} = \{c_l\}$, $l, m = 1 \dots s$, denoting the coefficients of the explicit method, $\tilde{\mathbf{A}} = \{\tilde{a}_{lm}\}, \tilde{\mathbf{b}} = \left\{\tilde{b}_l\right\}$, and $\tilde{\mathbf{c}} = \{\tilde{c}_l\}$, $l, m = 1 \dots s$ denoting the coefficients of the implicit method, and s representing the number of stages of the method. We refer, e.g., to [6, 19] for a detailed analysis of the order and coupling conditions of the two companion schemes. A generic IMEX-RK l stage with time step Δt for the Euler equations reads therefore as follows:

$$\begin{aligned}
&\rho^{(l)} = \rho^n - \sum_{m=1}^{l-1} a_{lm}\Delta t\,\nabla\cdot\left(\rho^{(m)}\,\mathbf{u}^{(m)}\right) \\
&\rho^{(l)}\mathbf{u}^{(l)} + \tilde{a}_{ll}\Delta t\,\nabla\, p^{(l)} = \mathbf{m}^{(l)} \\
&\rho^{(l)}E^{(l)} + \tilde{a}_{ll}\Delta t\,\nabla\cdot\left(h^{(l)}\rho^{(l)}\,\mathbf{u}^{(l)}\right) = \hat{e}^{(l)},
\end{aligned} \tag{2}$$

where we have set

$$\mathbf{m}^{(l)} = \rho^n\,\mathbf{u}^n - \sum_{m=1}^{l-1} a_{lm}\Delta t\,\nabla\cdot\left(\rho^{(m)}\,\mathbf{u}^{(m)}\otimes\mathbf{u}^{(m)}\right) - \sum_{m=1}^{l-1}\tilde{a}_{lm}\Delta t\,\nabla\, p^{(m)} \tag{3}$$

$$\hat{\mathbf{e}}^{(l)} = \rho^n E^n - \sum_{m=1}^{l-1}\tilde{a}_{lm}\Delta t\,\nabla\cdot\left(h^{(m)}\rho^{(m)}\,\mathbf{u}^{(m)}\right)) - \sum_{m=1}^{l-1} a_{lm}\Delta t\,\nabla\cdot\left(k^{(m)}\rho^{(m)}\,\mathbf{u}^{(m)}\right) \tag{4}$$

Notice that, substituting formally $\rho^{(l)}\,\mathbf{u}^{(l)}$ into the energy equation and taking into account the definitions $\rho E = \rho e + \rho k$ and $h = e + p/\rho$, the following nonlinear Helmholtz-type equation for the pressure is obtained:

$$\rho^{(l)}\left[e(p^{(l)},\rho^{(l)}) + k^{(l)}\right] - \tilde{a}_{ll}^2\Delta t^2\,\nabla\cdot\left[\left(e(p^{(l)},\rho^{(l)}) + \frac{p^{(l)}}{\rho^{(l)}}\right)\nabla p^{(l)}\right] + \tilde{a}_{ll}\Delta t\,\nabla\cdot\left[\left(e(p^{(l)},\rho^{(l)}) + \frac{p^{(l)}}{\rho^{(l)}}\right)\mathbf{m}^{(l)}\right] = \hat{e}^{(l)}, \tag{5}$$

which is solved through a fixed point procedure. We refer to [12, 13, 18] for further references and details on the theoretical properties of the method.

The spatial discretization is based on a Discontinuous Galerkin (DG) method [5], which combines high-order accuracy and flexibility in a highly data-local framework. In particular, this method is particularly well suited for mesh adaptive approaches [14, 15]. The shape functions correspond to the products of Lagrange interpolation polynomials for the support points of $(r+1)$-order Gauss-Lobatto quadrature rule in each coordinate direction, with r denoting the polynomial degree. The discrete formulation of method (2) can therefore be expressed as

$$\mathbf{A}^{(l)}\mathbf{U}^{(l)} + \mathbf{B}^{(l)}\mathbf{P}^{(l)} = \mathbf{F}^{(l)} \tag{6}$$

$$\mathbf{C}^{(l)}\mathbf{U}^{(l)} + \mathbf{D}^{(l)}\mathbf{P}^{(l)} = \mathbf{G}^{(l)}, \tag{7}$$

where $\mathbf{U}^{(l)}$ represents the vector of degrees of freedom of the velocity field, while $\mathbf{P}^{(l)}$ is the vector of the degrees of freedom of the pressure field. We refer to [12, 14, 17] for the detailed expression of all the matrices and vectors. We report only the expression of the matrix $A^{(l)}$ that will be useful for the discussion in Section 3:

$$A_{ij}^{(l)} = \sum_{K\in\mathcal{T}_{\mathcal{H}}}\int_K \rho^{(l)}\boldsymbol{\varphi}_j\cdot\boldsymbol{\varphi}_i\mathrm{d}\mathbf{x}, \tag{8}$$

where $\mathcal{T}_{\mathcal{H}}$ represents the computational mesh, $\boldsymbol{\varphi}_i$ denotes the basis function of the space of polynomial functions employed to discretize the velocity, and $\mathrm{d}\mathbf{x}$ is the spatial element of integration. Formally, from (6) one can derive

$$\mathbf{U}^{(l)} = \left(\mathbf{A}^{(l)}\right)^{-1}\left(\mathbf{F}^{(l)} - \mathbf{B}^{(l)}\mathbf{P}^{(l)}\right), \tag{9}$$

and obtain

$$\mathbf{D}^{(l)}\mathbf{P}^{(l)} + \mathbf{C}^{(l)}\left(\mathbf{A}^{(l)}\right)^{-1}\left(\mathbf{F}^{(l)} - \mathbf{B}^{(l)}\mathbf{P}^{(l)}\right) = \mathbf{G}^{(l)}. \tag{10}$$

The above system is then solved following the fixed point procedure described in [13]. In Sections 3.1 and 3.3 we describe the impact of using different preconditioning and solving techniques for system (10).

3. Numerical results

In this Section, we show results of simulations of an idealized three-dimensional test case of atmospheric flow over orography [11, 15] using the numerical method described in the previous Section and focusing on its scalability. The simulations have been run using up to 1024 2x AMD EPYC Rome 7H12 64c 2.6GHz CPUs at MeluXina HPC facility[1] and OpenMPI 4.1.5 has been employed. The compiler is GCC version 12.3 and the Vectorization level is 256 bits.

[1] https://docs.lxp.lu/

We consider a three-dimensional configuration of a flow over a bell-shaped hill, already studied in [11, 14, 15]. The computational domain is $(0, 60) \times (0, 40) \times (0, 16)$ km. The mountain profile is defined as follows:

$$h(x, y) = h_c \left[1 + \left(\frac{x - x_c}{a_c} \right)^2 + \left(\frac{y - y_c}{a_c} \right)^2 \right]^{-\frac{3}{2}}, \tag{11}$$

with $h_c = 400\,\text{m}, a_c = 1\,\text{km}, x_c = 30\,\text{km}$, and $y_c = 20\,\text{km}$. The buoyancy frequency is $N = 0.01\,\text{s}^{-1}$, whereas the background velocity is $\overline{u} = 10\,\text{m}\,\text{s}^{-1}$. The final time is $T_f = 1\,\text{h}$. The initial conditions read as follows [4]:

$$p = p_{ref} \left\{ 1 - \frac{g}{N^2} \Gamma \frac{\rho_{ref}}{p_{ref}} \left[1 - \exp\left(-\frac{N^2 z}{g} \right) \right] \right\}^{1/\Gamma}, \qquad \rho = \rho_{ref} \left(\frac{p}{p_{ref}} \right)^{1/\gamma} \exp\left(-\frac{N^2 z}{g} \right), \tag{12}$$

where $p_{ref} = 10^5\,\text{Pa}$ and $\rho_{ref} = \frac{p_{ref}}{R T_{ref}}$, with $T_{ref} = 293.15\,\text{K}$. Finally, we set $\Gamma = \frac{\gamma - 1}{\gamma}$. We refer to [15] for the implementation of boundary conditions. Unless differently stated, we consider a) a uniform mesh composed of $N_{el} = 120 \times 80 \times 32 = 307200$ elements with polynomial degree $r = 4$, leading to about 38.5 million unknowns for each scalar variable and a resolution of 125 m; and b) a non-conforming mesh composed of $N_{el} = 204816$ elements, yielding around 25.6 millions of unknowns for each scalar variable and a maximum resolution of 62.5 m. A non-conforming mesh is characterized by neighbouring cells with different resolution on both the horizontal and the vertical direction [15, 17] and its use provides sizeable computational savings [15, 17]. The following optimization flags are employed throughout the runs:

```
-O2 -funroll-loops -funroll-all-loops -fstrict-aliasing -Wno-unused-local-typedefs
```

which are the standard ones in the Release mode of `deal.II` (see https://www.dealii.org/developer/users/cmake_user.html). Unfortunately, due to limitations on computational resources, the scaling analyses reported in this Section could only be performed once for each of the different configurations.

3.1. Impact of preconditioning technique

The preconditioned conjugate gradient method with a geometric multigrid preconditioner is employed to solve the symmetric positive definite linear systems, while the GMRES solver with a Jacobi preconditioner is employed for the solution of non-symmetric linear systems. A matrix-free approach is employed [2], meaning that no global sparse matrix is built and only the action of the linear operators on a vector is actually implemented. This strategy is very efficient for high-order discretization methods [7]. However, it leads to some difficulties in the preconditioning, since the matrix is not available and standard techniques as ILU cannot be employed. For symmetric positive-definite systems, an efficient preconditioning technique compatible with the matrix-free approach is based on the so-called Chebyshev polynomial that relies on an estimate of the eigenvalues of the matrix so as to damp the eigenvalue range [9].

The approach based on the eigenvalue estimate cannot be directly employed for the solution of (10) because it requires symmetry and positive definiteness of the preconditioned matrix [21]. Moreover, the entries of the matrices $(\mathbf{A}^{(l)})^{-1}$, $\mathbf{D}^{(l)}$, and $\mathbf{C}^{(l)}$ are not readily available because of the matrix-free approach. In a matrix-free framework, a matrix representation of the operator can be obtained by applying the operator on all unit vectors. Apparently, this is a very inefficient procedure because it requires to perform n operator evaluations for a $n \times n$ matrix. In practice, the integration is so efficient that the computation completes without significant overhead [2]. However, using this procedure

[2] https://www.dealii.org/current/doxygen/deal.II/step_37.html

for the computation of $(\mathbf{A}^{(l)})^{-1}$, $\mathbf{D}^{(l)}$, and $\mathbf{C}^{(l)}$ turns out to be very inefficient because it requires the evaluation of three operators as well as the solution of a linear system at each operator evaluation to compute $(\mathbf{A}^{(l)})^{-1}$. Hence, the simplest solution consists in considering only the contribution provided by $\mathbf{D}^{(l)}$, which takes into account the internal energy. For low Mach number flows, as those considered in this work, this approximation can be considered reliable since velocities are typically low and the internal energy is therefore dominant with respect to the kinetic energy. This is the strategy adopted, e.g., in [15], and we refer to it as *internal energy preconditioner*.

We consider here a novel alternative approach that can be easily implemented in a matrix-free framework and consists of defining an operator that takes into account the underlying nonlinear Helhmholtz-type equation (5) when solving (10). More specifically, we consider the volumetric contribution of the weak form associated to a DG discretization of (5). Hence, the preconditioner is the inverse of the diagonal part of the matrix $\tilde{\mathbf{S}}^{(l)}$ whose entries are

$$\tilde{S}_{ij}^{(l)} = \sum_{K\in\mathcal{T}_{\mathcal{H}}} \int_K \rho^{(n,l)} e\left(\Psi_j, \rho^{(l)}\right) \Psi_i \mathrm{d}\mathbf{x} + \sum_{K\in\mathcal{T}_{\mathcal{H}}} \int_K \tilde{a}_{ll}^2 \Delta t^2 \left(e\left(p^{(l)}, \rho^{(l)}\right) + \frac{p^{(l)}}{\rho^{(l)}} \right) \nabla \Psi_j \cdot \nabla \Psi_i \mathrm{d}\mathbf{x}. \tag{13}$$

where Ψ_i denotes the basis function of the space of polynomial functions employed to discretize the pressure. Matrix $\tilde{\mathbf{S}}^{(l)}$ is the sum of two contributions: a first one that considers the internal energy and that coincides with matrix $\mathbf{D}^{(l)}$, and a second one that considers the contribution due to specific enthalpy and takes into account the underlying elliptic character of (5). The pressure $p^{(l)}$ in (13) in the fixed point procedure is evaluated at the previous fixed point iteration. We refer to this strategy as *Helmholtz preconditioner*.

An analysis of time to solution using the two preconditioning techniques reveals that, when using a uniform mesh, the use of the Helmholtz preconditioner gives a relatively modest computational time saving - about 18% with 128 cores (1 node) - that decreases for higher core counts (Figure 1). A more sizeable computational time saving is established for the non-conforming mesh which amounts to 60% employing 128 cores and to 46% employing 2048 cores (Figure 1). Hence, strategy *Helmholtz preconditioner* provides a significant advantage in terms of time-to-solution, in particular for non-conforming meshes. In the following, we will restrict our attention only to this preconditioning technique and to experiments using the non-conforming mesh.

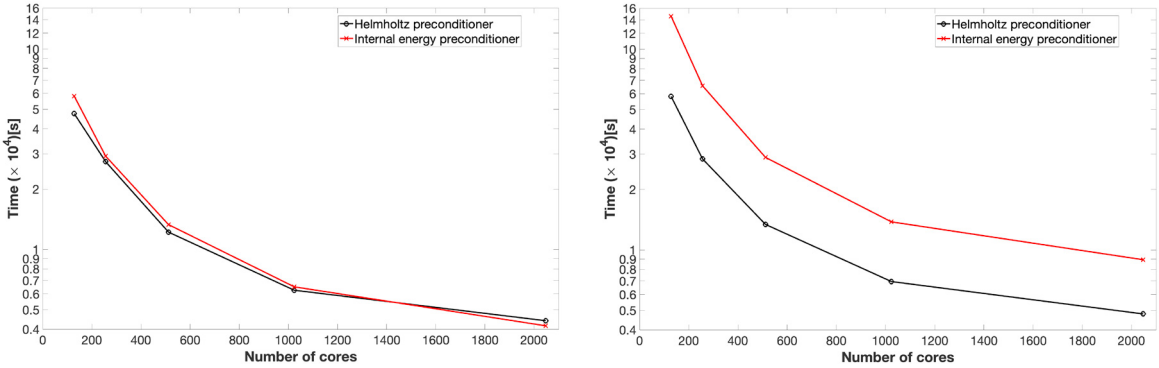


Fig. 1: Analysis of preconditioning technique, wall-clock time in the numerical solution of problem (2), flow over a 3D hill test case. Left: uniform mesh. Right: non-conforming mesh. The black lines show the results with the *Helmholtz preconditioner* approach, whereas the red lines show the results with *internal energy preconditioner* approach.

3.2. Impact of execution flags and version

Next, we analyze the impact of some execution flags and of the use of the `deal.II` 9.6.2 version [1] instead of the `deal.II` 9.5.2 version [2]. Thanks to the support of EPICURE team at MeluXina, it has been shown that the execution flags

```
export DEAL_II_NUM_THREADS="$SLURM_CPUS_PER_TASK" export OMP_NUM_THREADS="$SLURM_CPUS_PER_TASK"
```

improve the parallel performance of the solver in terms of both strong scaling and wall-clock time (Figure 2). However, an even more sizeable improvement in the parallel performance is obtained using the `deal.II` 9.6.2 version, leading to a computational time saving of around 28% compared to using the 9.5.2 version in a run with 2048 cores (i.e. 16 nodes). For the results in the following Section 3.3, `deal.II` 9.6.2 version with the optimal execution flags is therefore employed.

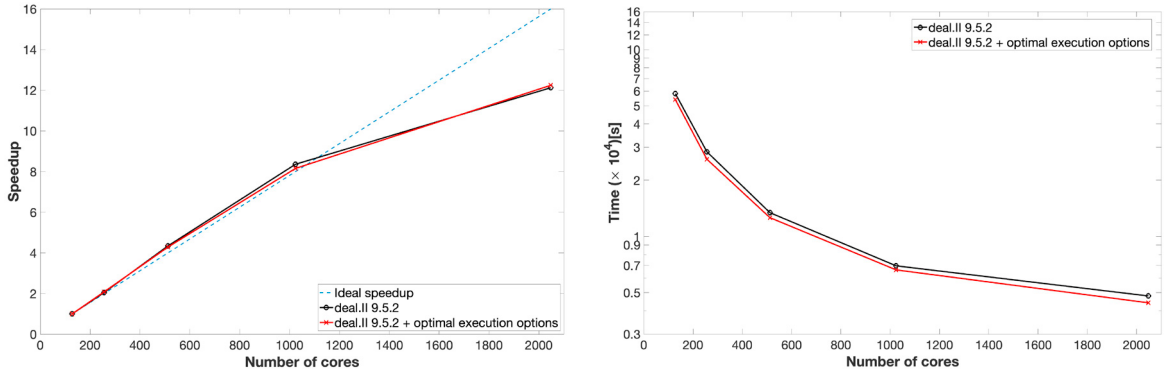


Fig. 2: Performance impact of using different execution flags in the numerical solution of problem (2), flow over a 3D hill test case. Left: strong scaling. Right: WT time. The results are obtained with the non-conforming mesh. The black lines show the results with the standard execution flags, whereas the red lines show the results with optimal execution flags.

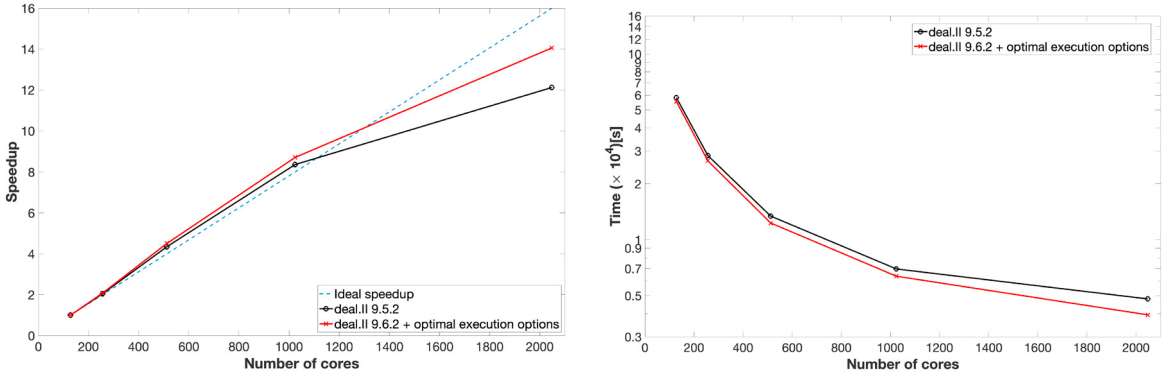


Fig. 3: Performance impact of using different `deal.II` versions in the numerical solution of problem (2), flow over a 3D hill test case. Left: strong scaling. Right: WT time. The results are obtained with the non-conforming mesh. The black lines show the results obtained using the `deal.II` 9.6.2 version and with optimal execution flags, whereas the red lines show the results obtained using the `deal.II` 9.5.2 version with standard execution flags.

