Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) iii-iv

Procedia Computer Science

www.elsevier.com/locate/procedia

Table of Contents

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 1–2

Procedia Computer Science

www.elsevier.com/locate/procedia

Preface

The European High Performance Computing Joint Undertaking (EuroHPC JU) is delighted to present the 4th edition of the EuroHPC User Days Book of Proceedings which highlights some of the best work completed with access to the EuroHPC JU supercomputing resources in the last year.

Since 2021, EuroHPC JU has been supporting scientists across Europe to advance their research in critical scientific areas, by providing access to EuroHPC JU supercomputers through different access modes in AI and HPC. In 2025, access to EuroHPC JU supercomputers was granted to more than 1,600 groundbreaking research projects. In June 2026, EuroHPC JU added a new access mode for European researchers in quantum computing.

EuroHPC JU allows the European Union, and its Participating States, to coordinate their efforts and pool resources to deploy technological advancements in the field of supercomputing, AI and quantum computing. The EuroHPC JU ecosystem boosts scientific excellence, industrial strength and digital transformation while ensuring European technological sovereignty. Each year EuroHPC works to expand the base of European scientific and industrial users who leverage these resources to advance scientific, social, and business knowledge and actions.

EuroHPC JU also funds various user support services such as the National Competence Centres (NCCs) and Centres of Excellence (CoEs), as well as dedicated skills initiatives including EPICURE, Minerva, Fortissimo Plus (FFplus) and Evita. These services provide expert guidance to new users as well as delivering training initiatives, supporting application development, and sharing technological expertise to accelerate European research and competitiveness. EuroHPC JU also supports cross-deployment and cross-benchmarking with non-EU nations (Japan, India, and Latin American and Caribbean region) increasing the breadth and availability of scientific codes in all regions.

The papers in this Book of Proceedings were presented as part of the 2026 EuroHPC User Days in Dublin, Ireland. EuroHPC User Days is an annual event and an important opportunity to engage with user communities and showcase and celebrate research completed using EuroHPC resources. The diversity of this research is well represented in the span disciplinary areas covered in the papers this year which include computational physics, earth sciences and climate, artificial intelligence, engineering, and life sciences.

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.012

EuroHPC JU is grateful to the researchers who submitted their work to this publication. Papers were selected based on optimal use of allocated resources for innovative research. Peer review of the received manuscripts was performed by an editorial board of scientific reviewers, and we would take this opportunity to thank the reviewers, presenters and the organising partners who contributed to the success of the EuroHPC User Days 2026. I would also like to thank the EuroHPC JU team and in particular, Krishnakshi Bhuyan, the Editor-in-Chief, for her work on this publication. 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 286 (2026) 3

Procedia Computer Science

www.elsevier.com/locate/procedia

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.013

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 4–13

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

Mitigating Catastrophic Forgetting in Multilingual Continual Pretraining: Lessons from EU Institutional LLMs

Carolina Oliveira Costa[a,*], Bhavani Bhaskar[a,*], Hans Ewetz[a,*], Amin Sleimi[a,*], Markus Runonen[a,*], Csaba Oravecz[a], Simon Hengchen[b,*], Mario Ceresa[c], Mihai Cristian Brașoveanu[a], Ilja Rausch[a,**]

[a] *European Commission, Directorate-General for Translation*
[b] *European Commission, Directorate-General for Communications Networks, Content and Technology*
[c] *European Commission, Joint Research Centre*

Abstract

Third-party large language models often lack the domain expertise, language coverage, and regulatory compliance that EU institutions require. We performed continual pretraining of Mixtral Mixture-of-Experts 8×7B model on over 100 billion tokens across all 24 official EU languages from EURAMIS, the European Commission's translation support system. To mitigate catastrophic forgetting, we applied hierarchical data packaging with token density normalization, integrated replay data using language-specific ratios, and employed warm-up cycles with adaptive learning rate scheduling. We evaluated on standard benchmarks, a custom EU Formal Language sentence completion task, and toxicity probes. The results revealed trade-offs between multilingual adaptation and high-resource language preservation. Replay mechanisms partially mitigate degradation. This paper provides empirically grounded guidance for practitioners developing multilingual LLMs for public sector applications using European HPC infrastructure.

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026

Keywords: Large Language Models; Continual Pretraining; Catastrophic Forgetting; Low-resource Languages;

1. Introduction

Large language models (LLMs) have transformed natural language processing, yet their development by third-party entities present significant challenges for public institutions, particularly regarding domain expertise, data quality, language coverage, transparency, and regulatory compliance. These challenges are amplified for the European Union (EU) with its 24 official languages and strict regulations on transparency, accountability, and fairness. EU institutions require certain capabilities, such as EU-specific terminology and parliamentary language services that commercial LLMs fail to provide.

* Contributed as a contractor to the European Commission
** Corresponding author. Tel.: +32 2 29 86108
E-mail address: Ilja.RAUSCH@ec.europa.eu, Bhavani.BHASKAR@ext.ec.europa.eu, Carolina.OLIVEIRA-COSTA@ext.ec.europa.eu

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.014

To promote digital sovereignty and address these needs, we developed EU institutional LLMs through continual pretraining of Mixture-of-Experts (MoE) models [12] from the Mixtral family on high-quality multilingual data from EURAMIS, the European Commission's translation support system [22]. However, continual pretraining faces a key challenge: catastrophic forgetting, where models lose prior knowledge when adapting to new data [13, 19, 25]. In our previous study, we observed this phenomenon when continually pretraining Llama2 13B on Slovenian and Croatian data from EURAMIS [14, 20], motivating the strategies adopted in the present work. Unlike fine-tuning or training from scratch, continual pretraining updates existing models with new data to increase knowledge while preserving broad capabilities. This approach requires full-weight updates across multiple languages, demanding high-performance computing and sophisticated parallelization to manage computational bottlenecks.

Catastrophic forgetting remains unsolved for large-scale LLMs and must be mitigated [11]. Existing mitigation techniques show limited efficacy, with most studies confined to small-scale experiments. In multilingual settings with 24 languages, forgetting is amplified by linguistic disparities and data imbalance: low-resource languages like Irish or Maltese lack sufficient training data, while high-resource languages like English or French may experience pronounced forgetting under distributional shifts [10], degrading performance across critical tasks including translation, summarization, and dialogue systems. This paper shares practical insights from mitigating catastrophic forgetting during continual pretraining of the Mixtral 8×7B model, on 100B+ token datasets. We document the adaptation of established mitigation strategies in this mid-scale, multilingual context, providing empirically grounded guidance for practitioners developing robust multilingual LLMs for public sector applications.

To address the challenges of multilingual continual pretraining, we developed a hierarchical data packaging structure that balances textual information across all 24 languages (detailed in Section 2.2). We investigated replay data integration by incorporating content similar to the base model's original training data with varying ratios across language groups. Our experimental framework included three primary checkpoints: Mixtral-EU-24, trained exclusively on EURAMIS data, Mixtral-EU-24-replay-1 and Mixtral-EU-24-replay-2, which incorporate replay mechanisms to mitigate catastrophic forgetting. Training was conducted on the Leonardo Booster supercomputer [23] via EuroHPC JU, utilizing data parallelism, pipeline parallelism, and expert parallelism to optimize computational efficiency.

We evaluated our models across multiple dimensions to assess their capabilities and limitations. Standard benchmarks (ARC [6], HellaSwag [26], Global MMLU [21]) measured general reasoning and language understanding, while a custom EU Formal Language (EUFL) sentence completion task assessed adaptation to institutional texts. We further evaluated translation using EURAMIS and Flores+ datasets [17], and toxicity resistance through the garak framework [7]. Our evaluation revealed fundamental trade-offs between multilingual adaptation and preserving capabilities in high-resource languages.

The remainder of this paper is organized as follows: Section 2 describes the data preparation pipeline and data packaging strategies for EURAMIS and replay data; Section 3 presents the model architecture, training infrastructure, and efficiency metrics; Section 4 details the evaluation methodology and results across multiple benchmarks including standard tests, EU formal language tasks, translation capabilities, and toxicity assessment; Section 5 concludes the paper.

2. Data

The continual pretraining data of our EU institutional LLM consists of a large-scale, multilingual dataset comprising 24 languages. The dataset was sourced from the EURAMIS internal translation support system at the European Commission, which contains a wide range of documents, including parliamentary proceedings, legislative texts, and administrative documents. To support conversion of EU institutional data into documents to be used in continual pretraining, we implemented a flexible data processing workflow. The workflow architecture allows for rapid modification of data processing steps and quick integration of new components, enabling fast iteration and adaptation.

2.1. EURAMIS as a source for training data

EURAMIS is a translation support system developed by the European Commission for translating official documents into the 24 official EU languages [4, 14]. The system centers on a large-scale, multilingual database of trans-

lation memories, providing high-quality translated text data across multiple languages. Extensive quality controls are applied before adding new data to EURAMIS, minimizing the risk of including poorly written documents.

EURAMIS contains approximately 2B segments across all 24 EU official languages, spanning documents from 1998 to 2024. EURAMIS data has a skewed distribution: high-resource languages such as English, French, and German are substantially better represented than low-resource ones like Irish or Maltese. The data is stored in a segment-based format, where a segment is a piece of raw, unprocessed text extracted from various documents across several EU institutions, covering legislative acts, court judgments, parliamentary questions and administrative correspondence. Full documents can be reassembled using the metadata attached to each segment.

Due to the standardized format of EURAMIS documents, we investigated whether text duplication could compromise data diversity. We applied both the MinHash algorithm and exact matching to detect near-duplicates and duplicates at segment and document levels. We analyzed randomly sampled datasets ranging from 1K to 10K documents, with sample sizes varying by comparison type. The results reveal sufficient diversity within EURAMIS data (99% non-duplicate documents for exact matches, 84% for segments; 88% and 76% respectively for close matches), suggesting limited gains from de-duplication.

Despite EURAMIS's high-quality data, filtering was required. EURAMIS documents contain Excel tables, Word tables, and PowerPoint slides, often unsuitable for LLM training due to lack of context as well as a prevalence of very short sentences. Since EURAMIS lacks metadata identifying table-like data, sentence-level removal was unsafe. Instead, we incorporated rudimentary filters in the data processing workflow to detect and remove this content at the document level.

2.2. *Packing EURAMIS data into training data*

We identified the following aspects related to the packaging of tokenized text into training data: (i) structural aspects; (ii) informational content based on language; (iii) over and under sampling of data; (iv) language order.