3.3. Fast evaluation of block-diagonal operators

Finally, we have investigated the distribution of the computational time spent in the different part of the algorithms. A profiling analysis show that, as expected, the vast majority of execution time is spent in solving problem (10) to compute the pressure field (Figure 4). The results are obtained using the non-conforming mesh, but analogous considerations hold for the uniform mesh (not shown). Apart from the computational time spent in computing the pressure field, one can immediately notice an increasing relative cost of computing the velocity field (orange bar in Figure 4) which ranges from about 3.6% with 128 cores to about 13% with 2048 cores. This part of the algorithm consists in computing $(\mathbf{A}^{(l)})^{-1}$ (we refer to [13, 14] for all the details), hence solving a linear system. However, this

operation is repeated several times in the fixed point loop because of the matrix-free approach, which explains the increasing computational burden.

The matrix $\mathbf{A}^{(l)}$ (8) is a simple block-diagonal operator. Efficient techniques to evaluate and to invert block-diagonal operators in a DG framework were presented in [7]. However, they require the use of the same number of quadrature points and degrees of freedom along each coordinate direction, since this yields, after a change of basis, diagonal blocks in the algebraic structure of the the block-diagonal operator [7]. More specifically, as explained in [7, 8], the block-diagonal operator is inverted as $\mathbf{S}^{-T}\mathbf{J}^{-1}\mathbf{S}^{-1}$. Here $\mathbf{J}$ is a diagonal matrix, whose entries are equal to the determinant of the Jacobian times the quadrature weight, while $\mathbf{S}$ is a square matrix with basis functions in the row and quadrature points in columns, i.e. $S_{ij} = \varphi_i(\mathbf{x}_j)$. The matrix $\mathbf{S}$ is then constructed as the Kronecker product (tensor product) of small one-dimensional matrices that can be inverted efficiently at each stage. In a nodal DG framework with nodes located at $r+1$ on the quadrature points of a Gauss-Legendre-Lobatto formula, as the one considered here, this means that it is possible to employ a Gauss-Legendre formula with $r+1$ quadrature points along each coordinate direction for the numerical integration.

The main drawback of this approach is that an aliasing error is introduced. Since numerical integration based on the Gauss-Legendre formula with $r+1$ quadrature points integrates exactly polynomials up to degree $2r+1$, the strategy presented in [7] does not introduce aliasing errors for classical mass matrices that consist of the product of two basis function. However, the matrix $\mathbf{A}^{(l)}$ (8) is a modified mass matrix and consists of the product between three polynomials. Hence, an aliasing error is introduced for $r > 1$. The numerical integration for the results shown so far is based on the so-called consistent integration or over-integration (see [16] for further details). More specifically, a Gauss-Legendre formula with $2r+1$ quadrature points along each coordinate direction is employed. Notice that the choice to consider a polynomial representation for density, velocity, and pressure avoids any polynomial division in the case of the ideal gas law and therefore all the integrals can be computed without aliasing errors thanks to the aforementioned consistent integration. Hence, the choice of the numerical quadrature formula is related to the need to integrate all the terms so as to avoid aliasing errors.

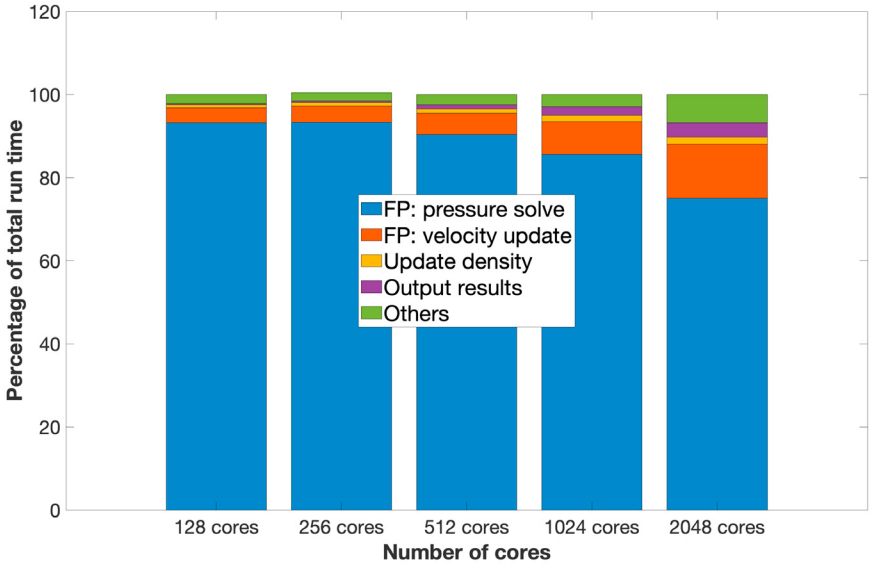


Fig. 4: Distribution of the computational time spent in various blocks of the algorithm for the solution of problem (2), flow over a 3D hill test case as a function of number of cores.

The strategy presented in [7] is already available in the `deal.II` library, but it was not considered in our previous work in order to avoid aliasing errors, as mentioned above. In this work we show the impact of this strategy on the numerical results and on the parallel performance. First, we analyze the impact of the use of the fast evaluation of block-diagonal operators in terms of accuracy. We consider a uniform mesh composed of $N_{el} = 60 \times 40 \times 16 = 38400$ elements, i.e. a resolution of 250 m with a final time $T_f = 1$ h. An excellent agreement is established between the two numerical strategies to solve the linear systems (Figure 5). A computational time saving of around the 20%

is established using the fast evaluation of the block diagonal operators. Hence, this first test suggests that the fast evaluation of block-diagonal operators provides similar results in terms of accuracy in spite of the aliasing error. We aim to further investigate this matter in future work.

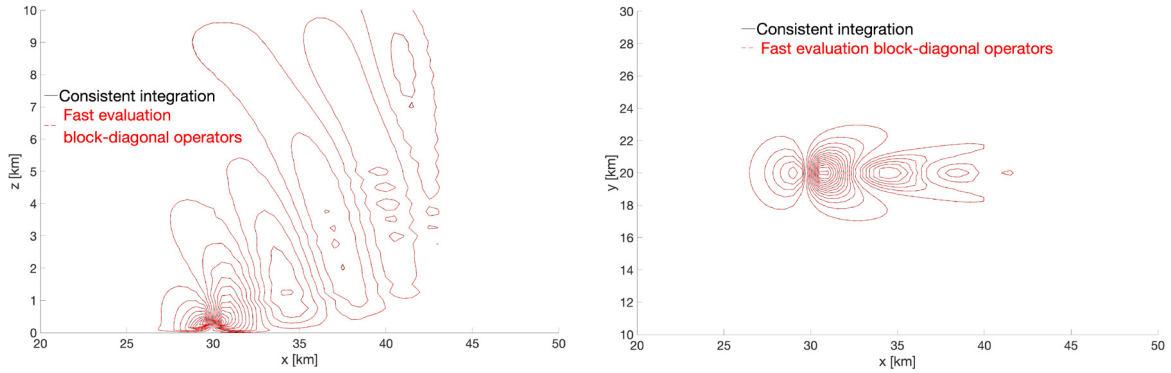


Fig. 5: Flow over 3D hill test case, computed vertical velocity at $T_f = 1$ h. Left: $x-z$ slice at $y = 20$ km. Right: $x-y$ slice at $z = 20$ km. Continuous black lines show the results with the consistent integration, whereas dashed red lines report the results with the fast evaluation of block-diagonal operators.

Finally, we show the impact of the fast evaluation of the block-diagonal operators on the parallel performance. One can easily notice how the computational time spent in computing the velocity field is significantly reduced (orange bars in Figure 6). More specifically, the percentage with respect to the run time is reduced to around 0.52% with 2048 cores. The strong scaling (Figure 7, top-left) when using the fast evaluation of the block-diagonal operators (red line) is apparently worse than when using consistent integration (black line) because the evaluation of the block-diagonal operators is so efficient that a small amount of time is required even for a relatively small number of cores (Figure 7, bottom-left). This consideration is further confirmed by the sizeable advantage in time-to-solution - 53% with 128 cores, about 47% with 2048 cores - brought by the fast evaluation of the block-diagonal operators (Figure 7, top-right). Moreover, one can easily notice that most of the computational time is required to solve (10) to compute the the pressure field (Figure 7, bottom-right).

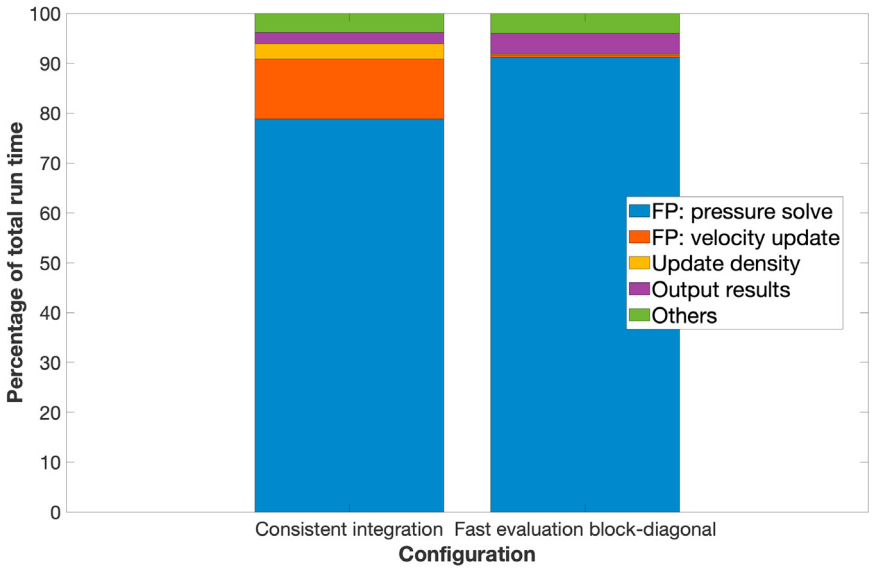


Fig. 6: Distribution of the computational time spent in various blocks of the algorithm for the solution of problem (2), flow over a 3D hill test case with 2048 cores. Left: consistent numerical integration. Right: fast evaluation of block-diagonal operators.

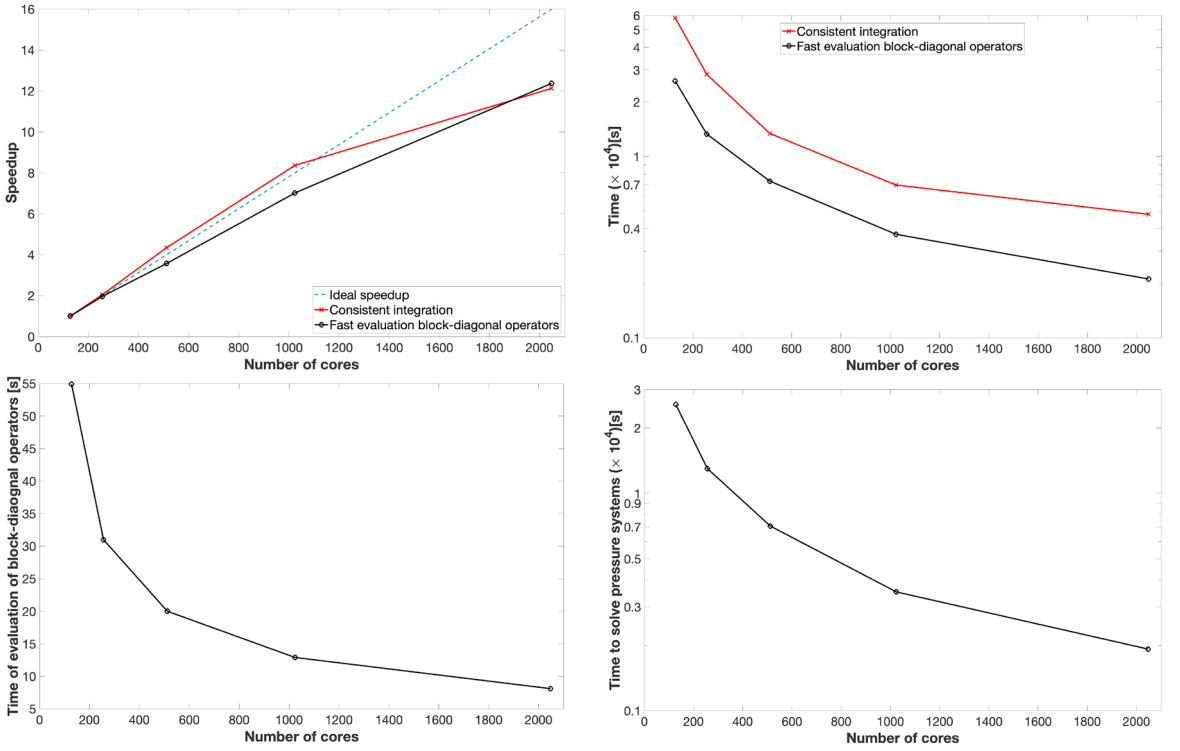


Fig. 7: Parallel performance of the solver for problem (2), flow over a 3D hill test case with fast evaluation of block-diagonal operators. Top-left: strong scaling. Top-right: wall-clock times. Bottom-left: wall-clock times of evaluation of block-diagonal operators. Bottom-right: wall-clock times to solve linear systems for the pressure field (10). In the top panels, the black lines show the results with the fast evaluation of block-diagonal operators, while the red lines report the results obtained employing the consistent integration.

4. Conclusions

This paper has reported the recent advances in the parallel performance of the IMEX-DG model for atmospheric dynamics simulations presented in [13, 14] and follows the analysis presented in [17]. First, we have shown the impact of the use of a suitable preconditioner, then the impact in the use of suitable execution flags and latest version of `deal.II`, and, finally, the use of efficient matrix-free techniques for block-diagonal operators [7].

In future work, we aim to further analyze the theoretical properties of the preconditioner, to further investigate the impact of the use of the direct inversion of the block-diagonal operator, and to exploit this technique to build a more suitable algebraic/geometric preconditioner since $(\mathbf{A}^{(l)})^{-1}$ can be readily evaluated. However, we point out that no theoretical result is available about the convergence properties of matrix-free multigrid preconditioners for non-symmetric linear systems arising from hyperbolic problems. Moreover, we plan to include more complex physical phenomena, in particular moist air, and to develop a three-dimensional dynamical core in spherical geometry including rotation so as to enable the testing on more realistic atmospheric flows. Finally, the data locality and high parallel efficiency of the DG method as well as the matrix-free approach which performs several computations on the fly makes the numerical method employed in this work particularly well-suited for a GPU implementation. Currently the `deal.II` library does not support discontinuous finite elements for GPU use. However, there is an ongoing effort of the `deal.II` community to provide a more complete and efficient GPU infrastructure including support for discontinuous finite elements. We aim to provide and test a GPU implementation once the library will be ready for this purpose.

Acknowledgements

We would like to thank the two anonymous reviewers for their constructive suggestions and remarks, which have contributed to improving the manuscript. The simulations have been run thanks to the computational resources made available through the EuroHPC JU Benchmark And Development project EHPC-DEV-2024D10-054. We thank the Application Support EPICURE Team and in particular W.A. Mainassara for the support and help. This work has been partly supported by the ESCAPE-2 project, European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 800897).

References

[1] P. Africa, D. Arndt, W. Bangerth, B. Blais, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, S. Kinnewig, M. Kronbichler, J. Thiele, B. Turcksin, and V. Yushutin. The deal.II library, Version 9.6. *Journal of Numerical Mathematics*, 32:369–380, 2024.

[2] D. Arndt, W. Bangerth, B. M., M. Feder, M. Fehling, J. Heinz, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, B. Turcksin, D. Wells, and S. Zampini. The deal.II library, Version 9.5. *Journal of Numerical Mathematics*, 31:231–246, 2023.

[3] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II: a general-purpose object-oriented finite element library. *ACM Transactions on Mathematical Software*, 33:24–51, 2007.

[4] T. Benacchio, W. O'Neill, and R. Klein. A blended soundproof-to-compressible numerical model for small-to mesoscale atmospheric dynamics. *Monthly Weather Review*, 142:4416–4438, 2014.

[5] F. Giraldo. *An Introduction to Element-Based Galerkin Methods on Tensor-Product Bases.* Springer Nature, 2020.

[6] C. Kennedy and M. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44:139–181, 2003.

[7] M. Kronbichler and K. Kormann. Fast matrix-free evaluation of discontinuous Galerkin finite element operators. *ACM Transactions on Mathematical Software (TOMS)*, 45:1–40, 2019.

[8] M. Kronbichler, S. Schoeder, C. Müller, and W. Wall. Comparison of implicit and explicit hybridizable discontinuous Galerkin methods for the acoustic wave equation. *International Journal for Numerical Methods in Engineering*, 106:712–739, 2016.

[9] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards*, 45:255–282, 1950.

[10] H. Lean, N. Theeuwes, M. Baldauf, J. Barkmeijer, G. Bessardon, L. Blunn, J. Bojarova, I. Boutle, P. A. Clark, M. Demuzere, et al. The hectometric modelling challenge: Gaps in the current state of the art and ways forward towards the implementation of 100-m scale weather and climate models. *Quarterly Journal of the Royal Meteorological Society*, 150:4671–4708, 2024.

[11] T. Melvin, T. Benacchio, B. Shipway, N. Wood, J. Thuburn, and C. Cotter. A mixed finite-element, finite-volume, semi-implicit discretization for atmospheric dynamics: Cartesian geometry. *Quarterly Journal of the Royal Meteorological Society*, 145:2835–2853, 2019.

[12] G. Orlando and L. Bonaventura. Asymptotic-preserving IMEX schemes for the Euler equations of non-ideal gases. *Journal of Computational Physics*, 529:113889, 2025.

[13] G. Orlando, P. Barbante, and L. Bonaventura. An efficient IMEX-DG solver for the compressible Navier-Stokes equations for non-ideal gases. *Journal of Computational Physics*, 471:111653, 2022.

[14] G. Orlando, T. Benacchio, and L. Bonaventura. An IMEX-DG solver for atmospheric dynamics simulations with adaptive mesh refinement. *Journal of Computational and Applied Mathematics*, 427:115124, 2023.

[15] G. Orlando, T. Benacchio, and L. Bonaventura. Robust and accurate simulations of flows over orography using non-conforming meshes. *Quarterly Journal of the Royal Meteorological Society*, 150:4750–4770, 2024.

[16] G. Orlando, T. Benacchio, and L. Bonaventura. Impact of curved elements for flows over orography with a Discontinuous Galerkin scheme. *Journal of Computational Physics*, 519:113445, 2024.

[17] G. Orlando, T. Benacchio, and L. Bonaventura. Efficient and scalable atmospheric dynamics simulations using non-conforming meshes. *Procedia Computer Science*, 255:33–42, 2025. Proceedings of the Second EuroHPC user day.

[18] G. Orlando, S. Boscarino, and G. Russo. A quantitative comparison of high-order asymptotic-preserving and asymptotically-accurate IMEX methods for the Euler equations with non-ideal gases, 2025.

[19] L. Pareschi and G. Russo. Implicit-explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxation. *Journal of Scientific computing*, 25:129–155, 2005.

[20] J. Steppeler, R. Hess, G. Doms, U. Schättler, and L. Bonaventura. Review of numerical methods for nonhydrostatic weather prediction models. *Meteorology and Atmospheric Physics*, 82:287–301, 2003.

[21] R. Varga. *Matrix Iterative Analysis*. Springer Science & Business Media, 2009.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 237–245

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Towards efficient high-resolution simulations of dust storm formation over large areas in North Africa using HPC

Samira Karbasi*, Jose A. G. Orza

SCOLAb, Dept. of Applied Physics, Universidad Miguel Hernandez de Elche, Avda. de la Universidad, Elche 03202, Spain

Abstract

This study presents the results of scalability tests of the WRF-Chem model on the MareNostrum 5 GPP supercomputing system, aimed at enabling high-resolution simulations of dust storm formation across large regions of North Africa. Simulations were conducted with three nested domains, reaching a 2 km horizontal resolution in the innermost domain. Two large areas of different size were evaluated. For the smaller domain, the optimal wall-clock time was achieved using approximately 2,000 cores, resulting in a simulation-to-wall-clock time ratio of 2:1. However, parallel efficiency declined significantly beyond ~1,200 cores, indicating limited scalability at higher core counts. In contrast, simulations over the larger domain maintained high efficiency between 1,680 and 2,016 cores, even outperforming the 2,500-core configuration in normalized cost. These findings highlight the importance of balancing runtime and resource efficiency. Future work should investigate which components of WRF-Chem dominate computational cost to better exploit HPC resources and guide model optimization

Keywords: WRF-Chem; dust storms; scalability test; HPC

1. Introduction

Desert dust aerosols, created by wind erosion on arid and semi-arid surfaces worldwide, can reduce visibility to near zero in the source regions and are regularly transported thousands of kilometers before being deposited on land or in ocean waters. While suspended in the atmosphere, dust aerosols impact air quality and human health, affect ecosystems, and play a significant role in Earth's radiative balance and atmospheric processes [1]. They influence the climate system through the scattering and absorption of solar and terrestrial radiation and by acting as cloud condensation and ice nuclei, which affect cloud formation and precipitation. In addition, dust deposition contributes to biogeochemical cycles by supplying essential nutrients to both terrestrial and marine environments. Dust contributes more than two-thirds of the global aerosol mass in the atmosphere [2].

10.1016/j.procs.2025.08.250

Surface soil susceptibility to erosion and wind friction velocity govern dust deflation. Consequently, dust sources are unevenly distributed across the globe. Over the past two decades, significant progress has been made in characterizing the impacts of dust outbreaks across different world regions. Studies have examined their connections to source dynamics, emission mechanisms, and diverse transport pathways. North African dust emissions are primarily carried westward to the tropical North Atlantic within the Saharan Air Layer, yet a considerable fraction is also transported northward into the Mediterranean basin and Europe (e.g. [3, 4, 5, 6, 7, 8, 9]).

The main meteorological features leading to dust emission have been identified in recent decades. These include low-level jets, synoptic-scale circulations, convective systems and mountain-induced winds (e.g. [10, 11, 12, 13, 14, 15]). Those winds are typically embedded within broader-scale atmospheric circulations such as African Easterly Waves, extratropical troughs, or subtropical highs, which modulate their frequency and intensity. Large-scale atmospheric instabilities at the upper troposphere organize the environment and are the precursors of the instabilities that cascade down to localized convective and orographic features at lower-levels (e.g. [16, 17]), so dust storm formation involve a complex sequence of multi-scale meteorological interactions.

Mountains and elevated terrain play a crucial role in shaping local and regional wind systems, many of which can contribute significantly to dust deflation and uplift. The interplay between complex terrain and atmospheric flow results in the formation of diverse orographic wind systems. Despite their recognized importance, mesoscale mountain-induced winds remain underrepresented in many dust emission and transport models. Their transient and localized nature poses challenges for both observation and numerical simulation, especially in regions with sparse in-situ monitoring. Advancing high-resolution modeling and incorporating satellite-derived topographic and atmospheric data, as well as surface observations, are essential steps to accurately capture these processes.

The goal of this work is to present the scalability tests conducted on the MareNostrum 5 GPP system to enable large-area, high-resolution simulations of dust storm formation across the Sahara Desert, with a focus on a selected case study. There are prominent mountain ranges and elevated plateaus in the Sahara, such as the Hoggar, Tibesti and Aïr Mountains. In addition, ranges in the edge of the Sahara like the Atlas Mountains are relevant for the emission and transport of African dust. These orographic features give rise to a variety of terrain-influenced wind systems, which significantly contributed to dust uplift across large areas of North Africa and facilitated long-range transport to Europe during two consecutive major dust storm events in March 2022.

2. High-resolution WRF-Chem simulations

2.1. Model description

The Weather Research and Forecasting (WRF) model is a regional numerical weather prediction system designed for both atmospheric research and operational forecasting applications. Its modular software architecture supports parallel computing and provides system extensibility for diverse atmospheric modeling needs. WRF-Chem, a comprehensive and online version of WRF coupled with atmospheric chemistry, enables the simultaneous simulation of weather and atmospheric chemistry processes, including the transport, transformation, and removal of trace gases and aerosols. The model supports two-way interactions, enabling atmospheric constituents to directly influence meteorological fields and vice versa. Due to the integration of chemistry and feedback mechanisms, WRF-Chem is significantly more computationally expensive than the standard WRF model. It was introduced by Grell et al. in 2005 [18].

The model supports nested grid capabilities and is fully parallelized using domain decomposition with Message Passing Interface (MPI), allowing efficient execution on HPC systems. Skamarock et al. [19] is the primary reference for WRF's numerical core, including time integration (third-order Runge-Kutta), grid staggering and nested grid capabilities. Grell et al. [18] introduced the WRF-Chem model and presented the chemistry-dynamics coupling and operator splitting. The use of mass conserving advection and other numerical features of WRF-Chem is described in [20].

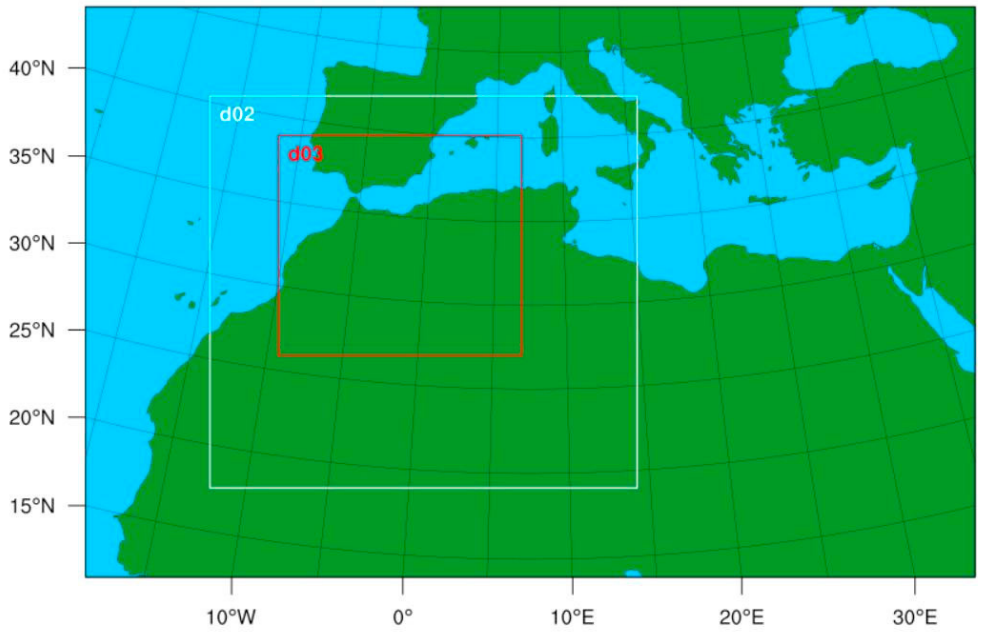


Fig. 1. WRF-Chem simulation domains, labeled 1, 2, and 3, with corresponding horizontal grid resolutions of 18, 6, and 2 km, respectively.

2.2. Model configuration

In this study, the WRF-Chem model (version 4.5.2) was configured with three nested domains at horizontal resolution of 18, 6 and 2 km (Figure 1), and 38 vertical sigma levels extending from the surface up 50 hPa in the stratosphere. The outermost domain spans from 10° to 45° N and 17°W to 35° E, encompassing North Africa, the Mediterranean, southern Europe, and part of the eastern North Atlantic. The inner domains used in our initial simulations cover different regions of the Sahara Desert, which include complex terrain features like the Atlas Mountains and others like the Hoggar, Tibesti and Aïr Mountains in order to capture orographically driven wind systems and associated dust uplift. For the scalability tests we considered two different configurations of the innermost (third) domain: one covering the Atlas region, parts of the Sahara near the Atlas Mountains, and the southern Iberian Peninsula; and another, larger domain encompassing the Hoggar, Tibesti, and Aïr regions.

The simulations ran for a 13-day period, from 00UTC on March 4 to 00UTC on March 17, 2022. A 10-day spin-up phase was included to allow for the model's dynamical and chemical fields (and particularly for aerosol concentrations) to stabilize before the analysis period of March 14-18. We note that long spin-up periods are important in aerosol simulations to allow for the buildup and proper distribution of emissions and chemical species within the model domain. Details of the model configuration and numerical simulation settings are presented in Table 1. The model was configured with micro-physics option by the Thompson scheme [21], the Mellor-Yamada-Janjic (MYJ) planetary boundary layer (PBL) scheme, and the RRTMG scheme from the Rapid Radiative Transfer Model (RRTM) for general circulation models (GCMs) to simulate the long-wave radiations [22, 23, 24], and the Dudhia scheme for short-wave radiation [25]. The GOCART (Goddard Chemistry Aerosol Radiation and Transport) dust emission scheme [26] was used to simulate dust emissions using wind-driven threshold friction velocity, surface properties, and soil moisture. It includes emission, transport, and deposition processes to represent the dynamics of dust aerosol.

WRF-Chem operates in parallel using domain decomposition, a method in which the spatial domain of the model is split into rectangular subdomains (tiles), one per MPI process or task. Each MPI process handles physics, dynamics, and chemistry computations of its assigned subdomain and includes halo or buffer zones that store the boundary information of neighboring tiles to maintain consistency across subdomain boundaries. At each timestep,

boundary data is exchanged between neighboring processes. Although WRF-Chem supports hybrid parallelism using OpenMP in addition to MPI, which can improve performance on multi-core nodes, only pure MPI parallelism was used in this work. The number of MPI tasks was varied in the scalability tests.

To increase the memory available for each MPI process, 6 cores per task were allocated. This is necessary because memory is physically limited to 2GB per core on MareNostrum 5. Therefore, to provide more memory to a single MPI process, the cpus-per-task parameter in the SLURM job script must be increased. In this case, the application requires 12GB per process (then cpus-per-task was set at 6), particularly when using a smaller number of nodes. This setting is maintained even in runs with larger node counts to maintain consistency in process distribution and memory availability.

2.3. Other data used in the study

In addition to provide initial and boundary conditions for the simulations, meteorological ERA5 reanalysis data [27], produced by the European Centre for Medium-Range Weather Forecasts (ECMWF), were used for the large-scale analysis of the dust storm events. ERA5 provides hourly atmospheric data at approximately 0.25° spatial resolution, available through the Copernicus Climate Data Store (https://cds.climate.copernicus.eu/).

Table 1. Overview of WRF model configuration.

Schemes	Option number	Variable namelist	WRF_Chem Option
Microphysics	8	mp_physics	Thompson scheme (Thompson, Field, Rasmussen and Hall, 2008)
Radiation	1	ra_lw_physics	RRTM (Mlawer et al., 1997)
	1	ra_sw_physics	Dudhia scheme (Dudhia, 1989)
Planetary boundary layer	2	bl_pbl_physics	Mellor-Yamada-Janjic (Janjic, 1994)
Land surface	2	sf_surface_physics	Noah Land Surface Model (Chen and Dudhia, 2001)
Surface model	2	sf_sfclay_physics	Monin-Obukhov (Janjic) scheme (Monin and Obukhov, 1954)
Cumulus parameterization	2	cu_physics	Betts-Miller-Janjic scheme
Dust_module	300	chem_opt	GOCART (Chin et al., 2002)
Meteorology Initial and boundary conditions	ERA5 (0.25^0)		
Chemical Initial and boundary conditions	EDGAR v5.0 emissions inventory speciated for the MOZART chemical mechanism (0.1° × 0.1°) (https://edgar.jrc.ec.europa.eu/) WACCM (https://www.acom.ucar.edu/waccm/download.shtml)		
Time period	20122_03-14_00:00 to 2022-03-19_00:00		
Spatial resolution	d01:18 km, d02: 6km, d03: 3km		

The Dust RGB product from the Meteosat Second Generation (MSG) satellites was employed to monitor the evolution of the dust plumes over North Africa and the Iberian Peninsula. It is a composite based on the combination of three infrared channels from the Spinning Enhanced Visible and Infrared Imager (SEVIRI) instrument [28]. While this product has limited spatial resolution, its high temporal resolution offers insight into the rapid evolution of the dust plumes.

SYNOP and METAR surface observations, including meteorological data and reports of visibility reduction due to airborne dust, were used to validate the WRF-Chem simulations. They are available in the Integrated Surface Database (https://www.ncei.noaa.gov/).

3. Results

3.1. Model performance and scalability tests

After a number of performance tests on the smaller study domain (corresponding to the one in Figure 1, see also Table 2), we selected a configuration of ntasks=420 and cpus-per-task=6. With this setup, each 5-day simulation

period required slightly less than 72 hours (3 days) of wall-clock time. To perform the full 13-day simulation we used the model's restart capability, sequentially running in 5-day segments and continuing the simulation from the generated restart files. The batch script used for these runs is provided in Figure 2.

The scalability tests were performed using simulations limited to 72 wall-clock hours and then extrapolated to a reference runtime of 10 simulation days. This approach avoided the need to manage restart files while still providing performance estimates which are not far to the required operational runs. The scalability tests were conducted by varying the number of tasks while keeping the number of CPUs per task constant (cpus-per-task=6). The total number of cores used is the product of these two parameters. As expected, wall-clock time decreased with an increasing number of cores (see Figure 3), reaching a minimum approximately 2,000 cores (ntasks=336). Beyond this point, performance slightly degraded, with wall-clock time increasing again when using 380 and 420 tasks. The most efficient configuration achieved a simulation time that was twice the wall-clock time, indicating a favorable balance between computational efficiency and resource usage.

```
#!/bin/bash
#SBATCH -J wrf_exe1
#SBATCH --ntasks 420
#SBATCH --cpus-per-task=6
#SBATCH -t 72:00:00
#SBATCH --qos=gp_resa
#SBATCH -o bt-wrf_chemresl-%j.out
#SBATCH -e bt-wrf_chemresl-%j.err
#SBATCH --constraint=highmem
#SBATCH -D .
#SBATCH --qos=gp_resa
#SBATCH -A umh82

module purge && module load oneapi/2023.2 hdf5 pnetcdf netcdf

export SRUN_CPUS_PER_TASK=$SLURM_CPUS_PER_TASK
srun ./wrf.exe
```

Fig. 2. SLURM batch script used to run the WRF-Chem simulation on MareNostrum 5.

Due to resource limitations, the scalability analysis for the larger simulation domain was more restricted. For this larger domain, the wall-clock time continued to decrease with increasing number of cores as shown in Figure 3. It is worth noting that wall-clock times using ntasks = 420 were very similar for both the smaller and larger domains, highlighting the potential for further optimization in larger configurations.

A normalized performance metrics such as the Normalized Computational Cost (NCC), or cost per grid point, is defined as

$$NCC = \frac{T_{wall\_clock} \cdot NC}{N_{grid\_cells}}$$

where T_{wall_clock} is the wall-clock time (hours) of the simulation, NC is the number of CPU cores used, and N_{grid_cells} is the total number of horizontal grid points in the model domain(s). This metric facilitates a fair comparison between simulations with different domain sizes and resolutions by normalizing the computational effort per grid point. We consider the optimal configurations for the small and large study domains. These values are listed in Table 2. As expected, the total computational cost is higher for simulations covering larger areas due to the increased number of grid cells. However, they also exhibit higher normalized costs reflecting decreased parallel efficiency and scalability at larger scales. Conversely, increasing spatial resolution (smaller grid spacing) raises the total computational cost, but tends to lower the normalized cost, indicating a more efficient use of resources per grid point, likely due to improved load balancing and reduced communication overhead relative to the computational workload.

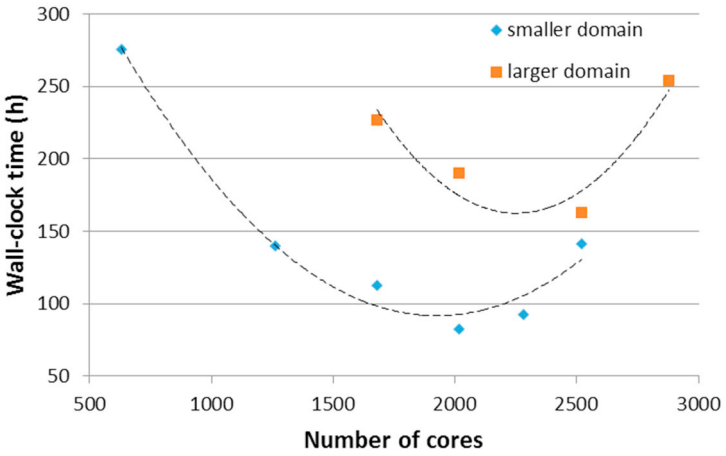


Fig. 3. Scalability plots for two simulation domains. Dashed lines are quadratic fits.

Table 2. Number of grid points and resolution for the nested domains in the two simulation domains considered in this study. The last row shows the normalized computational cost (NCC) for the optimal configurations of the two domains, according to Fig. 3.

	Smaller domain (Atlas Mnts.)			Larger domain (Central Sahara)		
	d01	d02	d03	d01	d02	d03
x_grid_points	330	475	811	427	661	1306
y_grid_points	212	436	733	301	592	1015
z_grid_points	38	38	38	38	38	38
Horiz. resol.	18 km	6 km	2 km	18 km	6 km	2 km
NCC	2,38	0,80	0,28	5,86	1,98	0,69

We also analyzed parallel efficiency and strong scaling behavior by comparing performance across core counts for two fixed domain sizes. These complementary metrics help to identify bottlenecks such as I/O limitations or inter-process communication costs that may affect scalability at high core counts. This metric is defined as

$$Parallel\ Efficiency = \frac{T_{wall\_clock\_ref} \cdot NC_{ref}}{T_{wall\_clock} \cdot NC}$$

where $T_{wall\_clock\_ref}$ is the wall-clock time at the reference simulation, NC_{ref} is the number of CPU cores used in the reference simulation, T_{wall_clock} is the wall-clock time for other simulations where the number of CPU cores is NC. It shows the efficiency degradation beyond 2016 cores (in the small system domain) and 2520 cores (in the large system domain). In the small system, efficiency drops significantly as the number of cores decreases beyond 2016, with only 46% efficient than the selected reference performance at 2520 cores. Best efficiency is around 1000 cores, suggesting that scaling beyond 2000 cores is inefficient due to communication overheads and underutilized cores, which in turn suggests the system size is small relative to the number of processors working in parallel. In turn, for the large system efficiency is higher than one for fewer cores, suggesting that the reference case does not fully utilize the resources or suffers from parallel overheads and that scaling is more balanced at lower core counts (1680-2016), but increasing to approximately 2800 reduces also the efficiency to 56%.