The structural aspect refers to how data is partitioned for a given language as a stream of tokens. Specifically we packaged data following a hierarchical structure as shown in Figure 1.

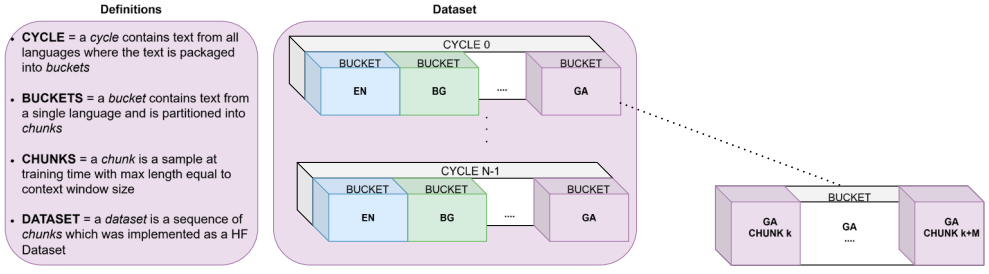


Fig. 1. Hierarchical structure of data packaging

Using the tokenizer of the base model yields varying token counts for the same text across languages due to uneven token distribution. We applied mitigation strategies to address this disparity. We used informational content to package equal amounts of textual information across languages. First, we calculated token density [2]: the ratio of tokens in a language to tokens in English for the same textual content, computed across a large set of translated documents. The token density enabled us to normalize tokenized text and select an equivalent amount of textual information across tokenized text in each language. Token density ranged from 1.0 (English) to 4.6 (Greek). The informational content of text was then calculated as the number of tokens divided by the language's token density.

Over and under sampling becomes necessary if we want the model to see the same amount of data for each language, when the amount of data is not evenly distributed across languages. This implies that before the model has seen all English data, it will have seen the Irish data multiple times. In the case of EURAMIS data, the large discrepancy between the amount of data in English compared to Irish can cause overfitting for Irish. To prevent such overfitting, we selected Portuguese as our mid-distribution baseline and calculated its available informational content in EURAMIS. Low-resource languages were oversampled by randomly reusing previously seen data, while high-resource languages were undersampled by randomly discarding excess data.

Multiple approaches exist for language ordering (bucket ordering). We selected a random order initially and repeated it for each cycle, as we found no compelling rationale for any specific ordering strategy.

We set the English bucket size to 8M tokens and used token density to calculate equivalent amounts for the remaining 23 EU languages. Token counts for the same text volume varied significantly across the 24 languages due to Mixtral's unbalanced tokenizer. The data was processed through a workflow, serialized into a HuggingFace dataset, stored on disk, and streamed for continual pretraining.

To mitigate catastrophic forgetting caused by the distributional shift between the base model's training data and the continual pretraining data, a warm-up data cycle, structured like a standard cycle but reduced to 10% of its size, was introduced.

2.3. Packing EURAMIS and replay data into training data

The initial continual pretraining run revealed varied performance across languages. High-resource languages showed decreased performance due to catastrophic forgetting, while low-resource languages showed an improvement. To address this, we incorporated replay data—similar to the base model's original training data—to mitigate catastrophic forgetting. We partitioned languages into 4 groups, each with a specified replay-to-EURAMIS data ratio as shown in Table 1. Replay data was appended as a block after EURAMIS data within each bucket. We maintained the original EURAMIS data volume in each bucket and padded the bucket with a specified percentage of replay data. The goal was to counteract potential knowledge loss: the model would learn from a large EURAMIS block but might forget base model knowledge, which the subsequent replay block would help restore.

Table 1. EURAMIS /replay ratio.

Language	replay/Euramis
en	70%
fr it de es	50%
bg cs da hr hu nl pl pt ro sl sv	30%
el et fi ga lt lv mt sk	10%

We sourced replay data from FineWeb-Edu[1] (English) and FineWeb2[2] (all other EU languages). These datasets were selected for their high-quality, multilingual web-crawled content (covering all 24 EU languages), which likely aligns with Mixtral's undisclosed training data. Their rigorous de-duplication, quality filtering, and large scale delivered sufficient diversity to avoid oversampling while providing a realistic distribution of web content.

Towards the end of continual pretraining with replay data we observed catastrophic forgetting in English and responded by repackaging data so every second bucket was English. We also reduced bucket sizes to 70% of their previous size to increase English exposure frequency. This intervention proved effective in mitigating catastrophic forgetting for English, as discussed further in Section 4.

3. Model architecture and training

3.1. Base models and final checkpoints

We used the model Mixtral-8×7B-v0.1 (~47B parameters) from the Mixtral family [12], which has MoE architecture. Continual pretraining produced several checkpoints: Mixtral-EU-24, Mixtral-EU-24-replay-1 and Mixtral-EU-24-replay-2. We selected the Mixtral family of models for two reasons: first, their European origin aligns with EU strategic priorities for a sovereign AI ecosystem; second, their computationally tractable yet sufficiently large scale suited deployment on our allocated EuroHPC Joint Undertaking AI and Data-Intensive Applications infrastructure.

[1] https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu
[2] https://huggingface.co/datasets/HuggingFaceFW/fineweb-2

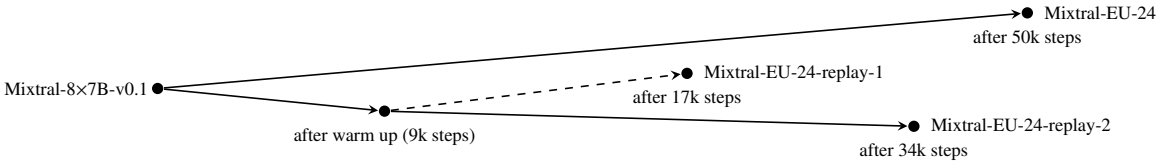


Fig. 2. Model lineage. All checkpoints originate from Mixtral-8×7B-v0.1. Replay-2 forks from replay-1 at step 9k. Dashed arrow: discontinued.

3.1.1. Mixtral-EU-24

The Mixtral-EU-24 checkpoint was obtained through continual pretraining of the Mixtral-8×7B-v0.1 model on approximately 108 billion tokens sourced from the EURAMIS dataset. The objective was fluency across all 24 official EU languages, including low-resource languages such as Irish and Maltese.

Training ran for over 50,000 steps, processing ∼2M tokens per step, requiring approximately seven weeks on 16 Leonardo Booster nodes. The training data consisted of one warm-up cycle followed by 264 full cycles, structured as described in the Section 2 using EURAMIS-only data. The learning rate schedule used was a warm-stable-decay with cosine annealing [24], a 10% warm-up and a maximum learning rate of 10^{-5}.

While Mixtral-EU-24 improved performance in low-resource languages, it suffered catastrophic forgetting in high-resource languages (see Section 4 for more details). This first checkpoint served as the baseline for follow-up experiments aimed at mitigating catastrophic forgetting through controlled integration of replay data.

3.1.2. Mixtral-EU-24-replay

The Mixtral-EU-24-replay-1 checkpoint was obtained through continual pretraining of the Mixtral-8×7B-v0.1 base model. The checkpoint aimed to mitigate catastrophic forgetting, particularly in English, while improving overall performance beyond the Mixtral-EU-24 baseline. The Mixtral-EU-24-replay-1 checkpoint employed a training dataset combining EURAMIS and replay data. Data was organized into one initial warm-up cycle followed by 309 complete training cycles, each ensuring equal representation of all languages. Data buckets incorporated a controlled blend of EURAMIS and replay data, with proportions adjusted per language (see Section 2). To enable adaptive training, we applied a flat cosine annealing learning rate schedule with reduced warm-up compared to Mixtral-EU-24, allowing dynamic adjustment of the EURAMIS-to-replay data ratio. Despite these optimizations, the replay augmentation did not effectively address catastrophic forgetting in English after training for ∼17k steps (see Section 4 for more details), prompting a strategic revision of the data packaging strategy.

To address Mixtral-EU-24-replay-1's performance limitations, we trained a second checkpoint by repackaging the dataset. First, we reduced language bucket sizes. Second, we inserted English-only buckets after each non-English bucket to enhance English exposure. We resumed pretraining from an early Mixtral-EU-24-replay-1 checkpoint (post-warmup, before the onset of English degradation). We adopted the flat cosine annealing strategy for the replay experiments as a technique to enable us to continue training with new data without having to warm up. However, we found that after the initial annealing, a schedule with re-warming and re-decay, allowed us to update the data mix mid-run without collapsing learning. Hence, we maintained the identical learning rate schedule but modified the annealing phase to start at ∼50% of training progression, followed by cycles of rewarming and cosine annealing for the remaining duration. The new checkpoint, Mixtral-EU-24-replay-2 outperformed both Mixtral-EU-24 and Mixtral-EU-24-replay-1 on standard benchmarks, but underperformed on the EURAMIS sentence continuation task (see Section 4 for more details). This stems from (1) lower EURAMIS concentration per bucket, reducing model fit to EURAMIS data and (2) fewer training steps due to resource constraints. As shown in Figure 2, Mixtral-EU-24 branches independently from the base model, while Mixtral-EU-24-replay-2 forks from the same post-warmup checkpoint as Mixtral-EU-24-replay-1 with a revised data strategy.

3.2. Training

Large-scale continual pretraining of models such as Mixtral 8×7B or larger on datasets exceeding 100 billion tokens requires substantial high-performance computing resources.

To optimize training efficiency, we utilized Colossal-AI [15], a scalable deep learning system designed for large-model parallel training. Our training approach employed a three-dimensional parallelism strategy with the training configuration show in Table 2.

Table 2. Training configuration

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
Precision	bf16	Max. sequence length	4096	Batch size	32
Microbatch size	1	Optimizer	HybridAdam	ZeRO [18] stage	1
Pipeline Parallelism size	4	Expert Parallelism size	4	Data Parallelism size	1

We evaluated performance using two key metrics: floating-point operations per second (FLOPS) and model FLOPs utilization (MFU) [16]. FLOPS derives from total FLOPs per training step divided by iteration latency while MFU quantifies the ratio of achieved FLOPS per GPU to theoretical maximum throughput. Accurate measurement is particularly challenging for large-scale models and sparse architectures like MoE, where selective expert activation complicates performance profiling.