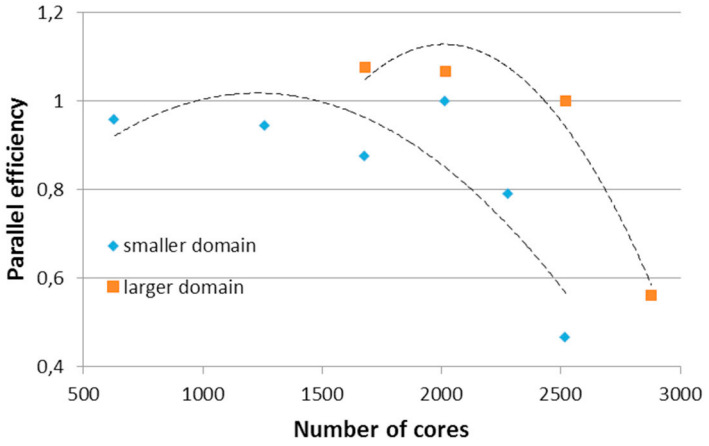


Fig. 4. Parallel efficiency plots for two simulation domains. Dashed lines are quadratic fits.

3.2. Case study – Simulation results

On March 14 and 15, 2022, two major dust storms formed over North Africa, subsequently impacting the Iberian Peninsula and much of western Europe. These events were preceded by upper-level dynamic disturbances, including Rossby wave breaking, located to the east and west of North Africa (details will be provided elsewhere). These features induced the penetration of strong and cold northerly/northeasterly winds into northeastern Africa and the advection of cold and high potential vorticity air into northwest Africa. A west-to-east pressure gradient and a cyclonic vortex over northwestern Africa drove easterly winds across the continent, interacting with the region's complex terrain to generate barrier flows and enhance near-surface wind speeds. The first dust storm formed early on March 14 over the Atlas Mountains, associated with a sub-synoptic-scale lee cyclone that triggered intense dust uplift, while the broader synoptic cyclonic circulation facilitated the northward transport to the Iberian Peninsula. The second dust storm, on March 15, was driven by low-level convergence between the easterly flow and the southwesterlies tied to the cyclonic system, resulting in strong northward dust advection and heavy dust loading over Iberia. Snaphots of the spatial structure of the dust and wind fields simulated by WRF-Chem is illustrated in Figures 5 and 6, offering a general overview of the model output. A more detailed understanding of the underlying processes typically requires higher resolution maps and additional analysis using vertical cross sections.

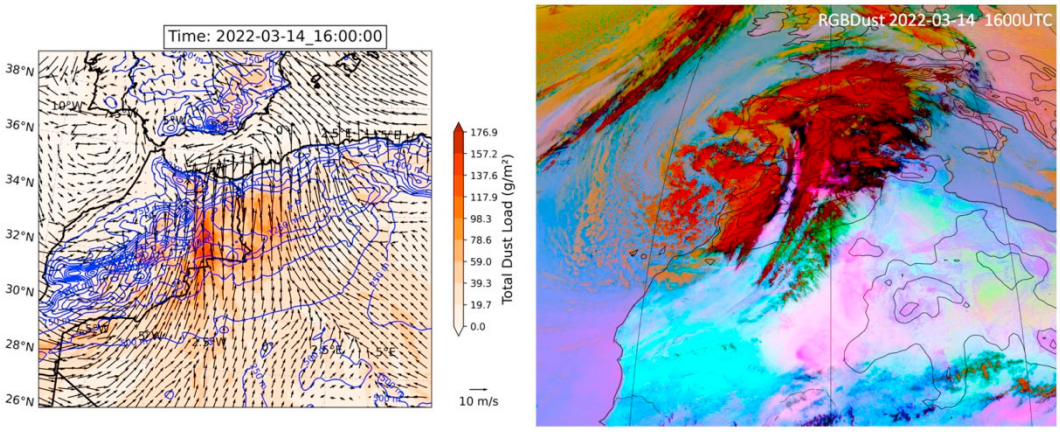


Fig. 5. First dust storm. Left: Modeled wind vectors and dust load for domain d03, with blue contours representing terrain elevation. Right: MSG Dust RGB composite imagery, with dust depicted in pink/magenta tones and thick high-altitude clouds appearing in brown.

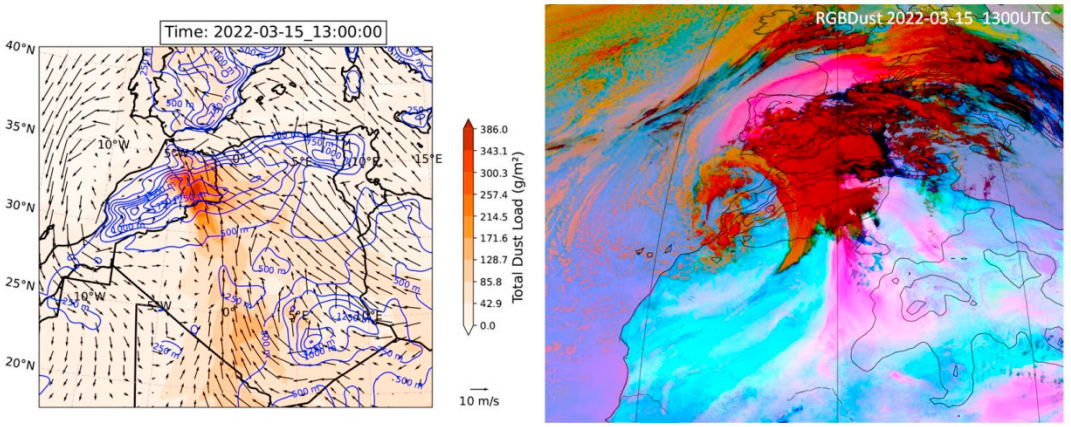


Fig. 6. Same as Figure 5, but for the second dust storm one day later.

4. Conclusions

The WRF-Chem model demonstrated good scalability on the MareNostrum 5 GPP system for high-resolution simulations of dust storm formation over relatively large geographic areas, using a 2 km horizontal resolution in the innermost domain. For the smaller domain, although the configuration with the shortest wall-clock time was observed with approximately 2,000 cores and resulted in a simulation time of twice the wall-clock time, parallel efficiency degraded significantly beyond about 1,200 cores. This suggests that over-parallelization leads to underutilized resources, and configurations using fewer cores (around 1,000) offer better efficiency for small-domain simulations. In contrast, for the larger domain, scaling seems to level off at 2500 cores and parallel efficiency remained high when using 1,680-2,016 cores, even outperforming the reference configuration of 2,520 cores in normalized efficiency. This indicates that optimal performance can be achieved with fewer cores, improving resource utilization without scarifying runtime significantly.

These results emphasize the importance of balancing runtime and computational efficiency, encouraging future work on high-resolution simulations over large domains, where core counts can be scaled more efficiently. While this study did not include detailed code profiling due to the constraints of the benchmarking period, future work should be done to profile WRF-Chem simulations to identify the most computationally intensive components of WRF-Chem (likely chemistry, I/O) that limit scalability.

Acknowledgements

We thank two anonymous reviewers for their constructive comments and suggestions, which improved the quality and clarity of this manuscript. This work was partially funded by the Spanish Government MCIN/AEI/10.13039/501100011033 under Grant PID2020-115153RB-I00 (rROSSETA Project). We acknowledge EuroHPC JU for providing access to the MareNostrum5 supercomputing facility through EHPC-BEN-2024B07-017 and thank the BSC support team for their valuable assistance.

References

[1] Boucher, O. (2015) “Atmospheric aerosols: properties and climate impacts” Springer, ISBN: 9789401778008.

[2] Kinne, S., et al. (2006) “An AeroCom initial assessment – optical properties in aerosol component modules of global models” *Atmospheric Chemistry and Physics* **6**:1815–1834.

[3] Moulin, C., et al. (1998). “Satellite climatology of African dust transport in the Mediterranean atmosphere” *Journal of Geophysical Research*, **103** (**D11**): 13137–13144.

[4] Gkikas, A., Hatzianastassiou, N., and Mihalopoulos, N. (2009). “Aerosol events in the broader Mediterranean basin based on 7-year (2000–2007) MODIS C005 data” *Annales Geophysicae*, **27**, 3509–3522.

[5] Israelevich, P., Ganor, E., Alpert, P., Kishcha, P., & Stupp, A. (2012) "Predominant transport paths of Saharan dust over the Mediterranean Sea to Europe" *Journal of Geophysical Research* **117**, D02205.

[6] Querol, X., et al. (2009) "African dust contributions to mean ambient PM10 mass-levels across the Mediterranean Basin" *Atmospheric Environment* **43**: 4266–4277.

[7] Pey, J., Querol, X., Alastuey, A., Forastiere, F., and Stafoggia, M. (2013) "African dust outbreaks over the Mediterranean Basin during 2001–2011: PM10 concentrations, phenomenology and trends, and its relation with synoptic and mesoscale meteorology" *Atmospheric Chemistry and Physics* **13**: 1395–1410.

[8] Varga, G., Újvári, G., and Kovács, J. (2014) "Spatiotemporal patterns of Saharan dust outbreaks in the Mediterranean Basin" *Aeolian Research* **15**: 151–160.

[9] Marinou, E., et al. (2017). "Three-dimensional evolution of Saharan dust transport towards Europe based on a 9-year EARLINET-optimized CALIPSO dataset" *Atmospheric Chemistry and Physics* **17**: 5893–5919.

[10] Knippertz, P., and Todd, M.C. (2012) "Mineral dust aerosols over the Sahara: meteorological controls on emission and transport and implications for modeling" *Reviews of Geophysics* **50**, RG1007.

[11] Schepanski, K., Tegen, I., Laurent, B., Heinold, B., and Macke, A. (2007) "A new Saharan dust source activation frequency map derived from MSG-SEVIRI IR channels" *Geophysical Research Letters* **34**, L18803.

[12] Schepanski, K., et al. (2009) "Meteorological processes forcing Saharan dust emission inferred from MSG-SEVIRI observations of sub-daily dust source activation" *Journal of Geophysical Research* **114**, D10201.

[13] Fiedler, S., Schepanski, K., Knippertz, P., Heinold, B., and Tegen, I. (2014) "How important are atmospheric depressions and mobile cyclones for emitting mineral dust aerosol in North Africa?" *Atmospheric Chemistry and Physics* **14** (**17**): 8983–9000.

[14] Fiedler, S., Kaplan, M.L., and Knippertz, P. (2015) "The importance of Harmattan surges for the emission of North African dust aerosol" *Geophysical Research Letters* **42**: 9495–9504.

[15] Pokharel, A.K., Kaplan, M.L., and Fiedler, S. (2017) "Subtropical Dust Storms and Downslope Wind Events" *Journal of Geophysical Research: Atmospheres* **122**: 10191–10205.

[16] Orza, J.A.G., S. Dhital, S. Fiedler, and M.L. Kaplan (2020) "Large Scale Upper-level Precursors for Dust Storm Formation over North Africa and Poleward Transport to the Iberian Peninsula. Part I: An Observational Analysis" *Atmospheric Environment* **237**, 117688.

[17] Dhital, S., M.L. Kaplan, J.A.G. Orza, and S. Fiedler (2021) "Poleward Transport of African Dust to the Iberian Peninsula Organized by a Barrier Jet and Hydraulic Jumps: Observations and High-Resolution Simulation Analyses" *Atmospheric Environment* **261**, 118574.

[18] Grell, G.A., Steven E. Peckham, Rainer Schmitz, Stuart A. McKeen, Gregory Frost, William C. Skamarock, and Brian Eder (2005) "Fully coupled 'online' chemistry within the WRF model" *Atmospheric Environment* **39**: 6957-6975.

[19] Skamarock, W. C., et al. (2019) "A description of the Advanced Research WRF Version 4". NCAR Technical Note NCAR/TN-556+STR. https://doi.org/10.5065/1dfh-6p97

[20] Fast, J. D., et al. (2006) "Evolution of ozone, particulates, and aerosol direct radiative forcing in the vicinity of Houston using a fully coupled meteorology–chemistry model" *Journal of Geophysical Research*, **111** D21305.

[21] Thompson, G., Rasmussen, R. M., and Manning, K. (2004) "Explicit forecasts of winter precipitation using an improved bulk microphysics scheme. Part I: Description and sensitivity analysis" *Monthly Weather Review* **132**: 519–542.

[22] Mlawer, E. J., Taubman, S. J., Brown, P. D., Iacono, M. J., and Clough, S. A. (1997) "Radiative transfer for inhomogeneous atmospheres: RRTM, a validated correlated-k model for the longwave." *Journal of Geophysical Research* **102** (**D14**): 16,663–16,682.

[23] Iacono, M. J., Mlawer, E. J., and Clough, S. A. (2000) "Impact of an improved longwave radiation model, RRTM, on the energy budget and thermodynamic properties of the NCAR community climate model, CCM3." *Journal of Geophysical Research* **105** (**D11**): 14873–14890.

[24] Zhao, J., Ma, X., Wu, S., and Sha, T. (2020) "Dust emission and transport in Northwest China: WRF-Chem simulation and comparisons with multi-sensor observations." *Atmospheric Research* **241**: 104978.

[25] Dudhia, J. (1989) "Numerical study of convection observed during the winter monsoon experiment using a mesoscale two-dimensional model" *Journal of Atmospheric Science* **46**: 3077–3107

[26] Chin, M., Ginoux, P., Kinne, S., Torres, O., Holben, B. N., Duncan, B. N., Martin, R. V., Logan, J. A., Higurashi, A., and Nakajima, T. (2002) "Tropospheric aerosol optical thickness from the GOCART model and comparisons with satellite and Sun photometer measurements." *Journal of Atmospheric Sciences* **59** (**3**): 461–483.

[27] Hersbach, H.B., et al. (2020) "The ERA5 Global Reanalysis" *Quarterly Journal of the Royal Meteorological Society* **146**: 1999-2049.

[28] Lensky, I.M., and Rosenfeld, D. (2008) "Clouds-aerosols-precipitation satellite analysis tool (CAPSAT)". *Atmospheric Chemistry and Physics* **8** (**22**): 6739-6753.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 246–255

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Scalable Detection of Environmental Events on EuroHPC MeluXina

Jini Cheriyan[a*], Jaime Santos[a*]

[a]*AiTecServ, Praça da Concórdia nº62, 2870-471 Afonsoeiro, Montijo, Portugal*

Abstract

This study explores the training and experimentation of AI-based environmental event detection models using large-scale datasets on the EuroHPC supercomputer Meluxina. The primary objective is to assess the scalability, performance, and efficiency of advanced deep learning-based image segmentation models for fire, smoke and water detection across diverse environmental conditions. By leveraging high-performance computing (HPC), we trained and evaluated multiple architectures to enhance detection accuracy, minimize training time, and improve model scalability. The experimental results demonstrate the advantages of HPC-driven deep learning in handling large datasets, accelerating model convergence, and enhancing detection precision. These advancements contribute to the development of efficient early warning systems, aiding in environmental protection and public safety.

Keywords: Fire-smoke Detection; Water detection; Image Segmentation; HPC; Meluxina; Deep Learning; Large-Scale Training

1. Introduction

Wildfires and other natural disasters have become increasingly frequent and intense due to climate change [1], posing significant threats to ecosystems, public safety, and infrastructure. Early detection and rapid response are crucial for minimizing environmental degradation, economic losses, and associated health risks.

Conventional fire and flood detection systems—such as satellite imaging, ground-based sensors, and human surveillance—have long played a vital role in environmental monitoring [2]. Satellite data offers broad coverage and, in many cases, near-real-time availability. However, satellite-based monitoring may still face limitations in spatial resolution for small or early-stage events, dependency on clear sky conditions in visible bands. These challenges

∗ Corresponding author.
E-mail addresses: jini.cheriyan@aitecserv.pt, jaime.santos@aitecserv.pt

1877-0509
10.1016/j.procs.2025.08.251

highlight the need for complementary ground-based systems capable of localized, continuous, and high-resolution observation.

Recent advancements in computer vision and deep learning have led to the emergence of AI-based systems that offer real-time monitoring through camera feeds and image segmentation. Looking at the big picture, segmentation is one of the high-level tasks that paves the way towards complete scene understanding [15]. These models are particularly suited for localized deployments (e.g., urban, forested, or industrial environments) where rapid detection of fire, smoke, or water anomalies is critical.

Nonetheless, real-world complexity continues to present challenges. Smoke can resemble clouds or mist; artificial lighting can create false positives; and water detection is sensitive to surface reflections and color variability. Additionally, environmental conditions, time of day, and object distance complicate feature extraction—especially for small-scale or distant events.

Training segmentation models on diverse, large-scale datasets is essential for reliable generalization. However, doing so demands significant computational power, often beyond the reach of standard infrastructure. To overcome these barriers, this study utilizes the EuroHPC Meluxina supercomputer [5], [6], which enables large-batch, high-resolution training on deep architectures using NVIDIA A100 GPUs.

To safeguard our proprietary technology, detailed specifications of the dataset and deep learning models are not disclosed. However, the core experimental setup and all reported metrics reflect realistic deployment scenarios, ensuring scientific and technical rigor.

The primary objectives of this research are:

- To train and experiment with state-of-the-art image segmentation models for fire and water detection using a large dataset.
- To assess the performance and scalability of different segmentation models/architecture on the Meluxina supercomputer.
- To analyze the computational benefits of using HPC resources for large-scale training and experimentation.

The remainder of this paper is organized as follows: Section 2 discusses the background and related work in fire, water detection and HPC-based training. Section 3 outlines the methodology, including dataset preparation, model selection, and HPC implementation on Meluxina. Section 4 presents the experimental results, evaluating model performance, scalability, and computational efficiency. Finally, Section 5 provides a discussion on the effectiveness, limitations, and potential future work.

2. Background and related work

2.1. Natural event detection techniques

Traditional environmental monitoring techniques have included satellite imagery, sensor networks, and manual surveillance [7]. Satellites remain indispensable for large-scale wildfire and flood monitoring, offering multi-spectral coverage and frequent revisits. Platforms such as MODIS, VIIRS, Sentinel-2 enable fire detection in visible, infrared, and thermal bands—often overcoming limitations like cloud interference [8]. However, satellite data may still face constraints in detecting small-scale or rapidly evolving local incidents with sufficient granularity and timeliness, especially in areas lacking frequent overpasses or with low-cost coverage constraints. Ground-based sensor networks provide localized and continuous monitoring but often require significant infrastructure investments and can be limited in spatial reach. Thermal cameras are also commonly used for fire detection; however, their effectiveness diminishes at long distances, as accuracy is highly dependent on temperature thresholds. Moreover, thermal imaging systems are often more expensive compared to alternative detection methods, making large-scale deployment less feasible. To complement these traditional approaches, recent developments in artificial intelligence (AI) have enabled real-time detection systems that process live image and video streams. Deep learning-based segmentation models, in particular, offer pixel-level classification that can distinguish between fire, smoke, water, and background across varied environments. Compared to thresholding or edge detection, convolutional neural networks (CNNs) have shown superior adaptability and accuracy in complex scenes [9], [10].

This study builds upon these advancements by evaluating segmentation model performance in an HPC setting, aiming to scale up training and experimentation across diverse datasets.

2.2. Computational Challenges in AI-Based event Detection

Natural event detection using AI requires processing high-resolution imagery and video feeds, posing significant computational challenges. Deep convolutional neural networks (CNNs) used in object detection and localization contain millions to hundreds of millions of parameters, demanding extensive GPU memory and processing power [11]. The key challenges include:

- **High Memory Demand**: Large batch sizes and deep architectures require substantial GPU memory, leading to bottlenecks in training and inference.
- **Training Efficiency**: Optimizing model convergence while balancing computational costs remains a key challenge, especially when dealing with large datasets.
- **Real-Time Processing Needs**: Natural event detection requires low-latency inference, which adds to the computational burden.
- **Scalability**: Training on large datasets necessitates efficient distribution of workloads across multiple processors.

These challenges necessitate the use of high-performance computing (HPC) infrastructure to accelerate model training and optimize inference speed [12].

2.3. Leveraging HPC for Large-Scale AI Model Training

High-Performance Computing (HPC) provides the computational power required to train complex deep learning models efficiently. By leveraging distributed computing and parallel processing, HPC systems like EuroHPC MeluXina address key limitations in AI-based fire detection, enabling:

- **Faster Training:** HPC accelerates deep learning by distributing workloads across multiple GPUs, significantly reducing training time.
- **Scalability & Distributed Computing:** Large datasets can be processed efficiently by splitting computations across multiple nodes.
- **Memory Optimization:** HPC platforms support deep learning models with high memory requirements, enabling the use of larger batch sizes.
- **Enhanced AI Capabilities:** While resource constraints still exist, HPC enables training of complex architectures with millions of parameters more efficiently than conventional systems.

Our study primarily utilized PyTorch, a widely adopted deep learning framework, due to its dynamic computation graph, ease of debugging, and native support for distributed training [13].

HPC has been widely used in deep learning applications such as medical imaging and climate modeling. However, its application to large-scale natural event detection remains underexplored. This study evaluates the effectiveness of using MeluXina for AI-based event detection on a dataset of several thousands of images, assessing its performance, scalability, and computational benefits.

2.4. Research Gap and Motivation

While HPC systems are increasingly used for deep learning, detailed studies on how they are leveraged for optimizing large-scale image segmentation models remain relatively scarce in the public domain, partly due to limited access to such infrastructures and the proprietary nature of many implementations. Traditional training methods often struggle with high computational demands, extended training times, and scalability limitations, making it challenging to experiment effectively with different model architectures, batch sizes, and other hyperparameters especially with single GPU and memory limitation of GPU [16].

This research addresses these challenges by leveraging the EuroHPC Meluxina supercomputer to train and experiment with fire and water detection models of varying complexity. By systematically evaluating scalability, accuracy, memory efficiency, and training speed, we aim to establish a computational framework for deploying, large-scale fire detection systems in environmental monitoring applications as also water detection systems to detect ruptures; margins transposition or imminent floods.

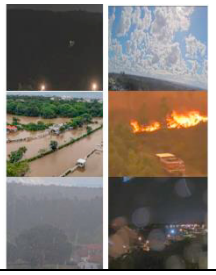
3. Methodology

3.1. Data Collection and Preparation

For this study, a large and diverse dataset was curated to facilitate robust training and evaluation of image segmentation models. The dataset was compiled by collecting publicly available images from various internet sources. Care was taken to ensure diversity in environmental conditions fire-smoke and flood scenarios to improve the generalization capability of the models. The dataset includes a diverse range of scenarios such as environmental landscapes (e.g., forests, agricultural areas, and cities), urban settings (including buildings and vehicles), industrial zones (like factories and infrastructure), indoor environments, and various non-hazardous backgrounds featuring fog, clouds, light reflections, or clear conditions without the presence of any critical event.

Table 1 provides a few representative examples from the training dataset used in this study. These images illustrate the diverse environmental conditions, lighting variations, and event types such as fire, smoke, water, and background. While the dataset covers multiple hazard categories, two separate models were trained: a multi-class segmentation model for detecting fire and smoke as distinct classes, and a binary segmentation model for identifying water presence versus background (i.e., water/no-water). The examples in the table are not exhaustive but reflect key visual challenges encountered in real-world scenarios, such as occlusions, low-contrast smoke, or reflections on water surfaces.

Table 1. Samples of training data

Image Sample	Description
	Night time scene and cloudy sky that resemble smoke
	Landscape with flood/water and active wildfire
	Forest scene in normal daylight with rain, Rainy urban night scene with reflections and lens flares

These annotated examples highlight the complexity of differentiating fire, smoke, and background elements, especially under variable lighting and atmospheric conditions.

3.2. Data Preprocessing

To prepare the dataset for model training, several steps were undertaken. Annotation: Each image was manually annotated to create precise pixel-wise segmentation masks of events like fire smoke and water regions and; Class Balance: To ensure reliable segmentation performance, the dataset was carefully curated to maintain a reasonable balance across all the object classes: background, smoke, fire, water**.** Specifically:

- A conscious effort was made to include a sufficient number of event category samples, as these classes tend to be underrepresented compared to background pixels. This strategy reduced the risk of false positives in real-world

deployment and helped the model learn finer distinctions between smoke and fire regions or between clear water and muddy water.

- The dataset used for training and evaluation was well-balanced, containing a comparable number of annotated examples for each class. These examples, referred to as instances, represent individual labeled regions within the images that correspond to distinct target categories (e.g., fire or smoke). This balance ensured that the model learned from all classes equally, reducing the risk of class imbalance and bias during training.
- To ensure consistency and optimize model performance, all input images were resized to a fixed resolution of 384x640 pixels during preprocessing. Although this may slightly reduce fine-grained spatial details in high-resolution images, it significantly accelerates training keeping sufficient visual fidelity for pattern detection.

3.3. Model Selection

For large-scale detection using image segmentation, deep learning models with varying levels of complexity were selected. These models differ in network depth, number of parameters, and computational efficiency, enabling a comparative analysis of segmentation accuracy, training speed, and hardware resource utilization.

3.3.1. High-Accuracy Segmentation Models with Multiple Variants (Models 1–4)

More complex segmentation models were used to assess the impact of network depth and computational cost on segmentation performance.

- Model 1 (Moderate, ~27M parameters): Balancing computational efficiency and segmentation accuracy.
- Model 2 (Moderate-High complexity, ~46M parameters): Capturing detailed feature representations with higher memory demands.
- Model 3 (High complexity, ~72M parameters): Deepest architecture, optimized for robust faint smoke detection, particularly at long distances.
- Model 4 (Very high complexity,>100M parameters): Designed with an advanced architecture comprising over 100 million parameters, this model is tailored for maximum segmentation precision in highly complex scenarios. Its increased depth and representational power make it particularly effective in challenging environments with fine-grained features, such as low-contrast smoke or partially obscured fire and evolution of water.

This approach enabled a comparative study of segmentation accuracy versus computational requirements, providing insights into scalability, memory efficiency, and performance trade-offs. By evaluating models with parameter sizes ranging from ~27M to ~72M, different training complexities, and inference speeds, this study aimed to determine the most computationally viable approach for large-scale event detection.

3.3.2. Model Training and Experimentation

The model training process followed a progressive and iterative approach, starting from limited data and computational resources and scaling up to large-scale experiments. This section presents the evolution of experimentation, the role of data scaling, and the impact of high-performance computing infrastructure on deep learning model development.

3.4. Initial Experimentation and Dataset Constraints

Our experimentation began with a relatively small dataset consisting of a few thousand images, each containing instances of fire, smoke and water under various conditions. However, early results revealed significant limitations - particularly in detecting small-scale fires, faint smoke, small amount of water and different colors of water in challenging environmental scenarios. The shallow architecture often struggled to generalize across diverse backgrounds, and segmentation performance remained suboptimal, especially for smoke detection.

To address the performance gap, we progressively expanded the dataset to several thousands of images, carefully curating balanced instances of events across diverse scenes (indoor, outdoor, urban, forested). This enriched dataset

significantly improved model learning and robustness. However, training deeper models on this large-scale dataset presented computational challenges, especially when aiming for larger batch sizes on standard hardware.

3.5. Leveraging EuroHPC Meluxina for Advanced Training

Access to the EuroHPC Meluxina supercomputing infrastructure played a pivotal role in overcoming traditional computational limitations, enabling the training of complex image segmentation models at scale. Meluxina's multi-GPU architecture, high memory bandwidth, and distributed processing capabilities allowed for:

- Efficient training of high-parameter models without memory bottlenecks,
- Exploration of larger batch sizes for performance benchmarking,
- Faster convergence across various model complexities.

This high-performance environment significantly accelerated our experimentation and facilitated robust performance improvements. Model development leveraged Meluxina's capabilities to execute extensive deep learning workflows while evaluating trade-offs between segmentation accuracy, memory utilization, and training efficiency.

3.6. Training Configuration

Table 2. Model configuration

Model ID	Complexity	Parameters (Millions)	Batch Size Feasibility
Model 1	Moderate	~27M	64-128
Model 2	Moderate-High	~46M	64-128
Model 3	High	~72M	64-128
Model 4	Very High	>100M	<16

3.7. Batch Size and Memory Considerations

Models 1–3 were successfully trained with batch sizes of 64 and 128 using 4 A100 GPUs. Model 3, despite being the most complex, remained within resource limits due to distributed training. Model 4 exceeded feasible memory thresholds and was not fully explored due to time-limited HPC access and its extreme memory demands.

Although advanced techniques such as gradient checkpointing and mixed-precision training were explored, they were not sufficient to enable successful training of the largest model within the current HPC resource constraints. These approaches, along with further optimization strategies, will be considered in future work on more powerful systems to address the demands of very high-complexity architectures. All models followed a consistent training workflow, including online data augmentation, hyperparameter tuning, and multi-GPU acceleration, with segmentation metrics and GPU usage tracked for performance evaluation.

.

4. Result and Discussions

When evaluating segmentation models for fire, smoke, and water localization, standard metrics such as mean Average Precision (mAP) and Intersection over Union (IoU) are commonly used [14]. In pixel-level tasks like segmentation, even slight annotation inconsistencies—such as faint smoke boundaries or partially visible fire—can lead to variations in performance scores. These minor discrepancies, particularly in ambiguous regions (e.g., smoke blending with clouds or background haze), may result in small metric fluctuations that do not necessarily reflect the model's actual qualitative performance.

4.1. Quantitative Evaluation

The models were evaluated using (mAP), a standard metric for object detection and segmentation accuracy. mAP(B) is the mean Average Precision for bounding box predictions and mAP(M)is defined as the mean Average

Precision for segmentation mask predictions. The mAP is calculated by finding Average Precision (AP) for each class and then average over a number of classes.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP\ i \qquad (1)$$

We report mAP at IoU = 0.50 (a lenient threshold) and the averaged mAP across IoU = 0.50–0.95 (in 0.05 increments), which offers a more robust evaluation of model performance. These metrics provide a comprehensive understanding of model performance in both localizing objects (bounding boxes) and segmenting their shapes (masks).

The table below summarizes their accuracy:

Table 3. Quantitative evaluation results

Model ID	mAP@50 (B)	mAP@50–95 (B)	mAP@50 (M)	mAP@50–95 (M)
Model 1	0.895	0.695	0.885	0.576
Model 2	0.905	0.713	0.890	0.592
Model 3	0.890	0.691	0.879	0.569

4.1.1. Accuracy Trend During Training

Model 1 (Moderate Complexity):

This model exhibited stable and gradual convergence, with both training and validation loss curves decreasing steadily throughout the training process. The smooth alignment between the two curves indicates consistent learning and minimal overfitting. However, it required 227 epochs to reach its best performance, making it the slowest converging model among the three. This highlights its reliability, albeit at the cost of extended training time.

Model 2 (Moderate-High Complexity):

Moderate-High Complexity achieved the highest mAP scores across all IoU thresholds. Its training curve showed a balanced and efficient convergence, with the validation loss closely mirroring the training loss, indicating strong generalization capability. The model reached peak performance at epoch 208, representing a good trade-off between accuracy and training duration, making it ideal for balanced resource-accuracy scenarios.

Model 3 (High Complexity):

Despite being the deepest model, converged the fastest, with optimal weights saved at epoch 125. While this indicates superior learning speed, the training process displayed slightly more fluctuation in validation loss, potentially due to sensitivity to data variance or minor overfitting. Nevertheless, the model maintained competitive mAP scores, showcasing its potential in time-critical deployments where faster convergence is preferred.

These results highlight how deeper and more complex models can achieve high accuracy with fewer training epochs but may require more careful tuning to prevent overfitting.

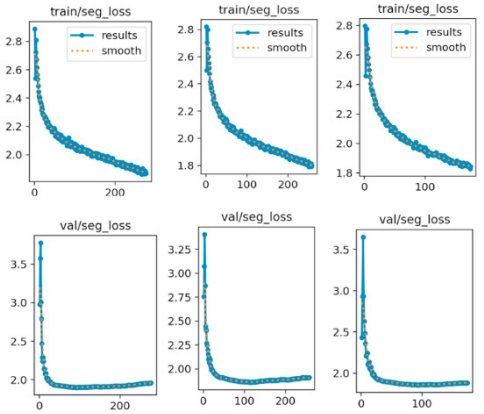


Fig. 1 Training and validation segmentation loss curves for all three models. Top row: training loss. Bottom row: validation loss.

Fig 1 provides detailed training graphs for each model, showing the trajectory of training and validation losses, showing the convergence behavior of each model confirm the quantitative analysis discussed in Section 4.1.

4.2 Qualitative Evaluation

To assess the real-world performance of the models beyond numerical metrics, qualitative evaluations were conducted on 60 videos from diverse environments and 2,000 real-world images. The primary objective was to determine how effectively each model detects and segments fire, smoke, and water regions under varying conditions, including:

- Different lighting conditions (day/night scenarios)
- Obstructed views (e.g., behind trees, structures, or smoke-covered regions)
- Long-range fire/smoke and water visibility
- Varying levels of fire intensity (small flame ignition to large-scale fire outbreaks)
- Challenging backgrounds such as fog, reflections, raindrops, and streetlights

Key Observations: Model 1 & 2 performed well in detecting flames but struggled to capture faint or distant smoke, particularly in noisy or obscured scenes. Model 3 demonstrated superior capability in detecting subtle smoke cues, even under low contrast, foggy, or nighttime conditions, while also significantly reducing false detections caused by non-fire elements like streetlights, reflections, and raindrops.

In categories like small smoke and faint smoke, Model 3 achieved the highest detection accuracy (80% and 99% respectively), outperforming the other models by a notable margin. Conversely, false positive rates in visually confusing scenarios were also lowest in Model 3—for instance, only 8% misclassification for streetlights and 20% for night lighting, compared to 10–28% in the other models.

Real-Time Performance: All models were tested in real-time deployment. Model 3 in particular provided timely and reliable alerts for early-stage events, making it the most practical candidate for real-world surveillance systems.

These findings emphasize the value of high-capacity models in early detection tasks. Although quantitative metrics like mAP showed only marginal differences, qualitative evaluations revealed clear distinctions in robustness, generalization, and real-world utility. This supports the use of deeper networks in critical applications, while acknowledging that their higher resource demands must be balanced against deployment constraints—particularly on edge devices with limited hardware.

4.3 Scalability and Resource Utilization Analysis

A critical aspect of this study was evaluating memory efficiency and scalability when training on huge number of images using HPC infrastructure. Scalability analysis showed that Models 1–3 scaled well on the HPC infrastructure, with resource demands increasing with complexity. Model 4 was not explored due to time-limited access to the HPC system and its significantly higher memory requirements.

Table 4. Scalability analysis results

Model ID	Memory Utilization (4GPUs)	Batch Size	Total GPU Hours	Node hours	Convergence speed
Model 1	~9 GB	64	~38	~9.5	Slowest
Model 2	~12 GB	64	~32	~8.0	Moderate
Model 3	~16 GB	64	~17	~4.25	Fastest
Model 4	>40 GB	<16	NA	NA	NA

Despite being the most complex, Model 3 achieved the fastest convergence, making it the most time-efficient in terms of total GPU hours. Model 4, requiring over 40 GB per GPU, could not be trained under standard conditions, underscoring the need for advanced memory optimization techniques for extremely deep models. Model 3 (the most complex trained model) required ~8 minutes per epoch on a single GPU, while multi-GPU training reduced this to ~2 minutes per epoch, yielding a 4× speedup. This corresponds to ~95% parallel efficiency considering communication overhead. Training was conducted using a single Meluxina node equipped with 4 A100 GPUs. GPU hours were calculated as the product of wall-clock training time and the number of GPUs, while node hours reflect actual node usage. For example, Model 3 consumed approximately 17 GPU hours, equating to ~4.25 node hours.

5. Conclusion and Future Work

5.1. Conclusion

This study explored and evaluated advanced deep learning architectures for large-scale natural event detection using image segmentation techniques. The investigation focused on models with varying architectural complexity and depth, trained on a diverse dataset of several thousands of images, representing different fire and water conditions and environments. The experimental results demonstrated that Higher-complexity models with deeper architectures achieved marginally higher segmentation accuracy but faced significant memory limitations, requiring smaller batch sizes and longer training durations. Moderate-complexity models provided a balanced trade-off between accuracy, training efficiency, and computational feasibility, enabling training with larger batch sizes without resource constraints. Real-world qualitative testing highlighted the effectiveness of lightweight architectures in event detection, particularly in scenarios requiring fast response and high scalability.

5.2. Future Work

While this study provides valuable insights into model performance and computational trade-offs, future research will explore even more complex architectures exceeding 100M parameters. Key optimizations include Gradient checkpointing for reducing memory overhead during backpropagation, Mixed-precision training to enhance computational efficiency, Model parallelization and distributed training for improved scalability. By addressing these challenges, the system can fully leverage high-capacity architectures while maintaining scalability, faster training times, and improved segmentation performance.

Acknowledgements

The authors extend their sincere gratitude to all team members for their invaluable contributions and support throughout this project. The majority of model training was conducted on the MeluXina supercomputer in Luxembourg, managed by LuxProvide and funded by the EuroHPC JU. The authors also acknowledge the invaluable support provided by both the EuroHPC and LuxProvide teams.

References

[1] D. M. J. S. Bowman *et al.*, "Fire in the Earth system," *Science*, vol. 324, no. 5926, pp. 481–484, 2009. DOI: 10.1126/science.1163886

[2] B. C. Ko, K. H. Cheong, and J. Y. Nam, "Fire detection based on vision sensor and support vector machines," *Fire Safety Journal*, vol. 44, no. 3, pp. 322–329, 2012. DOI: 10.1016/j.firesaf.2008.07.006

[3] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik, "Convolutional neural networks-based fire detection in surveillance videos," *IEEE Access*, vol. 6, pp. 18174–18183, 2018. DOI: 10.1109/ACCESS.2018.2812835

[4] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.