Our MoE models initially achieved ~28.8% MFU under standard conditions with proper node allocation. We computed FLOPs based on the ~13B active parameters per token (2 of 8 experts), not the full 47B. In total, our training on 16 nodes of around 876989 core hours with an average of 90 FLOPS/GPU was ca.35.5x 109 TFLOP. We calculated MFU using the methodology outlined in [5, 1]. This improvement highlights how architecture-specific optimizations critically impact training efficiency.

We compared our MFU measurements with results reported by Meta. Meta achieved 38-43% MFU on their dense Llama3 models [16]. The discrepancy between our initial MoE results and Meta's dense model performance stems from fundamental architectural differences: MoE's inherent sparsity reduces computational load per step since not all weights are actively trained at each iteration, often leaving GPU capacity underutilized compared to dense training. Additional factors include variations in resource availability and optimization strategies, highlighting the unique challenges in scaling and optimizing sparse LLM architectures.

4. Evaluation

4.1. Standard benchmarks

Our evaluation strategy employed three established benchmarks to assess different dimensions of language model capabilities. The AI2 Reasoning Challenge (ARC) [6] benchmark tests reasoning ability through multiple-choice questions. It comprises Easy and Challenge evaluation sets. The HellaSwag [26] benchmark evaluates common sense natural language inference by requiring models to select the most plausible continuation for a given context. The Global MMLU [21] benchmark assesses multilingual understanding and reasoning across both culturally sensitive and culturally agnostic knowledge domains.

All benchmarks were implemented using EleutherAI's lm-evaluation-harness framework [9]. ARC and HellaSwag were only available in English within this framework, so we machine translated them into target languages using DGT's in-house translation engines. Global MMLU was available in 15 languages, including six EU languages (German, English, Spanish, French, Italian, and Portuguese). We did not translate it further due to the complexity of accurately rendering culturally sensitive questions through machine translation.

Figure 3a) and 3b) show that the base model retains a slight advantage on high-resource languages, indicating some forgetting from continual pretraining. Both replay variants, however, yield substantial gains on low-resource languages: on ARC Easy, Irish improves from approximately 0.27 to 0.37, Lithuanian from 0.33 to 0.47, and Maltese from 0.32 to 0.50. Mixtral-EU-24-replay-1 and Mixtral-EU-24-replay-2-step-14k were both trained for a comparable number of steps (~14k). Mixtral-EU-24-replay-2-step-14k, which employs alternating English buckets and reduced bucket sizes, recovers more of the base model's English performance than Mixtral-EU-24-replay-1 while maintaining comparable low-resource gains, confirming the effectiveness of the repackaging intervention described in Section 2.

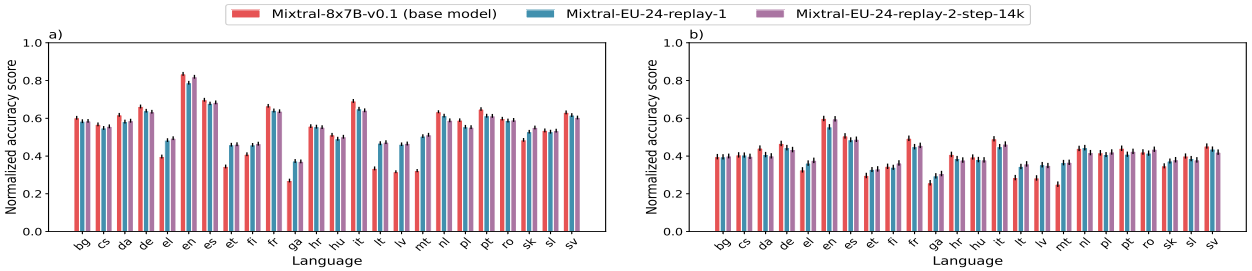


Fig. 3. **a)** Normalized accuracy on ARC Easy across 24 EU languages for the base model (red) and two replay variants trained for ~14k steps. Replay models show substantial gains on low-resource languages while Mixtral-EU-24-replay-2-step-14k (purple) better recovers English performance through alternating English buckets and reduced bucket sizes than Mixtral-EU-24-replay-1 (blue). **b)** Normalized accuracy on ARC Challenge across 24 EU languages at the ~14k step checkpoint. The harder benchmark confirms the same trend: replay variants improve low-resource language performance, with Mixtral-EU-24-replay-2-step-14k showing stronger English retention than Mixtral-EU-24-replay-1.

Figure 4a) and 4b) extend the ARC evaluation to all 24 EU languages, now including Mixtral-EU-24. The trends from the 14k checkpoint are confirmed and amplified: low-resource languages such as Irish, Maltese, Greek, and Lithuanian show the largest improvements over the base model on ARC Easy, while high-resource languages experience only moderate degradation. Among the replay variants, Mixtral-EU-24-replay-2 again demonstrates stronger English retention, consistent with the earlier checkpoint.

Figure 4c) presents HellaSwag results across 24 languages. The base model peaks at approximately 0.84 on English, with continually pretrained variants trailing by 4 to 7 percentage points on high-resource languages. Low-resource languages benefit substantially, with Irish, Maltese, and Greek improving by 10 to 21 percentage points. Mixtral-EU-24 and Mixtral-EU-24-replay-2 perform comparably, suggesting that commonsense reasoning is largely preserved regardless of the replay configuration. Figure 4d) shows Global MMLU results across six EU languages and reveals the clearest benefit of replay for knowledge retention. Without replay, Mixtral-EU-24 drops substantially to 0.43 to 0.55 compared to the base model's 0.64 to 0.71, while Mixtral-EU-24-replay-2 achieves intermediate scores (0.57 to 0.65). This consistent ordering across all six languages confirms that integrating replay data effectively mitigates catastrophic forgetting of general world knowledge, enabling the model to expand its multilingual coverage without sacrificing the broad capabilities of the original model.

These findings highlight a critical challenge: maintaining performance on established benchmarks while adapting models to new linguistic domains. Replay mechanisms offer only partial mitigation of capability degradation. The results reveal an inherent tension between optimizing for high-resource versus low-resource language performance.

4.2. EU formal language (EUFL) sentence completion

We developed the EUFL sentence completion task to evaluate model adaptation to EURAMIS data. This task presents models with an incomplete segment from formal EU texts and four possible continuations, only one of which is correct. The correct continuation was extracted from the original text, while three distractors were synthetically generated using GPT-4o mini. To prevent data leakage, we exclusively used documents created after the training data extraction date. The task provides a challenging benchmark requiring models to demonstrate comprehension of context, syntax, and semantic characteristics of EU institutional texts, offering insights into generalization capabilities within this specialized domain.

As shown in the Figure 5, the benchmark evaluation reveals that continual pretraining substantially improved model performance, with notable differences across training configurations. Mixtral-EU-24 achieved the best performance overall, while Mixtral-EU-24-replay-2 outperformed the base model across all languages but fell short of Mixtral-EU-24. This suggests that Mixtral-EU-24-replay-2 exhibited less fitting to EUFL texts, likely due to the inclusion of replay data and fewer training steps compared to Mixtral-EU-24. Mixtral-EU-replay-1 showed the weakest performance among the fine-tuned models, which can be attributed to its training on considerably less data than both Mixtral-EU-24 and Mixtral-EU-24-replay-2.

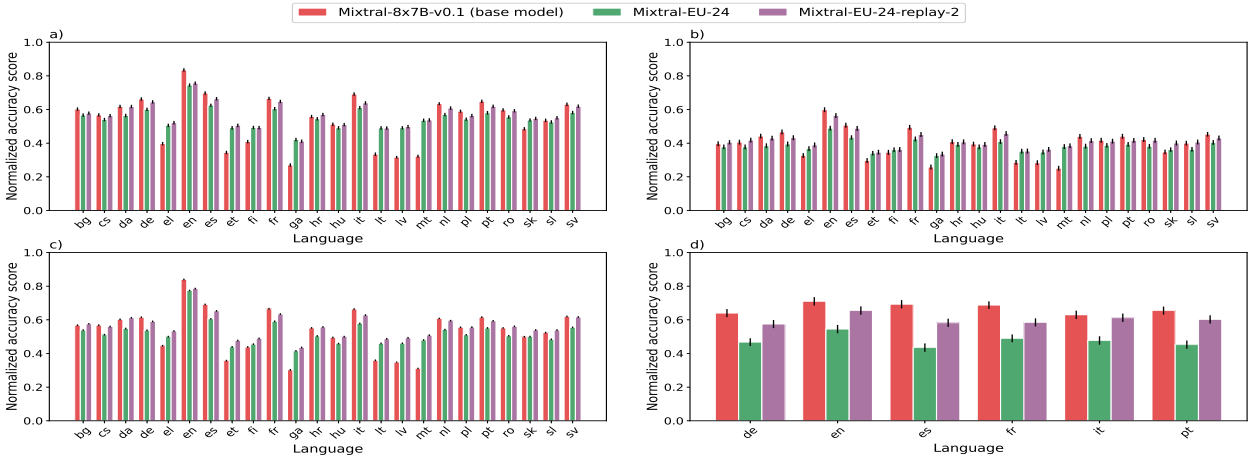


Fig. 4. **a)** Performance across 24 languages on ARC Easy. Base model (red) shows highest scores, particularly for English, while all of the continual pre-trained variants demonstrate more uniform cross-lingual performance. **b)** Normalized accuracy on ARC Challenge across 24 languages. Base model maintains advantage in high-resource languages, while continual learning variants show reduced performance gaps across languages. **c)** Commonsense reasoning performance across 24 languages. Base model peaks at 0.84 for English. The other variants maintain competitive performance with improved cross-lingual consistency. **d)** Multi-task language understanding across six European languages. Mixtral-EU-24 shows substantial degradation compared to base model, while replay strategies seem to have less catastrophic forgetting.

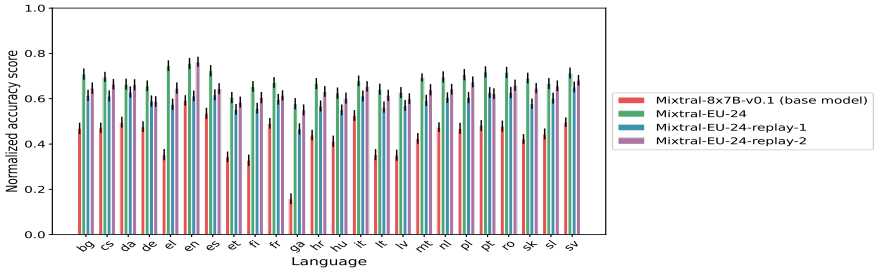


Fig. 5. Performance on EUFL Sentence Continuation task across 24 languages. continual pretrained models substantially outperform base model, indicating successful learning of EURAMIS linguistic conventions.