[5] EuroHPC Joint Undertaking, "MeluXina Supercomputer," [Online]. Available: https://www.eurohpc-ju.europa.eu/systems/meluxina

[6] LuxProvide, 2024. System overview - meluxina user documentation. URL: https://docs.lxp.lu/system/overview/. Accessed: July, 2024.

[7] M. A. Finney, "The challenge of quantitative risk analysis for wildland fire," Forest Ecology and Management, vol. 211, no. 1–2, pp. 97–108, 2005.

[8] Eduardo R. Oliveira et al. " The Detection of Small-Scale Open-Burning Agriculture Fires Through Remote Sensing,"(2025). DOI: https://doi.org/10.3390/rs17010051

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. CVPR, pp. 580–587, 2014.DOI: 10.1109/CVPR.2014.81

[10] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 834–848, 2018.DOI: 10.1109/TPAMI.2017.2699184

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.DOI: 10.1145/3065386

[12] Y. You, Z. Zhang, C. Hsieh, J. Demmel, and K. Keutzer, "ImageNet training in minutes," in Proceedings of the 47th International Conference on Parallel Processing (ICPP), 2018.DOI: 10.1145/3225058.3225069

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*,vol.32,2019.https://papers.nips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[14] Rayed, M.E., Islam, S.M.S., Niha, S.I., Jim, J.R., Kabir, M.M. and Mridha, M.F., 2024. Deep learning for medical image segmentation: State-of-the-art advancements and challenges. Informatics in Medicine Unlocked, 47, p.101504. https://doi.org/10.1016/j.imu.2024.101504

[15] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. arXiv preprint arXiv:1704.06857.

[16] Ben-Nun, T., & Hoefler, T. (2019). Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. ACM Computing Surveys (CSUR), 52(4), 1–43.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 256–270

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

The European High Performance Computing Joint Undertaking Infrastructure and Access Modes State-of-Play

Krishnakshi Bhuyan[a], Dora Marton[a], Catarina Guerreiro[a], Klara Meštrović[a]

[a]*The European High Performance Computing Joint Undertaking, 12E rue Guillaume Kroll, L-1882, Luxembourg*

Abstract

This paper presents the current state of play of the European High Performance Computing Joint Undertaking (EuroHPC JU) infrastructure with special emphasis on the access mechanisms and the projects that benefit from the allocated HPC resources. The EuroHPC JU procures world-class supercomputing, quantum computing and Artificial Intelligence (AI) Factories infrastructure in order to provide access to users whose research outputs and innovations contribute to societal benefits and industrial competitiveness. In 2025, the procurement of new systems and the launch of additional access calls were undertaken as part of an initiative to support projects using AI technologies in their applications, especially targeting industry, small and medium-sized enterprises (SMEs) and startups. Besides this new initiative, the EuroHPC JU continues to support all types of research coming from academia, industry and public sector with the existing well-established access mechanisms. In this context, an overview of the available and upcoming infrastructure is presented and an analysis of access mechanisms together with the projects accessing the infrastructure is showcased.

Keywords: High Performance Computing (HPC); Infrastructure; Supercomputers; Quantum Computers; AI Factories; Access calls; Artificial Intelligence (AI); HPC resources; node hours; research domains

1. The European High Performance Computing Joint Undertaking (EuroHPC JU)

The European High Performance Computing Joint Undertaking (EuroHPC JU) was established in 2018 in Luxembourg and it runs as a legal and funding entity whose main purpose is to make Europe a world leader in the field of supercomputing. In order to achieve this, the European Union (EU), together with the EuroHPC JU participating states, pool their resources to boost Europe's scientific excellence and industrial capabilities, support the digital transformation of its economy, and ensure its technological independence. Currently, there are 36 member states and associated countries that are members of the EuroHPC JU.

The key goal of the EuroHPC JU is to procure world-leading infrastructure (supercomputers, quantum computers, AI Factories) and to deploy, extend and maintain the corresponding ecosystem. The ecosystem is supported by a robust

10.1016/j.procs.2025.08.252

supply chain that ensures components, technologies and knowledge supporting a wide range of applications optimized to run on the procured infrastructure. With the availability of infrastructure, the EuroHPC goal is also to widen the use of the available systems to public and private users across Europe and to support the development of high-performance computing (HPC) skills for science, industry and public sector.

2. The EuroHPC JU Infrastructure

The EuroHPC JU has to date procured five (5) petascale systems – Vega in Slovenia, Karolina in Czechia, MeluXina in Luxembourg, Discoverer in Bulgaria and Deucalion in Portugal; three (3) pre-exascale systems – LUMI in Finland, Leonardo in Italy and MareNostrum 5 in Spain; and one (1) exascale system – JUPITER in Germany. In 2025, in addition to the existing supercomputers, access to JUPITER exascale system in Germany was made available to users, expanding the list of operational EuroHPC JU supercomputers. Additionally, in 2025, there were upgrades of two of the existing systems: Discoverer was upgraded to Discoverer+ with new functionalities enhancing its performance and AI capabilities, while Leonardo was upgraded to Leonardo Improved Supercomputing Architecture (LISA), which will incorporate an AI-optimized partition into Leonardo, providing greater support for the development of Large Language Models and multi-modal generative AI.

The most recent editions of the TOP500 and Green500 lists were released on 10 June 2025, where JUPITER ranked number 4 in the TOP500 list. With an intermediate computing power of nearly 800 petaflops, equivalent to 800 million billion calculations per second, JUPITER is currently Europe's fastest supercomputer. Soon to be capable of delivering 1 ExaFLOP of computing power, or over 1 trillion operations per second, JUPITER will become Europe's first Exascale supercomputer. Its first installed module, JEDI, has dominated the Green500 list since its debut in May 2024, maintaining a leading position. With more than 60 thousand millions operations per watt, the JUPITER Booster ranks 21st on the Green500 list of the world's most energy-efficient supercomputers. Once again, all operational EuroHPC supercomputers are featured on the TOP500 list of the world's most powerful systems, with two systems - LUMI (ranked 9th) and Leonardo (ranked 10th) - positioned in the global top 10.

JUNE 2025	TOP500	Green500
JUPITER	#4	#1(JEDI) #21(Booster)
LUMI	#9	#36
LEONARDO	#10	#70
MARENOSTRUM 5	#14	#44
MELUXINA	#136	#79
KAROLINA	#196	#76
DISCOVERER	#260	#244
DEUCALION	#299	#116
VEGA	#307	#287

Fig. 1: The June 2025 rankings of TOP500 and Green500 supercomputers lists

Additionally, the procurement process for the second exascale system, Alice Recoque in France, as well as two mid-range systems – Arrhenius in Sweden and Daedalus in Greece – are currently underway.

Apart from these developments, on 17 June 2024, EuroHPC JU received a new mandate to develop and operate AI Factories at locations surrounding EuroHPC supercomputing facilities. These AI Factories are dynamic ecosystems that will build around AI-optimized supercomputers, providing computing resources and support services to both

European industries and scientific users. With this new pillar of activities now formally integrated into its Regulation, the EuroHPC Joint Undertaking is able to acquire and operate dedicated AI-optimized supercomputers. This will include not only the procurement of new systems but also upgrading some existing EuroHPC supercomputers with enhanced AI capabilities. Since September 2024, three cut-offs have been launched for call for applications in order to facilitate the establishment of these AI Factories, of which, evaluations from the first two cut-offs have been finalized so far. Up until now, the JU has signed hosting agreements with 13 AI Factories, marking a major step forward in Europe's ambition to lead in artificial intelligence and supercomputing. The selected Hosting Entities include: LUMI AI Factory (Finland), HammerHAI and JAIF (Germany), Pharos (Greece), IT4LIA (Italy), L-AI Factory (Luxembourg), BSC AI Factory(Spain), MIMER (Sweden), AI:AT(Austria), BRAIN++ (Bulgaria), AI2F (France), PIAST(Poland), and SLAIF(Slovenia). Out of these, 11 sites will deploy new AI-optimized supercomputers, while France and Germany will operate alongside Europe's first exascale supercomputers, Alice Recoque and JUPITER respectively. Meanwhile, Spain's AI Factory will be established through the upgrade of the existing EuroHPC MareNostrum 5 system, and Greece will develop an AI Factory linked to DAEDALUS, the EuroHPC supercomputer currently being deployed.

Further, EuroHPC JU launched a call for proposals for the selection of entities or consortia of entities to establish AI Factory Antennas linked to selected AI Factories across Europe. The Antennas will ensure remote computing access to AI-optimized supercomputing resources from the linked AI Factory. The aim of the Antennas is to provide an opportunity to the EuroHPC JU Participating States to have access to AI Factory resources and services without having to invest in an establishment of a fully-fledged AI Factory.

In addition to this, the EuroHPC JU has invested in ten quantum computers offering 6 different quantum technologies, located across Europe. The procured system offering trapped-ions technology is Piast-Q in Poland; the systems offering the neutral atoms technology are Ruby in France, Jade in Germany and EuroQCS-Italy in Italy; the systems offering spin qubits technology are EuroSSQ-HPC in Netherlands and MeluXina-Q in Luxembourg; the system offering annealing technology is EuroQCS in Spain; the system offering star-shaped technology is VQL in Czechia; the system offering crystal technology is Euro-Q-Exa in Germany; and the system offering photonics technology is Lucy in France. Out of these ten, the inauguration of the first EuroHPC quantum computer, PIAST-Q in Poland, was undertaken on June 2025. PIAST-Q will offer a performance of 20 physical qubits along with several other unique features for European users. Like all other EuroHPC quantum computers currently under deployment, this system will be integrated with high-performance computing (HPC) infrastructure.

In order to connect all infrastructure, a platform for the federation of EuroHPC supercomputers, quantum computers and AI Factories is under development, which will act as a one-stop shop access point for users. This integration will enhance user access to advanced technologies and facilitate federated access to data lakes and data spaces across Europe to create a more connected and innovative supercomputing ecosystem in Europe.

3. Access to the EuroHPC JU Infrastructure

The EuroHPC JU offers a variety of access opportunities within different access modes. At the moment, the JU offers eight access calls, including calls for applications that use traditional HPC and calls for applications using AI. From 2021, the JU started offering access to the procured infrastructure by introducing the first calls that were Benchmark Access and Development Access, followed by the Regular Access and Extreme Scale Access. After a growing demand for access modalities for proposals using AI, the JU introduced in 2024 the AI & Data-Intensive Applications Access call. With the announcement of the upcoming AI Factories in 2025, the JU restructured its existing calls into Calls for Traditional HPC and additionally introduced AI Factories Access calls. Calls for AI Applications includes a call for scientific applications, called the AI for Science and Collaborative EU projects Access and three calls for Industrial Innovation – the Playground Access, Fast Lane Access and the Large Scale Access call.

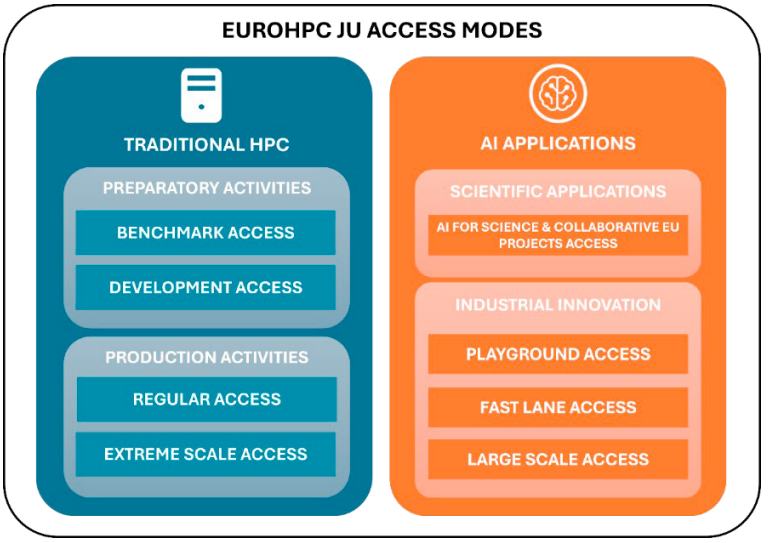


Fig. 2: EuroHPC JU Access Modes from April 2025

3.1. Access Modes for Traditional HPC Applications

The EuroHPC JU Access Modes for Traditional HPC Applications define the different modalities through which EuroHPC JU computing resources are made available to scientists and researchers for conducting fundamental research activities based on classical HPC usage. These access modes are categorized according to several criteria, such as the volume of computational resources offered, the complexity of the associated evaluation process, the type and maturity of applications targeted by each mode, and the frequency of submission cut-off dates. The EuroHPC JU allocates up to 45% of the current system resources dedicated to the EU share of access for the purposes of Traditional HPC Applications for all the categories of access presented in this section.

Taking into consideration calls for production activities (Extreme Scale Access, Regular Access and AI and Data-Intensive Applications access), in the cut-offs from December 2021 until April 2025, there were 691 proposals submitted out of which, the EuroHPC JU awarded 471 projects, with an overall of 112,516,283 node hours distributed to the successful applicants.

Table 1:EuroHPC JU awarded projects and resources throughout access modes for production activities in the period December 2021 – April 2025

Access call	Proposals awarded	Proposals awarded %	Node hours awarded	Node hours awarded %
Extreme Scale Access (Dec 2022-Oct 2024)	99	21%	72,719,646	65%
Regular Access (Dec 2021-Mar 2025)	293	62%	36,530,037	32%
AI And Data-Intensive Applications Access (Apr 2024-Apr 2025)	79	17%	3,266,600	3%
Total	**471**	**100%**	**112,516,283**	**100%**

Table 2: EuroHPC JU Awarded resources in different partitions throughout access modes for production activities in the period December 2021 – April 2025

Partition/call	Regular Access	Extreme Scale Access	AI and Data-Intensive Applications Access
Vega CPU	5,932,132	-	-
Vega GPU	313,978	-	7,100
MeluXina CPU	3,002,316	-	-
MeluXina GPU	1,475,775	-	100,000
Karolina CPU	2,755,099	-	-
Karolina GPU	361,750	-	7,500
Discoverer CPU	4,927,471	-	-
LUMI-C	8,674,022	14,035,287	-
LUMI-G	1,961,312	27,872,119	280,000
Leonardo Booster	2,706,696	20,435,010	2,200,000
Leonardo DCGP	1,227,989	994,400	-
MareNostrum5 GPP	1,665,698	5,860,630	-
MareNostrum5 ACC	596,092	3,522,200	672,000
Deucalion ARM	-	-	-
Deucalion x86	365,214	-	-
Deucalion GPU	37,840	-	-
JUPITER Booster	526,654	-	-
Total node hours	**36,530,037**	**72,719,646**	**3,266,600**

Among the 471 awarded projects, 107 projects (22,7%) are from Chemical Sciences and Materials, Solid State Physics, 31 projects (6.58%) are from Earth System Sciences & Environmental Studies, 161 projects (34.18%) are from Engineering, Mathematics and Computer Sciences, 118 projects (25.05%) are from Computational Physics: Universe Sciences, Fundamental Constituents of Matter, 52 projects (11.04%) are from Biochemistry, Bioinformatics, Life Sciences, Physiology and Medicine and 2 projects (0.42%) are from Socio-Economic Sciences and Humanities: Economics, Finance and Management, Linguistics, Cognition and Culture domain.

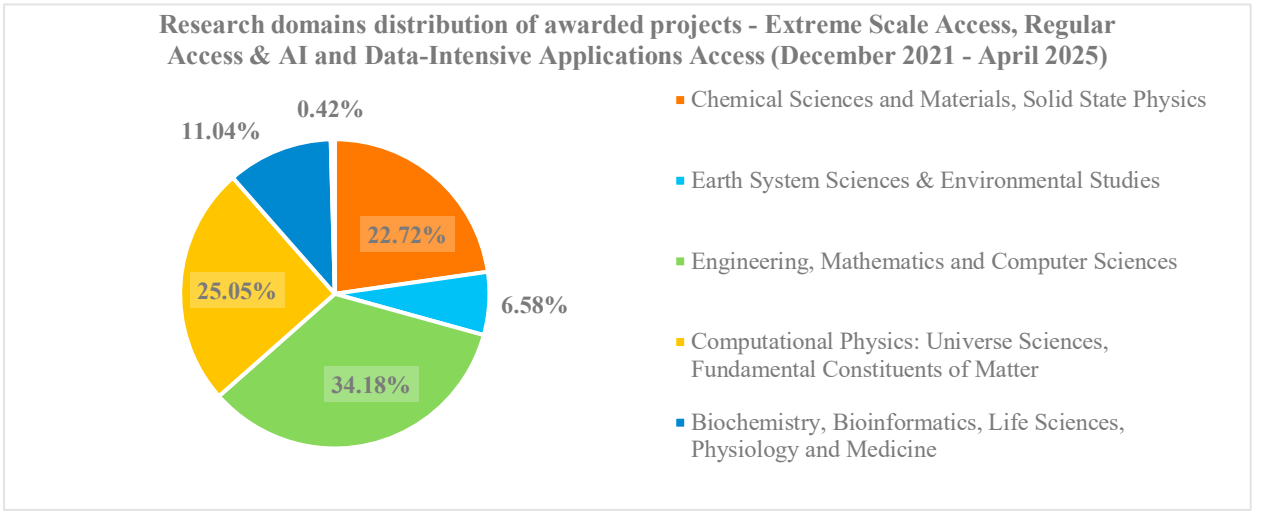
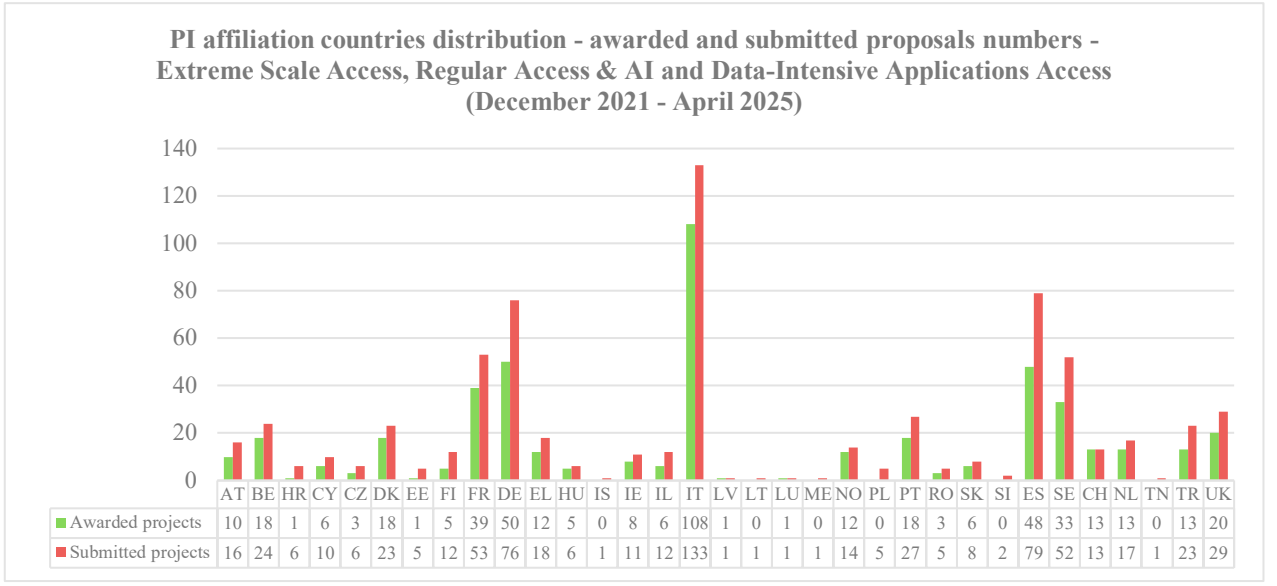


Fig. 3: Research domains distribution of awarded projects throughout access modes for production activities in the period December 2021 – April 2025

The Principal Investigator (PI) affiliation country distribution within these 691 submitted and 471 awarded projects shows that most of the projects are coming from PIs affiliated in Italy (22.93% awarded projects), Germany (10,62% awarded projects), Spain (10,19% awarded projects), France (8,28%) and Sweden (7,01% awarded projects). The remaining 40,97% of PI affiliation countries taking into consideration only awarded projects, are Austria, Belgium, Croatia, Cyprus, Czechia, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Israel, Latvia, Luxembourg, Netherlands, Norway, Portugal, Romania, Slovakia, Switzerland, Türkiye and United Kingdom.



	AT	BE	HR	CY	CZ	DK	EE	FI	FR	DE	EL	HU	IS	IE	IL	IT	LV	LT	LU	ME	NO	PL	PT	RO	SK	SI	ES	SE	CH	NL	TN	TR	UK
Awarded projects	10	18	1	6	3	18	1	5	39	50	12	5	0	8	6	108	1	0	1	0	12	0	18	3	6	0	48	33	13	13	0	13	20
Submitted projects	16	24	6	10	6	23	5	12	53	76	18	6	1	11	12	133	1	1	1	1	14	5	27	5	8	2	79	52	13	17	1	23	29

Fig. 4: Principal Investigator affiliation country distribution – awarded and submitted proposals throughout access modes for production activities in the period December 2021 – April 2025

When analyzing the PI gender of awarded projects, 84.71% were from male PIs, while 14.44% belong to the female PIs. The remaining 0.85% PIs preferred not to specify their gender.

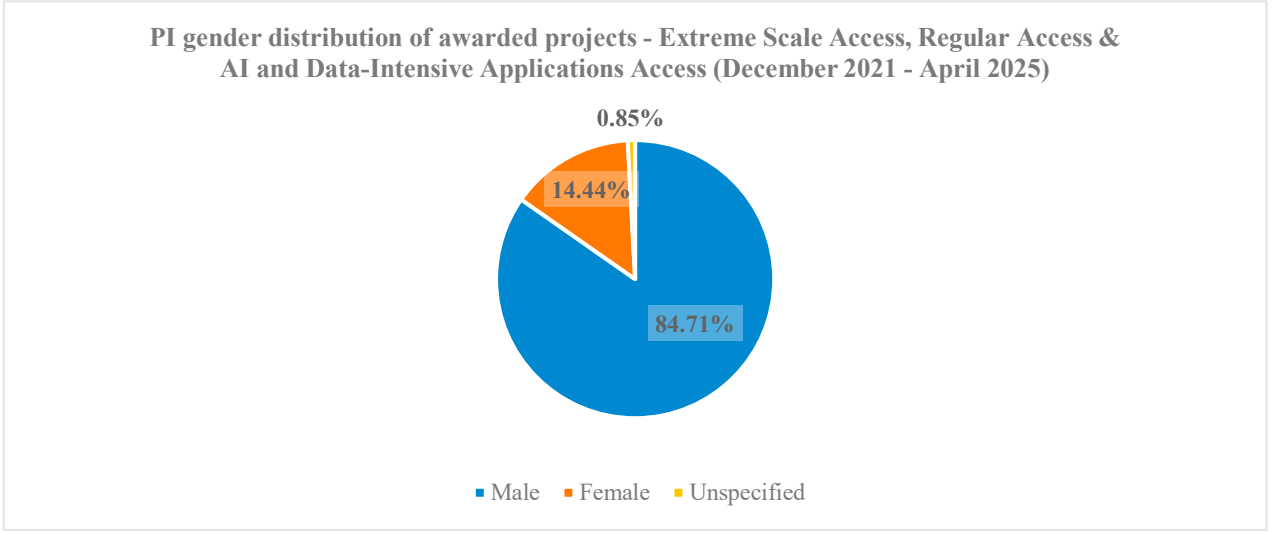


Fig. 5: Principal Investigator gender distribution – awarded proposals throughout access modes for production activities in the period December 2021 – April 2025

From December 2021 until April 2025, 131 projects that use AI technologies were awarded 12,891,481 node hours which is 11% of overall awarded resources of the calls for production activities.

Table 3: Awarded AI proposals throughout access modes for production activities in the period December 2021 – April 2025

Call	Awarded proposals	Resources awarded	Resources awarded %
Extreme Scale Access	18	7,112,121	55%
Regular Access	34	2,512,760	19%
AI and Data-Intensive Applications Access	79	3,266,600	25%
Total	**131**	**12,891,481**	**100%**

Within these 131 projects, 52 projects belong to Extreme Scale Access and Regular Access calls which is 40% of the total awarded AI projects. In 2025 within the first cut-off of the Regular Access call there was an increase of 29% of awarded AI projects. The growing trend of AI projects within the traditional HPC access calls shows the need for a special access mode for AI related applications.

The observed AI technologies used in these awarded projects are Machine Learning, Natural Language Processing, Generative Language Modeling, Deep Learning, Vision (image recognition, image generation, text recognition OCR, etc.), Audio (speech recognition, speech synthesis, etc.), Robotic process automation, Virtual agents, Decision management: Classified and statistical learning methods. The most represented technologies were Deep Learning (25,6%), Machine Learning (19,5%) and Generative Language Modeling (18,4%).

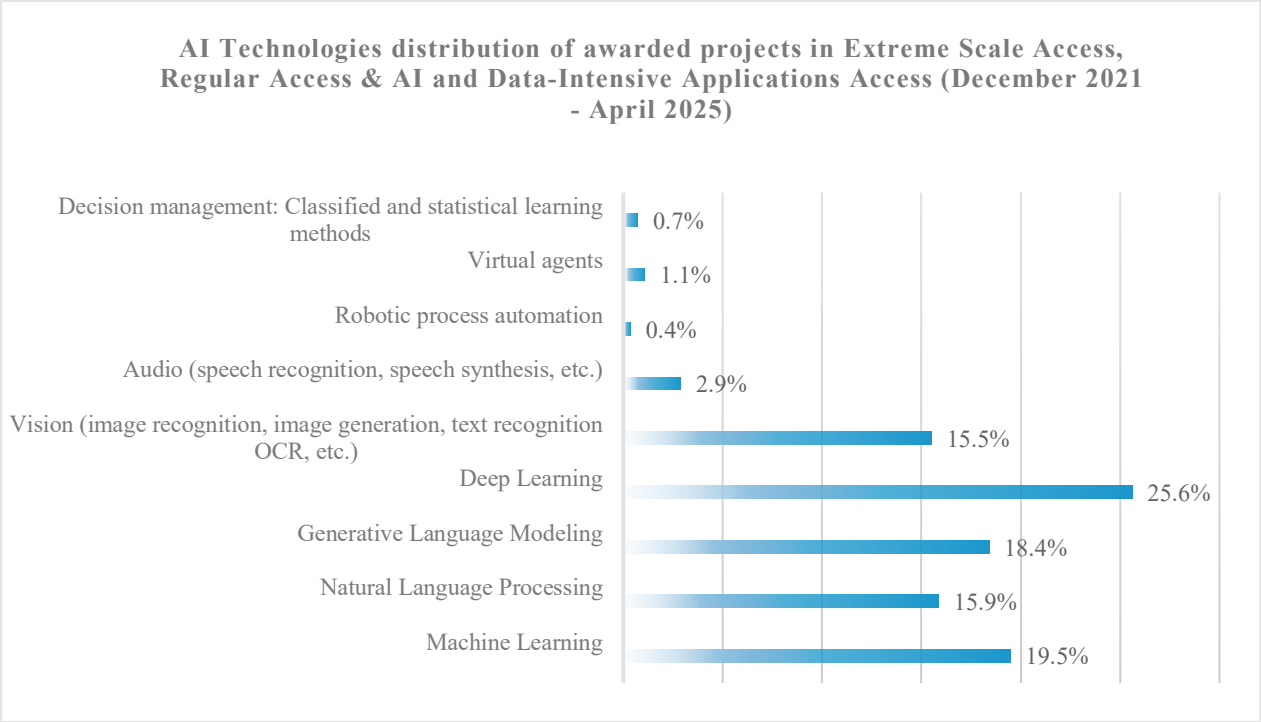


Fig. 6: AI technologies distribution within awarded AI proposals throughout access modes for production activities in the period December 2021 – April 2025

3.1.1. The Benchmark Access call

The Benchmark Access mode is intended for all categories of users who want to collect performance data or test a method, such as machine learning training, on a target system in order to document the technical feasibility of their applications to be submitted to other access modes.

Access is provided through a continuously open call with monthly cut-offs. All submitted proposals undergo an eligibility check and a Technical Assessment performed by the Hosting Entity of the selected partition(s). Access period is granted up to three (3) months.

In 2025 cut-off period (from 1st to 6th cut-off of 2025), 166 proposals were submitted in the call of which 147 (89%) have been awarded with a total of 465,400 node hours. Out of these 147 proposals, 47 were submitted by Principal Investigators (PIs) affiliated to Academia (32%), 33 were from Industry (22%) and 67 projects were from PIs affiliated to Public Sector (46%). Out of these 147 projects, 67 of them used AI technologies in their applications (46%).

Table 4: Benchmark Access proposal numbers from 1st to 6th cut-off of 2025

	Total	**Academia Involvement (PI)**	**Industry Involvement (PI)**	**Public Sector Involvement (PI)**	**AI Applications**
Submitted proposals	166	53	37	76	76
Awarded proposals	147	47	33	67	67

The distribution of awarded computational resources across the various systems that are available under the call is the following: SofiaTech (BG) 2%, MACC, FCT (PT) 4%, BSC (ES) 22%, CINECA (IT) 38%, CSC (FI) 22%, IT4Innovations (CZ) 4%, LuxProvide (LU) 3% and IZUM (SI) 4%.

3.1.2. The Development Access call

The Development Access mode is meant for projects focusing on code and algorithm development, development of workflows, HPC trainings, as well as Natural Language Processing, Foundation Models and other methods for AI applications. This access mode is mostly targeting medium size executions that do not target large scale production runs and is aiming for code and algorithmic validation before requesting access to an Extreme Scale Access or Regular Access call. Development access is provided through continuously open calls with monthly cut-offs. All submitted proposals undergo an eligibility check and a Technical Assessment performed by the Hosting Entity of the selected partition(s). Access period is granted for up to six (6) or twelve (12) months.

In the first two quarters of 2025, i.e., from January until June, 291 proposals were submitted in the call of which 259 (89%) have been awarded have been awarded with a total of 1,433,600 node hours. Out of these 259 proposals, 81 were from Principal Investigators (PIs) affiliated to Academia (31%), 52 were from Industry (21%) and a 126 projects were from PIs affiliated to Public Sector (47%). Out of these 259 projects, 153 of them used AI technologies for their application (59%).

Table 5: Development Access proposal numbers from 1st to 6th cut-off of 2025

	Total	Academia Involvement (PI)	Industry Involvement (PI)	Public Sector Involvement (PI)	AI Applications
Submitted proposals	291	91	62	138	173
Awarded proposals	259	81	52	126	153

The distribution of awarded computational resources across the various systems that are available under the call is the following: SofiaTech (BG) 3%, MACC, FCT (PT) 4%, BSC (ES) 24%, CINECA (IT) 28%, CSC (FI) 24%, IT4Innovations (CZ) 7%, LuxProvide (LU) 6% and IZUM (SI) 4%.

3.1.3. The AI and Data-Intensive Applications Access call

The AI and Data-Intensive Access mode was introduced in 2024 in order to accommodate the growing number submissions and requests related to Artificial Intelligence. The call is supposed to accommodate requests coming from all sectors with an emphasis on industry, SMEs and startups.

This access mode was established with faster evaluation process and overall quicker access to the resources considering the fast-paced environment of AI developments, with the results being communicated one (1) month after the cut-off date. The call had six (6) cut-off dates per year, with allocations granted for six (6) months or one (1) year. The submitted applications were peer-reviewed, and the process contained the following: an administrative check, technical assessment, expert evaluation and consolidation of the results.

From April 2024 until April 2025 cut-off period, there were 122 submitted proposals, out of which 79 were awarded. In total 3,266,600 node hours were distributed to successful applicants.

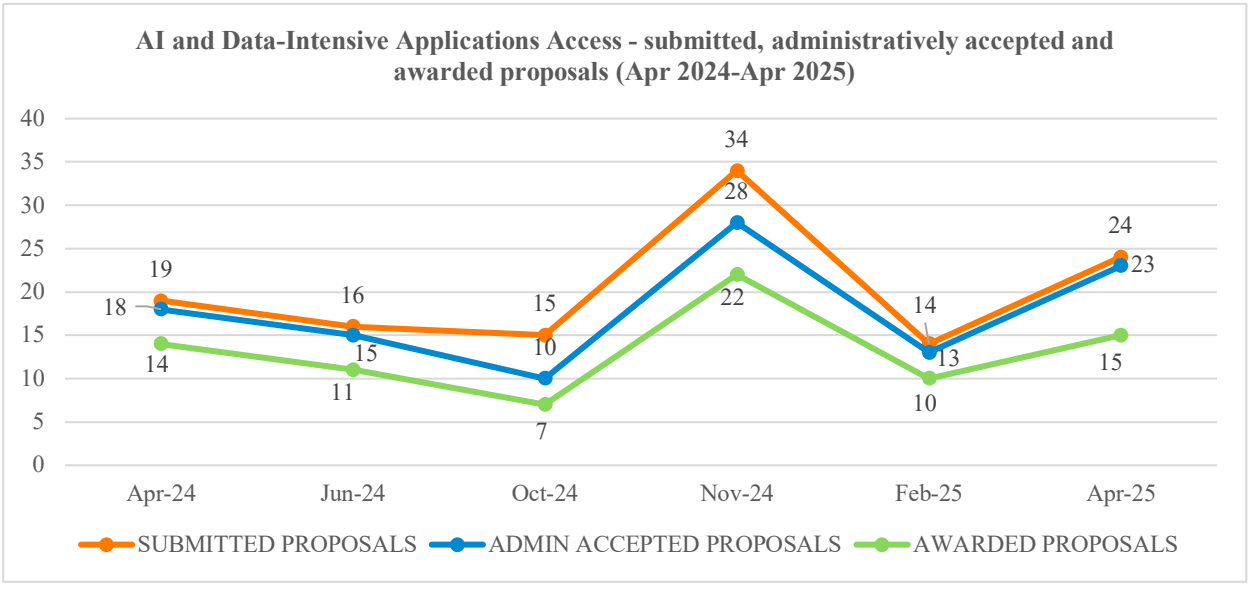


Fig. 7: AI and Data-Intensive Access mode-submitted, administratively accepted and awarded proposals (Apr 2024-Apr 2025)

Out of 79 awarded projects, the PI affiliation type distribution is the following: 38% (30 proposals) PIs are coming from an university, 25% (20 proposals) from an SME, 14% (11 proposals) from a research institute, 11% (9 proposals) from a startup, 6% (5 proposals) from a public entity and 5% (4 proposals) from a large enterprise.

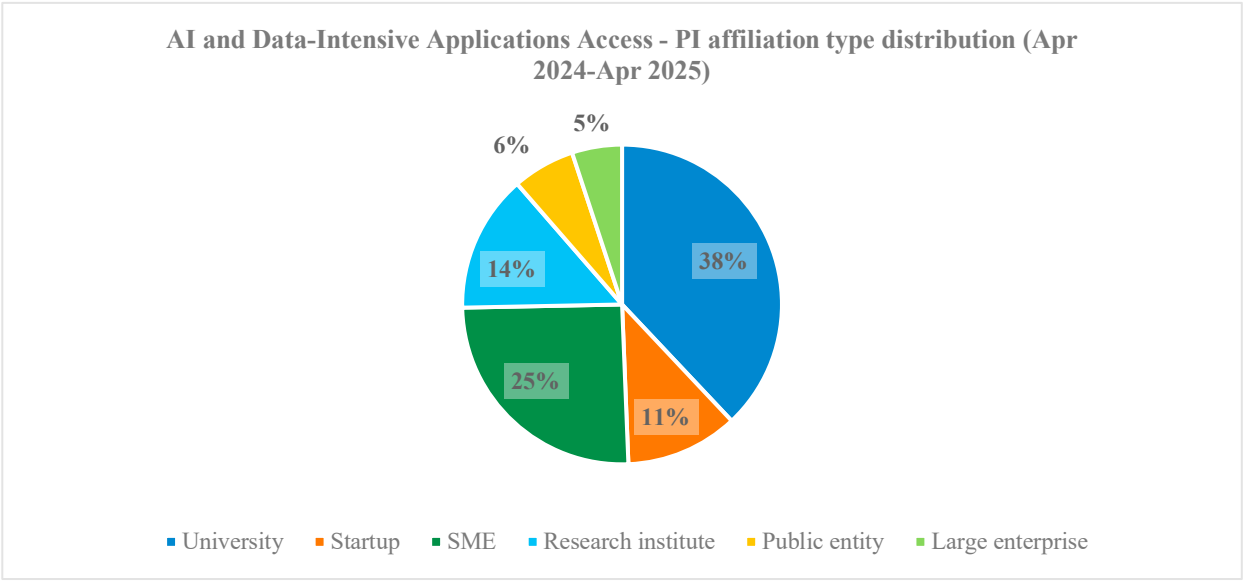


Fig. 8: AI and Data-Intensive Applications Access - PI affiliation type distribution (Apr 2024-Apr 2025)

3.1.4. The Regular Access call

The Regular Access call is open to all fields of science, industry and the public sector, and invites applications which present compelling cases that will enable scientific innovation in the domains covered. The expected impact in the application's domain should justify the need for large allocations in terms of compute time, data storage and support resources. This access mode allocates resources through a continuously open call for applications associated with two (2) cut-off dates per year, with allocations granted for a period of one (1) year.

Regular Access call allocate resources from all EuroHPC systems via a peer-review process for assessment of submitted proposals. The proposals undergo an evaluation process of four (4) months that entails an administrative check, technical assessment, rapporteur reporting and evaluation, domain based discussions and Super Panel (SP) meeting where all proposals are ranked, followed by a Resources Allocation Panel (RAP) meeting where the resources are distributed.

The Regular Access call is the longest running access mode for production activities. From December 2021 until March 2025, 293 proposals were awarded 36,530,037 node hours via the Regular Access calls. In the first cut-off of 2025, a total of 87 proposals were submitted in the call requesting a total of 11,872,706 node hours, of which 67 were awarded with 5,887,458 node hours.

Due to severe oversubscription within the March 2025 cut-off, in order to award more proposals, 7 proposals were suggested to be moved to the JUPITER Booster partition making them first projects being awarded on JUPITER (526,654 node hours). The allocation start for these projects will be slightly delayed until the system becomes fully operational.