4.3. Translation benchmarks

Evaluating our models for machine translation presented unexpected challenges, particularly in maintaining consistent performance across structured tasks. We assessed machine translation capability using two datasets: a custom EURAMIS dataset and the Flores+ [17] dataset, evaluating all 24 languages in both English-to-X and X-to-English directions under zero-shot and few-shot settings. After continual pretraining, checkpoint models showed worse instruction-following ability, producing hallucinations, repeated prompts, and nonsensical responses. Few-shot exemplars yielded marginal improvements but behavior remained inconsistent across input segments. These findings demonstrate that reliable translation evaluation requires explicit instruction alignment through either instruction-tuned data during training or post-training fine-tuning on translation tasks.

4.4. Assessing profanity generation

We investigated model vulnerabilities across 24 languages using garak [7], a framework for assessing LLM robustness through structured probing. We adapted the Language Model Risk Cards (lmrc) [8] probe family, which covers eight toxicity categories (Anthropomorphisation, Bullying, Deadnaming, Sexual Content, Sexualisation, Slur Usage, Profanity and QuackMedicine). We excluded QuackMedicine from our evaluation, creating 168 language- and category-specific probes and detectors with custom multilingual data for the remaining seven categories. Each cate-

gory has a probe containing one or more prompts, with outputs evaluated by a detector that checks whether the LLM generated forbidden words.

For each language, we retrieved toxic words from Toxicity-200, the toxicity subset of Flores-200, and used Anthropic's Claude 3 Sonnet [3] used to classify them into the 7 categories. Native speakers verified classifications before we machine-translated the probe prompts. For Hungarian, poor Toxicity-200 data quality required sourcing words from an alternative resource[3] with native-speaker validation.

Mixtral base passed 96.9% of tests, while the Mixtral-EU-24-replay-2 passed 98.0%, with failures occurring in multiple but not all languages. This confirms that continual pretraining on EURAMIS data did not degrade the base model's toxicity resistance and maintains performance comparable to instruction-tuned models.

Limitations include incomplete toxic vocabulary coverage in Toxicity-200, non-comparable toxicity across languages, basic string-matching detection, and machine-translated probes.

5. Conclusion

This paper demonstrates that continual pretraining of large MoE language models at the 47B parameter scale and beyond is feasible on EU institutional data. Using EURAMIS as a multilingual, high-quality source across 24 EU languages, we trained Mixtral-EU-24, Mixtral-EU-24-replay-1 and Mixtral-EU-24-replay-2 from the Mixtral-8×7B-v0.1 base model. Training on the Leonardo Booster supercomputer achieved ~28.8% MFU through three-dimensional parallelization (data, pipeline, and expert parallelism), demonstrating the feasibility of mid-scale multilingual LLM development within European HPC infrastructure.

Across the experiments we observed catastrophic forgetting from several angles: performance dropped on established English-centric benchmarks (ARC Easy/ Challenge, HellaSwag) during continual pretraining, even as the model improved on our in-domain EU formal language (EUFL) sentence continuation task; mixing of replay data improved outcomes in low-resource EU languages relative to the base model, but Global MMLU performance lagged behind the base model; and periodic mid-training probes showed clear loss of prior capability in high-resource languages. Translation evaluation exposed degraded instruction-following capabilities post-training, indicating that continual pretraining alone is insufficient for reliable task performance without subsequent instruction fine-tuning.

Nonetheless, we reduced catastrophic forgetting through a combination of learning rate, data, and training-process interventions. First, we adopted a warm-up–flat–anneal learning rate schedule with cosine annealing. Second, we introduced replay with language-specific ratios using FineWeb2 and FineWeb-Edu to retain general-domain competencies while fitting to EURAMIS data. Third, we restructured the data reducing all bucket sizes and exposed the model to English text more often. In aggregate, Mixtral-EU-24-replay-2 showed smaller degradation in high-resource languages and competitive or improved performance in several low-resource EU languages compared to the base model, although it underperformed the more tightly EU-fitted Mixtral-EU-24 on EUFL due to lower concentration of EURAMIS data and fewer training steps. Following continual pretraining, Mixtral-EU-24-replay-2 was instruction fine-tuned.

Future work should consolidate these lessons into a more principled continual pretraining regimen. Key directions include: re-tokenising to better reflect multilingual token distributions, especially for low-resource EU languages and to minimise drift when vocabularies or data distribution change; changing the base model to a larger multilingual model and scaling continual pretraining to completion for larger models. Broadening multilingual evaluation to include culturally sensitive tasks and comprehensive safety probes is another important direction of work for robust deployment in institutional settings.

Acknowledgements

We thank the native speakers who verified toxicity classifications across languages and contributed to sourcing and validating alternative toxic word lists where existing resources were insufficient. We acknowledge EuroHPC JU for awarding the project ID EHPC-AI-2024A02-026 access to the Leonardo Booster supercomputer by CINECA. The

[3] https://www.webtrax.hu/csszl/#dictionary

computational resources were essential for the large-scale continual pretraining conducted in this work. During the preparation of this work the authors used Claude AI in order to seek inspiration and ideas for the transformation of the authors' thoughts into text. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

[1] Abnar, S., Shah, H., Busbridge, D., Ali, A.M.E., Susskind, J., Thilak, V., 2025. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. URL: https://arxiv.org/abs/2501.12370, arXiv:2501.12370.

[2] Ali, M., et al., 2024. Teuken-7B-base & Teuken-7B-Instruct: Towards European LLMs. URL: https://arxiv.org/abs/2410.03730, arXiv:2410.03730.

[3] Anthropic, 2024. The Claude 3 model family: Opus, Sonnet, Haiku. URL: https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf. accessed: 2026-02-27.

[4] Braşoveanu, M.C., Bhaskar, B., Rausch, I., 2023. Better multilingual AI for Europe: Translators and supercomputers at the European Commission – DG Translation Internal publication/Presentation.

[5] Chowdhery, A., et al., 2022. Palm: Scaling language modeling with pathways. ArXiv abs/2204.02311. URL: https://api.semanticscholar.org/CorpusID:247951931.

[6] Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., Tafjord, O., 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. URL: https://arxiv.org/abs/1803.05457, arXiv:1803.05457.

[7] Derczynski, L., Galinkin, E., Martin, J., Majumdar, S., Inie, N., 2024. garak: A Framework for Security Probing Large Language Models .

[8] Derczynski, L., Kirk, H.R., Balachandran, V., Kumar, S., Tsvetkov, Y., Leiser, M.R., Mohammad, S., 2023. Assessing language model deployment with risk cards. URL: https://arxiv.org/abs/2303.18190, arXiv:2303.18190.

[9] Gao, L., et al., 2024. The language model evaluation harness. URL: https://zenodo.org/records/12608602, doi:10.5281/zenodo.12608602.

[10] Gogoulou, E., Lesort, T., Boman, M., Nivre, J., 2024. Continual learning under language shift. URL: https://arxiv.org/abs/2311.01200, arXiv:2311.01200.

[11] Guo, Y., Fu, J., Zhang, H., Zhao, D., Shen, Y., 2024. Efficient continual pre-training by mitigating the stability gap. URL: https://arxiv.org/abs/2406.14833, arXiv:2406.14833.

[12] Jiang, A.Q., et al., 2024. Mixtral of experts. URL: https://arclaudexiv.org/abs/2401.04088, arXiv:2401.04088.

[13] Lange, M.D., van de Ven, G., Tuytelaars, T., 2023. Continual evaluation for lifelong learning: Identifying the stability gap. URL: https://arxiv.org/abs/2205.13452, arXiv:2205.13452.

[14] Leick, J.M., 1995. EURAMIS: Integrated multilingual services for a large multilingual community, in: Proceedings of Machine Translation Summit V, Luxembourg, Luxembourg. URL: https://aclanthology.org/1995.mtsummit-1.15/.

[15] Li, S., Liu, H., Bian, Z., Fang, J., Huang, H., Liu, Y., Wang, B., You, Y., 2023. Colossal-ai: A unified deep learning system for large-scale parallel training, in: Proceedings of the 52nd International Conference on Parallel Processing, Association for Computing Machinery, New York, NY, USA. p. 766–775. URL: https://doi.org/10.1145/3605573.3605613, doi:10.1145/3605573.3605613.

[16] Llama Team, 2024. The Llama 3 Herd of Models. URL: https://arxiv.org/abs/2407.21783, arXiv:2407.21783.

[17] NLLB Team, Costa-jussà, M.R., Cross, J., Durdle, O., Elbayad, M., Feng, F., Ghazvininejad, M., Gupta, U., Hassidim, A., et al., 2024. Scaling neural machine translation to 200 languages. Nature 630, 841–846. URL: https://www.nature.com/articles/s41586-024-07335-x, doi:10.1038/s41586-024-07335-x.

[18] Rajbhandari, S., Rasley, J., Ruwase, O., He, Y., 2020. Zero: Memory optimizations toward training trillion parameter models. URL: https://arxiv.org/abs/1910.02054, arXiv:1910.02054.

[19] Ramasesh, V.V., Lewkowycz, A., Dyer, E., 2022. Effect of scale on catastrophic forgetting in neural networks, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=GhVS8_yPeEa.

[20] Rausch, I., Bhaskar, B., Safont-Andreu, A., Ewetz, H., Kolovratnik, D., Oravecz, C., Runonen, M., 2025. Towards effective continued pre-training of EU institutional LLMs on EuroHPC supercomputers. Procedia Computer Science 255, 13–22. URL: https://www.sciencedirect.com/science/article/pii/S1877050925006179, doi:https://doi.org/10.1016/j.procs.2025.02.256.

[21] Singh, S., et al., 2025. Global MMLU: Understanding and addressing cultural and linguistic biases in multilingual evaluation. URL: https://arxiv.org/abs/2412.03304, arXiv:2412.03304.

[22] Steinberger, R., Ebrahim, M., Poulis, A., Carrasco-Benitez, M., Schlüter, P., Przybyszewski, M., Gilbro, S., 2014. An overview of the European Union's highly multilingual parallel corpora. Language Resources and Evaluation 48, 679–707. URL: https://doi.org/10.1007/s10579-014-9277-0, doi:10.1007/s10579-014-9277-0.

[23] Turisini, M., Amati, G., Cestari, M., 2023. Leonardo: A pan-European pre-exascale supercomputer for HPC and AI applications. URL: https://arxiv.org/abs/2307.16885, arXiv:2307.16885.

[24] Wen, K., Li, Z., Wang, J., Hall, D., Liang, P., Ma, T., 2024. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. URL: https://arxiv.org/abs/2410.05192, arXiv:2410.05192.

[25] Winata, G.I., Xie, L., Radhakrishnan, K., Wu, S., Jin, X., Cheng, P., Kulkarni, M., Preotiuc-Pietro, D., 2023. Overcoming catastrophic forgetting in massively multilingual continual learning. URL: https://arxiv.org/abs/2305.16252, arXiv:2305.16252.

[26] Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., Choi, Y., 2019. Hellaswag: Can a machine really finish your sentence? URL: https://arxiv.org/abs/1905.07830, arXiv:1905.07830.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 14–23

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

Scaling and performance aspects of the P3M method on multiple GPUs

Rodrigo A. C. Bartolomeu[a], René Halver[a], Godehard Sutmann[a,*]

[a] *Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich, Jülich, 52428, Germany*

Abstract

Long-range electrostatic interactions constitute a major computational bottleneck in particle-based simulations, particularly when periodic boundary conditions are employed. Mesh-based Ewald methods, such as the particle–particle particle–mesh (P3M) approach, reduce the computational complexity to $O(N \log N)$ but introduce challenges related to parallel scalability, communication overhead, and efficient utilization of modern heterogeneous architectures. While highly optimized implementations exist in established molecular dynamics packages, these are typically tightly coupled to full simulation frameworks, limiting their flexibility and reuse in emerging HPC applications.

In this work, we present a performance-portable library for electrostatic solvers, providing implementations of both classical Ewald and P3M methods targeting CPU and GPU architectures. The library is designed to enable efficient execution on multi-GPU systems while maintaining portability across heterogeneous platforms. We investigate scaling and performance characteristics of the P3M method, with particular focus on the FFT-based long-range component, which is known to limit scalability. Strong-scaling experiments demonstrate that small problem sizes are dominated by communication overhead, leading to reduced parallel efficiency, whereas larger systems achieve near-ideal scaling over a wide range of GPU counts.

Our results highlight the importance of balancing computation and communication in mesh-based electrostatics and demonstrate that portable, decoupled implementations can achieve competitive performance on modern HPC systems. The presented approach facilitates integration into diverse simulation workflows and provides a foundation for further optimization of long-range solvers in exascale computing environments.

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026

Keywords: Electrostatics; FFT; P3M; Scaling; GPU computing

1. Introduction

Particle-based simulation methods, such as Molecular Dynamics (MD) and Monte Carlo techniques, are well-established tools for investigating the structural and dynamic properties of nanoscale many-particle systems [1]. In

* Corresponding author.
E-mail address: g.sutmann@fz-juelich.de

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.015

these simulations, long-range interactions, most notably electrostatic interactions, have a significant importance [2]. However, accurately computing them remains a major computational bottleneck.

Directly evaluating all pairwise interactions scales as $O(N^2)$, which is prohibitively expensive for large systems. Furthermore, naive truncation of long-range interactions introduces artefacts, necessitating the use of methods that account for all particle contributions. This challenge is commonly faced by the use of periodic boundary conditions, which are employed to reduce the effects of the system's finite size and mimic bulk behaviour. Under these conditions, the system is replicated infinitely and the resulting Coulomb sum only converges conditionally due to the long-range nature of the interactions [3].

The classical Ewald summation method addresses this issue by decomposing the electrostatic potential into a short-range contribution, which is evaluated in real space, and a long-range contribution, which is treated in Fourier space [4]. This decomposition transforms the original, slowly converging series into two rapidly converging sums, enabling controlled approximations with tunable accuracy. With appropriate parameter optimisation, the computational complexity of the Ewald method can be reduced to $O(N^{3/2})$.

Further improvements are achieved by mesh-based Ewald methods, which accelerate the evaluation of the long-range contribution using fast Fourier transforms (FFTs), reducing the complexity to $O(N \log N)$. These approaches introduce a regular grid onto which particle properties are interpolated, enabling efficient computation in reciprocal space. Prominent examples include the particle–particle–particle–mesh (P3M) method [5], the particle–mesh Ewald (PME) method [6] and the smooth particle–mesh Ewald (SPME) method [2], which differ primarily in their interpolation schemes and force assignment strategies.

The increasing use of heterogeneous high-performance computing (HPC) architectures, particularly GPU-accelerated systems, has made mesh-based Ewald methods essential for large-scale molecular simulations. Their practical performance is strongly influenced by communication patterns, FFT scalability and memory access efficiency, despite their favourable asymptotic complexity. These are all critical on modern multi-GPU platforms. Consequently, substantial effort has been devoted to redesigning these algorithms for GPU-resident execution and distributed-memory parallelism. However, achieving efficient strong and weak scaling across multiple GPUs remains challenging, particularly for the long-range mesh-based portion, where global communication and FFT operations can create bottlenecks. These challenges have driven the development of optimised implementations in widely used MD packages and supporting libraries.

In this context, mesh-based Ewald methods form a core part of the majority of contemporary molecular dynamics packages, such as GROMACS (PME) [7], LAMMPS (P3M) [8], and AMBER (SPME) [2]. These implementations have been extensively optimised for heterogeneous HPC architectures and provide highly efficient solutions for large-scale simulations. However, as they are tightly integrated into full MD application frameworks, they are challenging to reuse outside of their respective software ecosystems and difficult to adopt in custom simulation workflows or emerging multiphysics and multiscale applications. By contrast, the work presented here focuses on developing a standalone, performance-portable library for electrostatic solvers. It provides implementations of both P3M and classical Ewald summation methods designed to run efficiently on CPUs and GPUs. By decoupling these algorithms from monolithic MD codes and targeting modern heterogeneous architectures, this approach enables flexible integration and facilitates algorithmic experimentation. It also addresses the current lack of reusable, scalable electrostatics libraries for HPC environments.

2. Theoretical background

Ewald techniques are able to compute the long-range interactions contributing to electrostatic forces for many systems. Within the scope of this work, we are particularly interested in methods that concern Coulomb point charges, as they are present in both single and multi-scale particle simulations, such as atomistic and coarse-grained techniques.

Previous descriptions and implementations of such methods can be found in the literature [3]. Parallel CPU implementations are available in the Scalable Fast Coulomb Solvers (ScaFaCoS) Library, available through the ScaFaCoS project website. However, most modern architectures make the use of accelerators to offload compute intensive algorithms. Therefore, within this section, we revisit the theoretical foundation to highlight the most important considerations related to GPU implementations of the Ewald Sum and P3M algorithms.

2.1. Ewald Summation Method

The electrostatic energy between two point charges decays inversely proportional to the distance with $(1/r)$. In a system with N point charges, each with a nominal charge q_i in a cubic periodic three-dimensional space with box length L at positions $\mathbf{r}_i$ is given by:

$$E = \frac{1}{2} \sum_{i,j=1}^{N} \sum_{\mathbf{n}\in\mathbb{Z}}^{'} \frac{q_i q_j}{|\mathbf{r}_{ij} + \mathbf{n}L|}, \tag{1}$$

with $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and excluding the contribution for $i = j$ for $\mathbf{n} = 0$.

The potential in Equation 1 presents two significant issues: it decays slowly at large distances and varies abruptly at small distances. If only one of these issues were present, it would be relatively easy to address, as a short-range potential could be treated by a simple cutoff. This problem is addressed by the means of splitting into short and long-range contributions making use of the identity:

$$\frac{1}{r} = \frac{f(r)}{r} + \frac{1 - f(r)}{r}, \tag{2}$$

therefore, giving the possibility of circumventing the slowly decaying long range contribution of the Coulomb potential have with the straightforward summation. Here, the first term of the RHS, $f(r)/r$, is a good approximation for the short-range contribution to the total energy. The function $f(r)$ is usually chosen as a smoothly decaying function, where its decay and support can be controlled by a parameter. For the present type of application most often the complementary error function is chosen for $f(r)$. For a proper choice of parameters of the complementary error function, the second term $1 - f(r)/r$, varies slowly for all values of r, allowing its Fourier transform to be accurately represented by a limited number of $\mathbf{k}$ vectors, where $|\mathbf{k}| < k_{max}$. This enables an efficient calculation of the contribution to the total electrostatic potential in reciprocal space.

Due to the linearity of the field equations, the sums of both aforementioned terms can be performed without significant loss of accuracy. Different contributions can be split into four terms. These can be computed in parallel and are given by:

$$\begin{aligned}
E^{(r)} &= \tfrac{1}{2} \textstyle\sum_{i,j=1}^{N} \sum_{\mathbf{m}\in\mathbb{Z}}' q_i q_j \frac{\operatorname{erfc}(\alpha|\mathbf{r}_{ij}+\mathbf{m}L|}{|\mathbf{r}_{ij}+\mathbf{m}L|}, \\
E^{(k)} &= \tfrac{1}{2L^3} \textstyle\sum_{\mathbf{k}\neq 0} \frac{4\pi}{k^2} e^{-k^2/4\alpha^2} |\tilde{\rho}(\mathbf{k})|^2, \\
E^{(s)} &= -\tfrac{\alpha}{\sqrt{\pi}} \textstyle\sum_i q_i^2, \\
E^{(d)} &= \tfrac{2\pi}{(1+2\epsilon')L^3} \left(\textstyle\sum_i q_i \mathbf{r}_i\right)^2,
\end{aligned} \tag{3}$$

where, $E^{(r)}$ is the real space contribution, $E^{(k)}$ is the reciprocal space, $E^{(s)}$ is the self-energy, $E^{(d)}$ is the dipole correction, the splitting function f is the complementary error function $\operatorname{erfc}(r) = \frac{2}{\sqrt{\pi}} \int_r^{\infty} e^{-t^2} \mathrm{d}t$, and the Fourier transformed density $\tilde{\rho}(\mathbf{k})$ is:

$$\tilde{\rho}(\mathbf{k}) = \int_{V_b} \mathrm{d}^3 r \rho(\mathbf{r}) e^{-i\mathbf{k}\cdot\mathbf{r}_j}. \tag{4}$$

The parameter α, referred to here as the Ewald parameter, controls the weights between the contributions from real and reciprocal space, allowing the time spent calculating each contribution to be tuned. Given that the complexity of solving particle-particle interactions in real space and the scaling behaviour of algorithms for Fourier computations differ, it is possible to find a compromise in the choices of parameters for the real and reciprocal spaces and the splitting parameter, leading to complexity of $O(N^{3/2})$. The ability to control this splitting parameter has become increasingly relevant as modular HPC systems have evolved, in which each method can be computed using different resources.