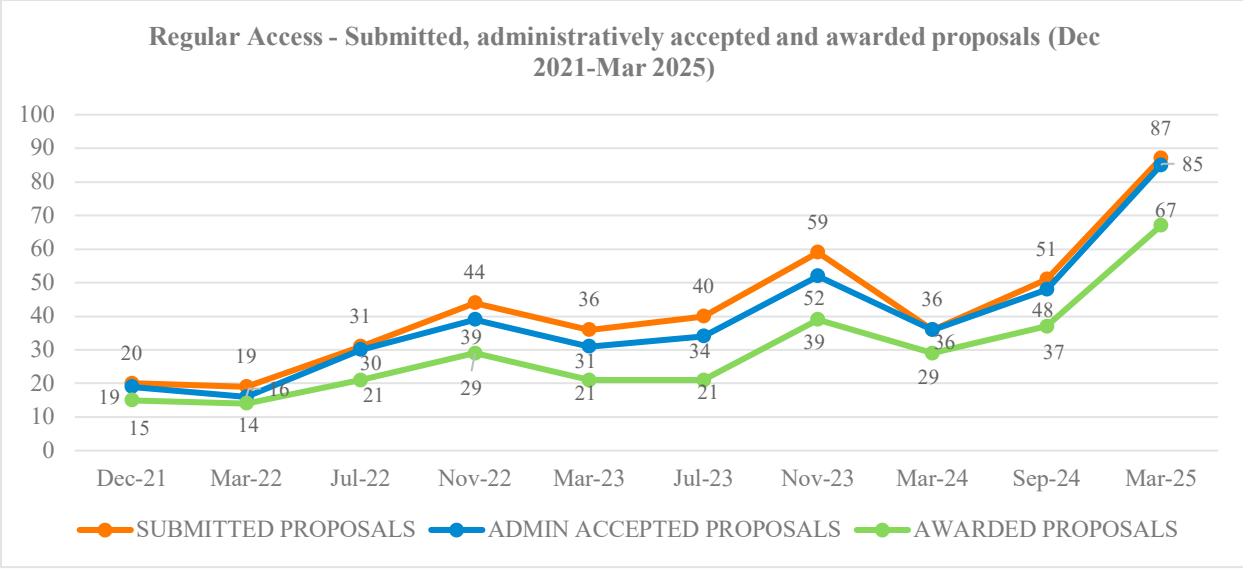


Fig. 9: Regular Access - Submitted, administratively accepted and awarded proposals (Dec 2021-Mar 2025)

The projects awarded within the Regular Access call come from different research domains and the distribution can be observed in the chart below.

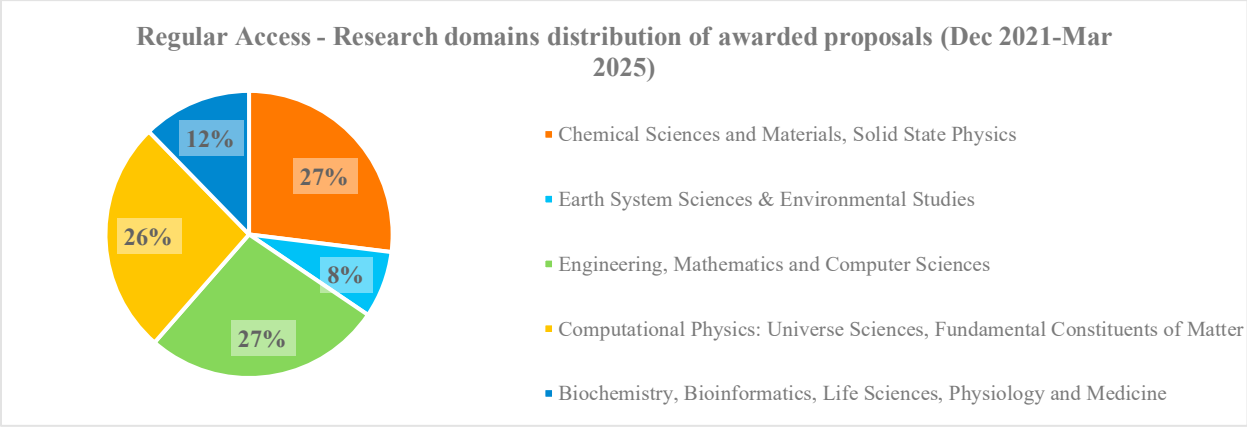


Fig. 10: Regular Access - Research domains distribution of awarded proposals (Dec 2021-Mar 2025)

3.1.5. The Extreme Scale Access call

This access mode calls for applications with high-impact, high-gain innovative research, open to all fields of science, industry and public sector justifying the need for and the capacity to use extremely large allocations in terms of compute time, data storage and support resources. Flagship scientific applications that are able to exploit the full scale of EuroHPC exascale and pre-exascale supercomputers are the main target for Extreme Scale Access. Resources are allocated through a continuously open call for applications with two (2) cut-offs per year, with allocations granted for a period of one (1) year. Extreme Access calls allocate resources from the high-end EuroHPC systems, i.e., pre-exascale and exascale. All submitted applications under the call are peer reviewed. The proposals undergo an evaluation process of six (6) months that entails an administrative check, technical assessment, scientific evaluation,

response to reviews stage, rapporteur reporting and evaluation, Access Resource Committee (ARC) meeting where all proposals are ranked, followed by a Resources Allocation Panel (RAP) meeting where the resources are distributed.

The first cut-off of 2025 was on 24 April 2025, with total of 44 proposals administratively approved, requesting 36,207,203 node hours across the offered infrastructure, the EuroHPC JU's pre-exascale systems LUMI, Leonardo and MareNostrum 5, as well as the exascale system JUPITER, the Booster partition that is offered for the first time within the JU's access calls. The 44 proposals are undergoing evaluation and the results are expected to be communicated in late September/early October 2025.

The Extreme Scale Access call remains to be the call that offers and allocates most of the JU access time, which is visible from the overall resources distribution across different access modes, with a notice that the last submissions belonging to the April 2025 cut-off are not taken into consideration.

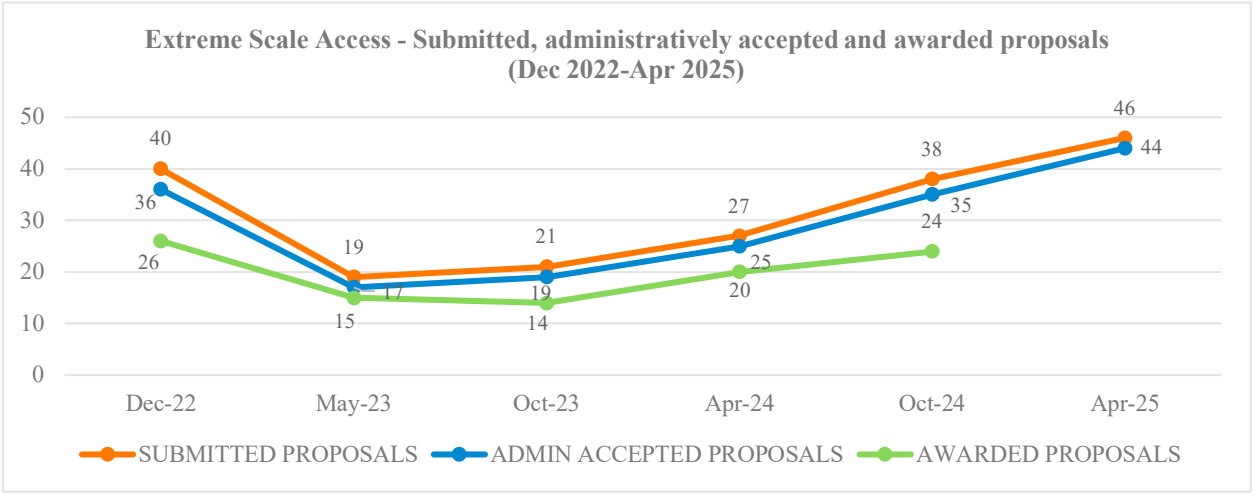


Fig. 11: Extreme Scale Access - Submitted, administratively accepted and awarded proposals (Dec 2022-Apr 2025)

The projects awarded within the Extreme Scale Access call come from different research domains and the distribution of these domains can be observed in the chart below.

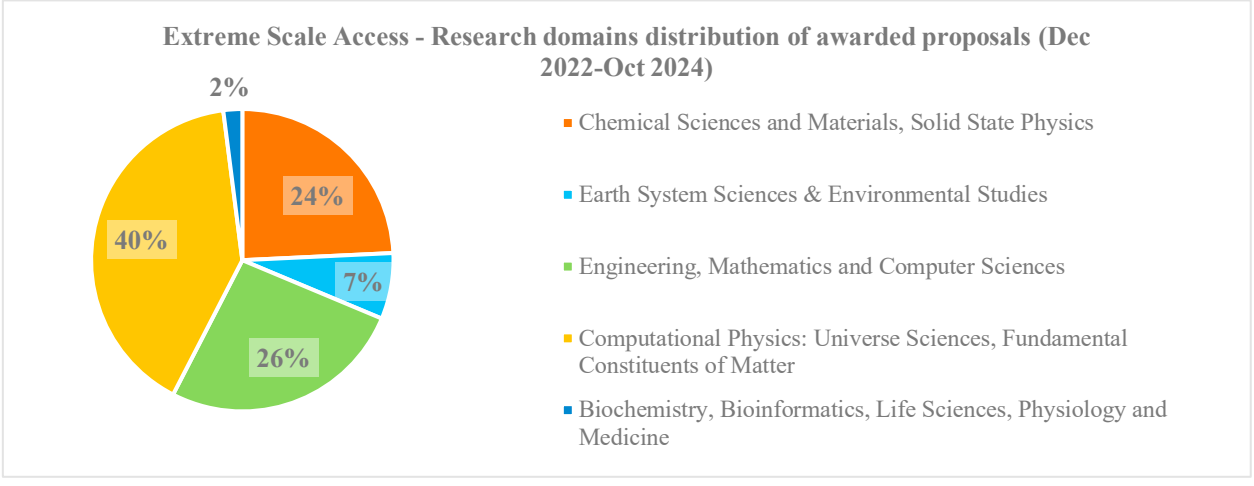


Fig. 12: Extreme Scale Access - Research domains distribution of awarded proposals (Dec 2022-Oct 2024)

3.2. AI Factory Access Modes

During 2025, the EuroHPC JU is in the process of selecting the Hosting Entities that will establish the AI Factories. In order to serve the immediate needs of the European AI industrial and scientific users, EuroHPC JU will rely on the existing operational supercomputers (pre-exascale and petascale systems) capable to support AI applications, to offer resources to the European AI users until the newly procured AI Factories are operational. For that purpose, the EuroHPC JU introduced different AI Factories Access modes. The available partitions within these modes are the following: Vega GPU, MeluXina GPU, Discoverer GPU, LUMI-G, Leonardo Booster and MareNostrum 5 ACC.

The AI Factories Access modes AI for Science and for Collaborative EU Projects and AI for Industrial Innovation aim to support ethical Artificial Intelligence, using a spectrum of different AI technologies such as machine learning and generative AI. These access modes allocate:

- AI for Science and Collaborative EU Projects: up to 25% of the overall EuroHPC share of access time;
- AI for Industrial Innovation: up to 30% of the overall EuroHPC share of access time.

3.2.1. The AI for Science and Collaborative EU Projects Access

This access mode supports AI applications for science, covering all types of scientific users, users from public sector, as well as industrial users participating in R&I projects funded by EU Programmes such Horizon Europe or the Digital Europe Programme.

This access mode allocates resources from the existing EuroHPC partitions with AI capabilities and allocates resources through a continuously open call for applications with six (6) cut-off dates per year (one cut-off date every two months). The allocations are granted for six (6) months after undergoing a peer review process for assessment of submitted proposals. The peer-review process entails an administrative check, technical assessment, expert evaluation and consolidation of the results.

The first proposals submitted to this call (within the June 2025 cut-off) are under evaluation and will not be presented in this paper.

3.2.2. The AI for Industrial Innovation Access calls

The AI Factories Industrial Innovation Access calls are intended for users coming from industry, SMEs and startups that need HPC resources for AI applications. These calls offer up to 30% of the EU share of the access time. The AI Factories Industrial Innovation Access calls include three access modes, targeting different users and compute needs:

- **Playground Access**, providing fixed resources per application (5,000 GPU hours) for entry-level users, with allocation results communicated within two (2) working days. This call is continuously open with no cut-off dates. All submitted proposals undergo an eligibility check and a technical assessment.
- **Fast Lane Access**, for users already familiar with HPC requiring up to 50,000 GPU hours, with allocation results communicated within four (4) working days. This call is continuously open with no cut-off dates. All submitted proposals undergo an eligibility check and a technical assessment.
- **Large Scale Access**, intended for applications requiring more than 50,000 GPU hours, with allocation results communicated within ten (10) working days after the cut-off date. This call is continuously open with two (2) cut-offs date per month. All submitted proposals undergo an administrative check, technical assessment, Industrial Innovation Group evaluation and ranking performed by the selected AI Factory.

From April 2025 until July 2025, a total of 82 proposals were submitted out of which 50 were awarded resources. Overall, 10,616,560 GPU hours have been allocated to successful projects (93% resources were allocated within the Large Scale Access call).

Table 6: AI for Industrial Innovation Access calls - submitted and awarded proposals (April 2025 – July 2025)

Call	Proposals submitted	Proposals awarded	Awarded %
Playground Access	31	24	48%
Fast Lane Access	32	16	32%
Large Scale Access	19	10	20%
Total	**82**	**50**	100%

3.3. Strategic Access

The EuroHPC JU can allocate up to 10% of its resources share to strategic initiatives of public interest. The percentage limit is aggregated across all Strategic Access initiatives and these projects so not need to undergo a submission and peer-review process due to their nature.

Within this framework, from 2022, the European Commission's flagship project Destination Earth (DestinE) has access to the available EuroHPC JU infrastructure. The DestinE project is advancing the development of highly-accurate, high-resolution, digital model of the Earth, a Digital Twin, in order to monitor, model, simulate and predict natural phenomena (e.g. natural disasters) and to tackle environmental challenges. The project has access to LUMI, Leonardo, MareNostrum5, MeluXina, and also on the upcoming exascale system JUPITER. At present, over 300 scientists, analysts, and HPC professionals are engaged in DestinE project, creating an integrated, cross-disciplinary research and innovation environment. This collaborative initiative exemplifies how strategic support can transform scientific advances into operational climate intelligence for the benefit of society.

3.4. Experts involved in the peer-review process

The Access calls intended for production activities involve engagement of several experts in each cut-off of the respective calls and over the past years, the EuroHPC JU has established an extensive collaboration with these experts. The EuroHPC JU engages high level professionals in High Performance Computing across different domains who conduct an unbiased and transparent evaluation of submitted proposals. In total, since establishment of the first access call in 2021, more than a 1000 experts were involved in the review process till date, of which 81 have worked as Chairs, 543 as Rapporteurs and 412 as external referees (Scientific Reviewers). 49% of these experts were involved in the Extreme Scale call, 38% in the Regular Access call and 3% in the AI and Data-Intensive Applications Access call.

3.5. Application Support Team (AST)

Under the EuroHPC JU EPICURE project, the proposals awarded via the Access calls are able to have additional support up to 6 months for application porting, optimization and scalability improvements. The aim of the project is to improve the user support services for successful applicants, to ensure an efficient and timely execution of the applications. In addition, the project includes the development of a single point of contact (in the form of a European HPC Application Support portal) that will allow European HPC users from public and private sector including SMEs, to retrieve information on the systems offered by the Joint Undertaking, their architectures, their access mechanisms, and the support services available. Applicants can include in their proposal if they would need assistance from the Application Support Team (AST) while submitting their application to one of the EuroHPC JU Access calls or the respective Hosting Entities might suggest them during the evaluation process, if they see a need for it in a proposal.

AST service was launched in second half of 2024 for the access calls and since then, 165 proposals were supported via the EPICURE project in the Benchmark and Development call, 22 proposals were supported under the Regular Access call, 15 proposals under the Extreme Scale Access call and 37 proposals were supported under the AI and Data Intensive Applications call.

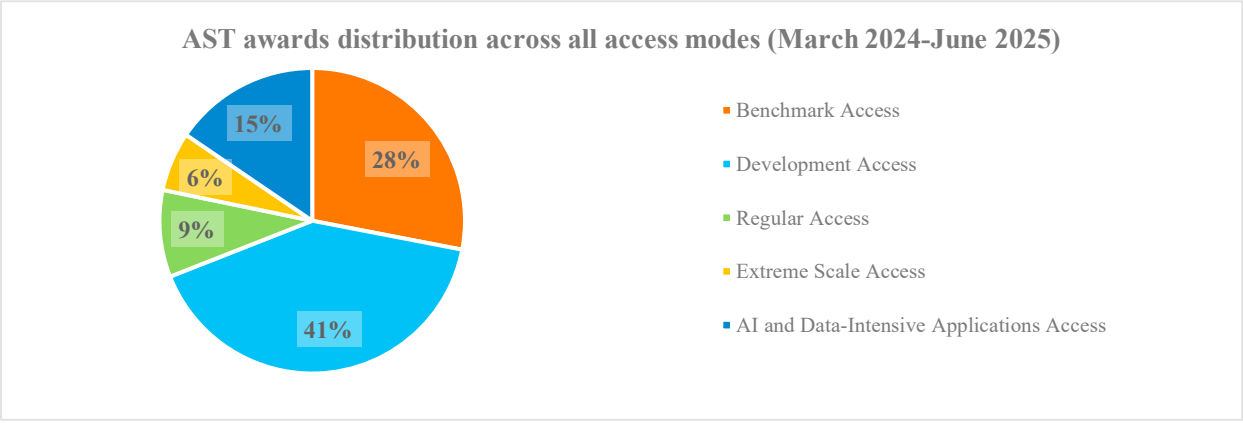


Fig. 13: Application Support Team awards distribution across all access modes (March 2024-June 2025)

4. Conclusion

Overall, in the first two quarters of 2025, it was observed that the number of submitted proposals have increased taking into account the previous years average; Regular Access with more than 100% increase in submitted proposals and Extreme Scale Access with approximately 35% increase in submitted proposals. Observing Benchmark and Development Access calls, in comparison with 2024, within the 2025 cut-off period (until June 2025) it is noted that the number of submitted proposals is already 70% of the total submitted proposals in 2024. If we consider the same numbers to project the next quarters of 2025, we can estimate the submission numbers to grow by 40% until the end of the year. In addition, proposals using AI technologies have substantially increased in comparison to 2024 across all calls, despite having a designated access call for AI applications. Due to this increase and interest to support industry in this sector, the EuroHPC JU additionally introduced AI Factories access calls for industrial innovation in 2025 which already show interest from the community in the few months that the calls have been opened. The impact of these newly introduced access calls will be better observed in the near future.

EuroHPC JU welcomes the increasing interest in applications submitted across various calls and remains committed to supporting groundbreaking and innovative research initiatives spanning diverse sectors.

Acknowledgement

The authors would like to thank their colleagues from the EuroHPC JU Infrastructure Unit, especially members of the Peer-Review Sector for their valuable contributions in the access calls management and their continous dedication.