In this context, for a given level of accuracy, two parameters can be chosen from the following set: the real-space cutoff (r_{cut}), the maximum number of k modes (k_{max}) and alpha. Therefore, if the implementation supports the calculation of different contributions on several platforms or architectures, one can optimise the use of resources on modern supercomputers. Additionally, the GPGPU parallelisation paradigm enables the calculation of k-modes per particle to be performed in a more efficient manner than CPUs.

2.2. Particle-Particle Particle-Mesh Method

The P3M method maps the system charges onto a mesh, enabling the necessary Fourier transformations to be efficiently performed using a Fast Fourier Transform algorithm. Simultaneously, the basic free space Coulomb Green's function ($4\pi/k^2$) is modified to ensure that the outcome of the mesh calculation closely approximates the continuum solution. The initial step involves generating a mesh-based charge density, denoted as ρ_{M}, which is defined at the mesh points ($\mathbf{x}_p$) given by:

$$\rho_{\mathrm{M}}(\mathbf{r}_p) = \frac{1}{h^3} \int_V \mathrm{d}\mathbf{r}\, W(x_p - x) W(y_p - y) W(z_p - z) \rho(\mathbf{r}), \tag{5}$$

where h represents the mesh spacing. The charge assignment function, W, is categorized by its order, P, which indicates the number of mesh points per coordinate direction over which each charge is distributed. Specifically, W is chosen to be a cardinal B-spline, a piecewise polynomial function with a weight of one. The order P corresponds to the number of sections that comprise the function and its representation in Fourier space is:

$$\widetilde{W}(\mathbf{k}) = h^3 \left(\frac{\sin\left(\frac{1}{2} k_x h\right)}{\frac{1}{2} k_x h} \frac{\sin\left(\frac{1}{2} k_y h\right)}{\frac{1}{2} k_y h} \frac{\sin\left(\frac{1}{2} k_z h\right)}{\frac{1}{2} k_z h} \right)^P . \tag{6}$$

Next, we calculate the mesh-based electric field $\mathbf{E}(\mathbf{r}_p)$ in the second step. The electric field is essentially the derivative of the electrostatic potential. However, there are several methods to implement differentiation on a lattice [9]. For this purpose, we describe the $i\mathbf{k}$ differentiation method, which involves multiplying the Fourier-transformed potential by $i\mathbf{k}$, and has been shown to be the most accurate. Using this approach, the electric field $\mathbf{E}(\mathbf{r}_p)$ can be expressed as:

$$\mathbf{E}(\mathbf{r}_p) = \overleftarrow{\mathrm{FFT}} \left[-i\mathbf{k} \cdot \overrightarrow{\mathrm{FFT}}[\rho_{\mathrm{M}}] \cdot \hat{G}_{\mathrm{opt}} \right] (\mathbf{r}_p), \tag{7}$$

where $\overleftarrow{\mathrm{FFT}}$ and $\overrightarrow{\mathrm{FFT}}$ are the backward and forward FFT respectively, and $\hat{G}_{\mathrm{opt}}$ is the modified optimal influence function, derived by Hockney and Eastwood [5] which is written as:

$$\hat{G}_{\mathrm{opt}}(\mathbf{k}) = \frac{\widetilde{\mathbf{D}}(\mathbf{k}) \cdot \sum_{\mathbf{m} \in \mathbb{Z}^3} \widetilde{U}^2 \left(\mathbf{k} + \frac{2\pi}{h} \mathbf{m} \right) \widetilde{\mathbf{R}} \left(\mathbf{k} + \frac{2\pi}{h} \right) \mathbf{m}}{\left| \widetilde{\mathbf{D}}(\mathbf{k}) \right|^2 \left[\sum_{\mathbf{m} \in \mathbb{Z}^3} \widetilde{U}^2 \left(\mathbf{k} + \frac{2\pi}{h} \mathbf{m} \right) \right]^2} \tag{8}$$

where

$$\widetilde{\mathbf{R}}(\mathbf{k}) = -i\mathbf{k}\frac{4\pi}{k^2}e^{-k^2/4\alpha^2} \qquad \text{and} \qquad \widetilde{\mathbf{U}}(\mathbf{k}) = \widetilde{W}(\mathbf{k})/h^3.$$

The number of mesh points, denoted as N_{M}, is defined by L/h along each direction (Equation 5). In theory, N_{M} can be chosen arbitrarily to achieve desired accuracy. However, practical constraints arise from the efficiency of FFT libraries (such as FFTW), which support optimized algorithms for computing factors of small prime numbers. A useful mesh size formula, taking into account these constraints and the efficiency of FFTs, is:

$$N_{\mathrm{M}} = 2^a \times 3^b \times 5^c \times 7^d \times 11^e, \tag{9}$$

or a power of two, as this optimization facilitates maximum efficiency in FFT computation. While finding the exact minimum mesh size necessary to meet accuracy requirements in terms of target k points is possible, it is not guaranteed to be optimal for performance. Therefore, it may be beneficial to explore nearby mesh sizes that optimize time to solution, even if this incurs slightly higher computational costs in other functions, such as charge assignment.

3. Methodology

3.1. Implementation Considerations

The presented performance-portable implementation of the P3M presented here uses the Cabana library [10] for data structures and communication algorithms, as well as providing method for the administration of the necessary grids and interfaces to the heFFTe library for the FFT computations. The Cabana library was developed within the Co-Design Center for Particle Applications (CoPA) within the Exascale Computing Project (ECP). Cabana is built on top of Kokkos [11], as well as additional, optional library dependencies. Sharing the same design philosophy, the performance-portable electrostatic library has dependencies (see Fig 1.Foe example: MPI for parallel distributed memory execution or heFFTe [12] for FFT computations and ALL [13] for load balancing, in order to target multiple backends.

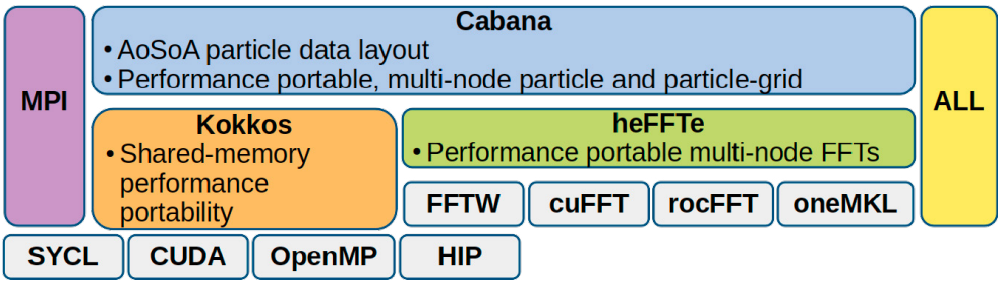


Fig. 1. Dependencies and supported backends for the current implementation of the performance portable electrostatic library.

The fact that the standard portable model for shared memory parallelism of Cabana is Kokkos, provides an intra-node parallelization model. Kokkos is suitable for particle based simulations, and the optimized conversion of data structure through the use of so-called *Views* allows for the use of optimal memory access patterns required by different parts of the electrostatics methods and many body simulations [14]. Previous work has shown, that Kokkos as a parallelization layer is suited for particle [14] and mesh-based [15] algorithms.

3.2. Benchmark setup

To explore the multi-GPU scaling capabilities of the particle-mesh based solver we have conducted computational experiments on the MareNostrum 5 Accelerated partition. Nodes in this partition are equipped with two Intel Xeon Platinum 8460Y CPUs and four NVIDIA Hopper H100 64GB HBM2 GPUs each, while providing 512 GB of 4800MHz DDR5 memory, and a ConnectX-7 NDR200 InfiniBand network connection with 800 Gb/s bandwidth per core.

For this work all libraries and executables were compiled with `-O2` optimization, having the OpenMP backend enabled for host and having the CUDA backend enabled for device execution. The used compilers and libraries versions were:

- GCC 13.2.0
- CUDA 12.8
- Kokkos 4.7.2
- Cabana 0.8.0
- heFFTe 2.4.1 using the cuFFT backend

The compilers were provided by the respective modules on the test system (March 2026).

3.3. System and algorithm parameters

Knowing that the long-range contribution is the more computationally more expensive part of the electrostatic calculation as discussed in [3], our benchmarks focused on the evaluation of the elapsed compute time spent in the computation of the *k-space* contribution. To fairly access the impact of FFT grid sizes, the real space cutoff, r_{cut}, was fixed at 4.6172 and reciprocal space parameters were tuned to achieve an absolute accuracy of 10^{-6}. The model Cloud Wall system (Fig. 2) containing 300 particles in the base cell was replicated periodically in x , y and z directions 8, 16, 32, and 64 times, referred to as the replication size or number from here on.

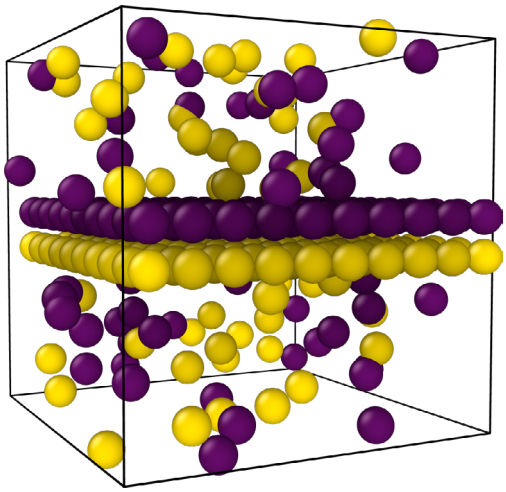


Fig. 2. Visualization of the Cloud Wall system (300 charges). The configuration was artificially created by Arnold *et al.* [3] to possess a strong long-range component. Each side has length of 10 arbitrary units.

This distribution of charged particles creates an artificial dipole. If the system is replicated, planes of opposing charges are formed. The same behaviour also occurs due to the periodic boundary conditions. Therefore, the long-range contribution is never negligible, making this configuration valuable for benchmarking the solver's accuracy.

To ensure that the predicted accuracy of the P3M method [9, 16] was achieved, results for systems up to 1228800 charges were validated through the root mean square error using a highly accurate Ewald summation tuned to yield

values close to machine precision as reference. Above this number of charges the computation of the Ewald method is considered to be prohibitively computationally expensive. As an example, for the validation of $16 \times 16 \times 16$ periodic replicas the following numbers were observed in one node using 1 GPU for long-range:

```
tuned Ewald values:
        r_max:  6.30579
        k_max (1D / 3D):        240 / 240 240 240
        alpha:  0.871125
        estimated RMS error in forces: 4.84886e-14
tuned P3M values:
        r_max:  4.6172
        FFT grid size (3D):     512 512 512
        charge assignment order:        7
        alpha:  0.846208
        h * alpha:      0.26444
        estimated RMS error in forces: 3.15284e-07
RMS potential :  4.02801676349598e-07
RMS fields:  7.64805475718963e-08 8.4353410644898e-08 3.97391771302327e-07
Time spent in run [Ewald] (avg, min, max): 2086.37708675 s 2085.901671 s 2086.535621 s
Time spent in run [P3M] (avg, min, max): 0.5853885 s 0.585388 s 0.585389 s
```

Where RMS indicates the absolute root mean square error with regard to per particle energy and fields. It can be seen, that the time to solution for the P3M solver is magnitudes faster than the Ewald solver ($\approx 2000s$ for the Ewald solver to $\approx 0.5s$ for the P3M solver).

Grid sizes, system details and resources used for strong scaling experiments are presented in Table 1. An important observation is that due to limitation in memory caused by the large number of FFT grid points, the larger systems can not be computed using a single computation node.

Table 1. System sizes and reciprocal space parameters of the P3M solver with fixed parameters: charge assignment order = 7, $\alpha = 0.846208$, and $r_{\text{cut}} = 4.6172$.

Replication	No. of charges	Grid size	No. of nodes	No. of GPUs for k-space
$8 \times 8 \times 8$	153600	256	1, 2, 4 and 8	1;2;3;4;6;8;12;16;24
$16 \times 16 \times 16$	1228800	512	1, 2, 4 and 8	1;2;3;4;6;8;12;16;24
$32 \times 32 \times 32$	9830400	1024	16 and 32	4;8;16;32;48;64;80;96;112;120;124
$64 \times 64 \times 64$	78643200	2048	16 and 32	32;48;64;96

The presented implementation of the P3M solver can shift MPI ranks between sub-communicators, which are used to compute the k-space and real-space contributions independently of each other. To achieve the following results, the number of processes in the k-space part was varied. A total of 64 GPUs were used for replication sizes of 8 and 16, and 128 GPUs were used for larger replication numbers (32 and 64).

4. Results

A statistical evaluation of the variation of multiple time steps using the static configuration was performed for the $32 \times 32 \times 32$ system for the presentation and post-processed P3M run times. The standard deviation remained below 3% for all numbers of GPUs. Once these values were represented graphically, it was found that they were smaller than the symbols. Therefore, error bars are omitted for clarity.

As the current study focuses on the FFT benchmark, the configuration is considered to be representative of the timings, despite the inhomogeneity of the system (which primarily affects the charge assignment routines due to load imbalance). As the library is designed to receive particle data from the calling program at each time step, particle movement is not considered to have a significant impact the current benchmarks.

Fig. 3 shows the normalised benchmark results for the Cloud Wall system with different replication sizes, as described in Section 3.3. The presented times are normalised to the smallest replication number (8×8×8) to demonstrate the similarity in scaling behaviour.

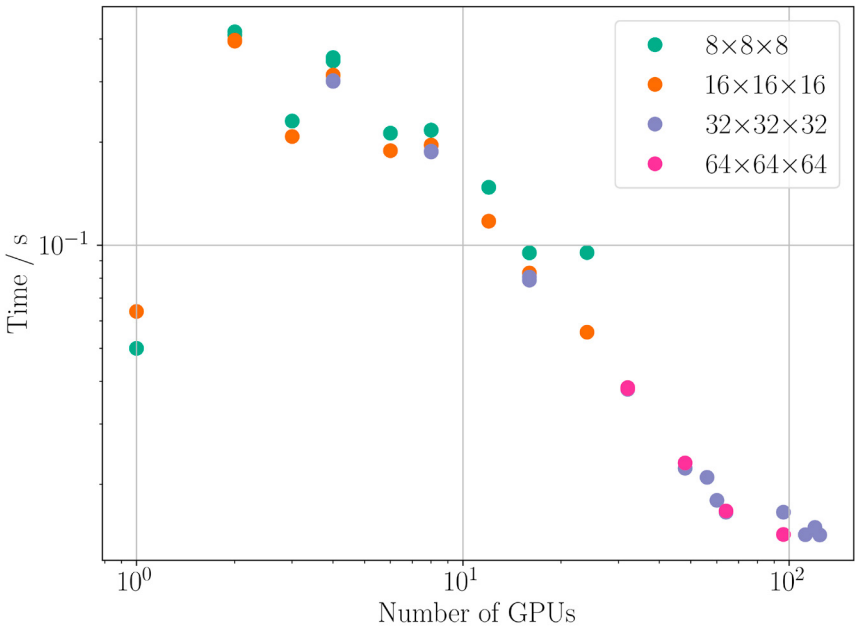


Fig. 3. Comparing strong-scaling benchmark results for different system replication numbers (8 to 64) on 1 to 128 GPUs (1 to 32 nodes) on MareNostrum5 accelerated partition. Times normalized to the smallest replication number.

The strong scaling behaviour is similar for different replication numbers, depending on the number of processes, but differs in terms of the length of the tail, depending on the total system size and thus the amount of computational work rerquired to compute fields and energy.

A clear parallel overhead is evident when using two or more GPUs, as the runtime shows a significant increase when using two GPUs for any given replication number. This increase in computation time can be attributed to the parallel computation of the FFT. FFTs are solved so efficiently that any necessary communication creates significant parallel overhead. Furthermore, the results show that this parallel overhead scales with regard to computation time on two GPUs.

In the conducted benchmarks, the k-space part was computationally more demanding than the short-range computation, regardless of the split between the sub-communicators used.

A more in-depth analysis can be conducted once speed-up and parallel efficiencies have been calculated. The results in Fig.4 highlight distinct scaling regimes depending on the problem size. For the smallest systems ($8 \times 8 \times 8$ and $16 \times 16 \times 16$), the speedup remains below ideal value and quickly saturates as the number of GPUs increases. This is also reflected in the parallel efficiency, which drops sharply below 10% already at moderate GPU counts. The dominant factor here is parallel overhead: communication and synchronization costs, particularly those associated with the FFT-based long-range solver, outweigh the relatively small computational workload.

As a result, adding more GPUs does not translate into proportional performance gains, and in some cases even leads to diminishing returns. This behavior is typical for strong scaling of small problem sizes on modern GPU clusters, where latency and communication costs become the limiting factors.

In contrast, larger systems ($32 \times 32 \times 32$ and $64 \times 64 \times 64$) exhibit significantly improved scaling behavior. The speedup follows the ideal trend much more closely over a wide range of GPU counts, and the parallel efficiency remains close to unity for intermediate configurations, even exceeding it slightly in some cases likely due to cache and load-balancing effects. Only at higher GPU counts does a gradual efficiency drop become visible, indicating the onset of communication-dominated execution. These results demonstrate that the computational intensity of the

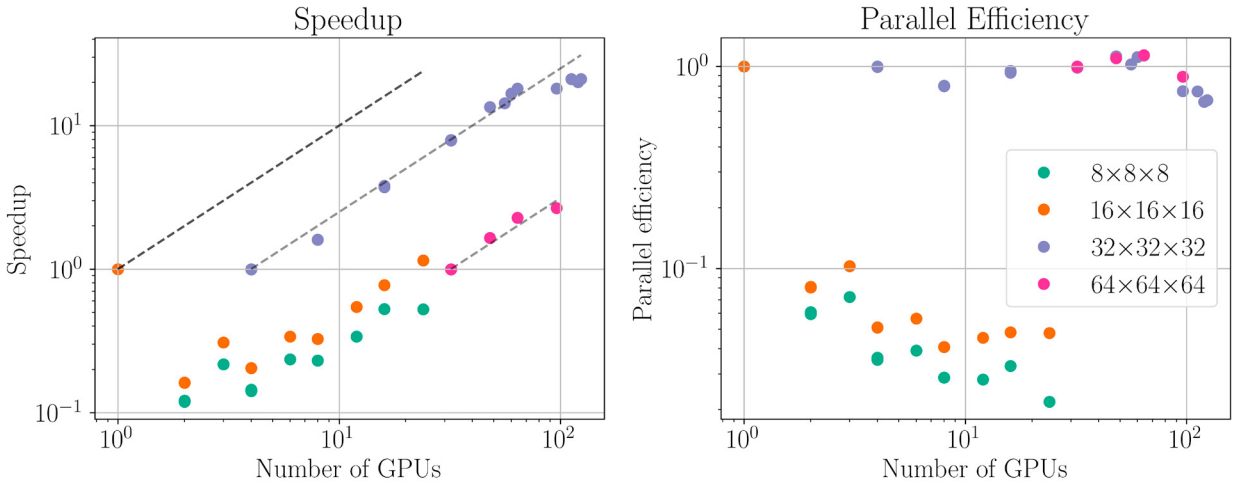


Fig. 4. Speedup (left) and Parallel efficiency (right) of strong-scaling benchmark results for different system replication numbers (8 to 64) on 1 to 128 GPUs (1 to 32 nodes) on MareNostrum5 accelerated partition. Dashed lines represent the ideal speedup taking as reference the smallest amount of resources that could compute the system size.

larger problem sizes is sufficient to amortize the communication overhead, allowing the GPU-parallel implementation to achieve high efficiency.

Overall, Fig. 4 confirms that the mesh-based electrostatics solver can be scaled up effectively on multi-GPU systems, provided that the problem size is large enough to balance computation and communication. Nevertheless, the FFT-based long-range component remains the primary scalability bottleneck.

5. Conclusions

The results show that the P3M Coulomb solver implementation presented is capable of running on the accelerated part of the MareNostrum5 system. As discussed previously, the following conclusions can be drawn regarding the sensible use of the implementation.

Firstly, if the FFT grid required to achieve the necessary solution accuracy in the fields is small enough to fit on a single GPU, it should be executed on a single GPU due to the significant parallel processing overhead created when using two or more GPUs. Secondly, if the grid does not fit on a single GPU, it can be seen that the k-space part of the solver scales with regard to the smallest number of GPUs necessary to compute the FFT. Therefore, if the FFT part does not fit on a single GPU, it may be sensible to use more GPUs, since depending on the grid size, the k-space part benefits from the scaling.

Further analysis is needed to better define the parameters indicating when large-scale execution is sensible and when smaller-scale execution is preferable. This is because there may be a crossover point with regard to parallel overhead, at which point parallel execution may be beneficial regardless of purely memory size restrictions. With this in mind, for system sizes that fall in between the optimal scaling shown in this study for large systems, and the large parallel overhead introduced by FFT communication it might be better to explore parallel replicas, umbrella sampling or task farming depending on each user's application.

Furthermore, in this study, we limited the work assigned to the real space. Increasing the real-space splitting parameter could considerably reduce the work required to achieve the desired accuracy through k-space partitioning.

Acknowledgements

We acknowledge EuroHPC JU for awarding the project ID **EHPC-DEV-2025D11-078** access to MareNostrum5 GPP and ACC partitions hosted by Barcelona Super Computing Centre and ee thank the Jülich Supercomputing

Centre, Forschungszentrum Jülich for access to the JURECA-DC Evaluation Platform and JUWELS Booster. R. A. C. B. and G. S. acknowledge funding by the European Union; this work has received funding from the European High Performance Computing Joint Undertaking under the grant agreement 101093169. R. H. and G. S. acknowledge funds by the Ministry of Culture and Science (MKW) of the state of North-Rhine-Westphalia through funding of the Gauss Centre for Supercomputing (GCS).

References

[1] M. P. Allen, D. J. Tildesley, Computer Simulation of Liquids, Oxford University Press, 2017. doi:10.1093/oso/9780198803195.001.0001.

[2] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, A smooth particle mesh ewald method, The Journal of Chemical Physics 103 (19) (1995) 8577–8593. doi:10.1063/1.470117.

[3] A. Arnold, F. Fahrenberger, C. Holm, O. Lenz, M. Bolten, H. Dachsel, R. Halver, I. Kabadshow, F. Gähler, F. Heber, J. Iseringhausen, M. Hofmann, M. Pippig, D. Potts, G. Sutmann, Comparison of scalable fast methods for long-range interactions, Phys. Rev. E 88 (2013) 063308. doi:10.1103/PhysRevE.88.063308.

[4] P. P. Ewald, Die berechnung optischer und elektrostatischer gitterpotentiale, Annalen der Physik 369 (3) (1921) 253–287. doi:10.1002/andp.19213690304.

[5] R. W. Hockney, J. W. Eastwood, Computer simulation using particles, crc Press, 2021.

[6] T. Darden, D. York, L. Pedersen, Particle mesh ewald: An nlog(n) method for ewald sums in large systems, The Journal of Chemical Physics 98 (12) (1993) 10089–10092. doi:10.1063/1.464397.

[7] S. Páll, A. Zhmurov, P. Bauer, M. Abraham, M. Lundborg, A. Gray, B. Hess, E. Lindahl, Heterogeneous parallelization and acceleration of molecular dynamics simulations in gromacs, The Journal of Chemical Physics 153 (13) (2020) 134110. doi:10.1063/5.0018516.

[8] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, S. J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Comp. Phys. Comm. 271 (2022) 108171. doi:10.1016/j.cpc.2021.108171.

[9] M. Deserno, C. Holm, How to mesh up ewald sums. i. a theoretical and numerical comparison of various particle mesh routines, The Journal of Chemical Physics 109 (18) (1998) 7678–7693. doi:10.1063/1.477414.

[10] S. Slattery, S. T. Reeve, C. Junghans, D. Lebrun-Grandié, R. Bird, G. Chen, S. Fogerty, Y. Qiu, S. Schulz, A. Scheinberg, A. Isner, K. Chong, S. Moore, T. Germann, J. Belak, S. Mniszewski, Cabana: A performance portable library for particle-based simulations, Journal of Open Source Software 7 (72) (2022) 4115. doi:10.21105/joss.04115.

[11] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, J. Wilke, Kokkos 3: Programming model extensions for the exascale era, IEEE Transactions on Parallel and Distributed Systems 33 (4) (2022) 805–817. doi:10.1109/TPDS.2021.3097283.

[12] A. Ayala, S. Tomov, A. Haidar, J. Dongarra, heffte: Highly efficient fft for exascale, in: Computational Science – ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part I, Springer-Verlag, Berlin, Heidelberg, 2020, p. 262–275. doi:10.1007/978-3-030-50371-0_19.

[13] R. Halver, S. Schulz, G. Sutmann, ALL - A loadbalancing library, C++ / Fortran library, https://gitlab.version.fz-juelich.de/SLMS/loadbalancing/-/releases.
URL https://gitlab.version.fz-juelich.de/SLMS/loadbalancing/-/releases

[14] R. A. Bartolomeu, R. Halver, J. H. Meinke, G. Sutmann, Effect of implementations of the n-body problem on the performance and portability across gpu vendors, Future Generation Computer Systems 169 (2025) 107802. doi:10.1016/j.future.2025.107802.

[15] R. Halver, C. Junghans, G. Sutmann, Kokkos-based implementation of mpcd on heterogeneous nodes, in: R. Wyrzykowski, J. Dongarra, E. Deelman, K. Karczewski (Eds.), Parallel Processing and Applied Mathematics, Springer International Publishing, Cham, 2023, pp. 3–13. doi:10.1007/978-3-031-30445-3_1.

[16] M. Deserno, C. Holm, How to mesh up ewald sums. ii. an accurate error estimate for the particle–particle–particle-mesh algorithm, The Journal of Chemical Physics 109 (18) (1998) 7694–7701. doi:10.1063/1.477415.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 24–33

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Fourth EuroHPC user day

Future Requirements of Lattice Field Theory Calculations on European High-Performance Computing Facilities

Gert Aarts[a], Gunnar Bali[b], Jacob Finkenrath[c], Stefan Krieg[d,e], Antonio Rago[f,*], Carsten Urbach[e], Hartmut Wittig[g]

[a] *Centre for Quantum Fields and Gravity, Department of Physics, Swansea University, Swansea, SA2 8PP, United Kingdom*
[b] *Institut für Theoretische Physik, Universität Regensburg, 93040 Regensburg, Germany*
[c] *Department of Physics, Bergische Universität Wuppertal, 42119 Wuppertal, Germany*
[d] *Jülich Supercomputing Centre (JSC), Center for Advanced Simulation and Analytics (CASA), Forschungszentrum Jülich, 54245 Jülich, Germany*
[e] *Helmholtz-Institut für Strahlen- und Kernphysik, Rheinische Friedrich-Wilhelms-Universität Bonn, Nussallee 14-16, 53115 Bonn, Germany*
[f] *Quantum Theory Center (QTC), IMADA, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark*
[g] *PRISMA++ Cluster of Excellence, Institut für Kernphysik and Helmholtz-Institut Mainz, Johannes Gutenberg-Universität Mainz, 55099 Mainz, Germany*

Abstract

Lattice field theory provides a first-principles framework for studying properties of strongly interacting quantum field theories in elementary particle physics. Researchers in lattice field theory are also among the largest and most efficient users of high-performance computing resources in fundamental science. In this contribution, we outline the computational profile of lattice QCD, from gauge-field generation to large-scale measurements, and discuss the main hardware, software, and human resource requirements needed to sustain progress on current and future European HPC infrastructures.

Keywords: Lattice field theory, QCD, Computational cost, Efficiency, HPC, Exascale

1. Introduction

The Standard Model of particle physics represents one of the most successful theoretical frameworks in modern science. Major international experiments carried out over several decades have established with remarkable precision the particle content and interactions of the electroweak and strong sectors. Measurements at facilities such as LEP, the Tevatron, and the Large Hadron Collider have confirmed the gauge structure of the Standard Model and determined many of its parameters with high accuracy.

Within this framework, the strong interaction described by Quantum Chromodynamics (QCD) plays a central role. QCD governs the dynamics of quarks and gluons and is responsible for the formation of hadrons, including baryons, the particles that make up most of the visible matter in the universe. A key feature of this theory is that the interaction becomes stronger at low energies. At the scale relevant for hadrons, the theory is therefore strongly coupled, meaning that standard perturbative techniques are no longer applicable. Understanding the properties of strongly interacting matter consequently requires genuinely non-perturbative approaches.

Lattice field theory provides such a framework. In this approach space and time are discretised on a finite grid, allowing the quantum field theory to be formulated in a way that can be evaluated numerically. Physical observables are

*Corresponding author. Tel.: +4565509258
Email address: rago@qtc.sdu.dk (Antonio Rago)

1877-0509
10.1016/j.procs.2026.08.016

computed by sampling gauge field configurations using Monte Carlo methods. This formulation enables calculations directly from the fundamental theory with systematically improvable uncertainties. Although originally developed for QCD, the method can be applied to any quantum field theory that admits a Euclidean path integral formulation and therefore has become a powerful tool for studying any strongly interacting field theory.

Lattice Quantum Chromodynamics represents the most mature realisation of this approach. Over the past decades lattice QCD has produced a number of major achievements. One of the earliest successes was the quantitative demonstration of colour confinement through the computation of the potential between a heavy quark and an antiquark, showing the emergence of a constant attractive force at large distances. Lattice calculations have also enabled precise determinations of the hadron spectrum, reproducing the masses of light hadrons with percent-level agreement with experimental measurements.

In recent years, lattice QCD has become an essential tool for determining fundamental parameters of the Standard Model, including the quark masses and the strong coupling constant. Its scope has expanded to increasingly complex observables, such as nucleon structure, hadron scattering amplitudes, and electroweak decay matrix elements. Lattice calculations also provide crucial theoretical input for precision tests of the Standard Model. A notable example is the determination of the hadronic contributions to the anomalous magnetic moment of the muon $(g-2)$, a quantity that showed a tension between experimental measurements and theoretical predictions [1]. Improving the precision of hadronic contributions is therefore essential for interpreting this discrepancy and assessing possible indications of physics beyond the Standard Model (BSM) [2]. Including temperature in simulations is straightforward and yields results relevant for the early universe and for relativistic heavy-ion collisions, e.g., at the Large Hadron Collider [3]. The community has embraced this challenge in line with the priorities identified by the European Strategy for Particle Physics[1] and the Nuclear Physics Collaboration Committee (NuPECC) Long Range Plan[2].

The challenges encountered are not limited to QCD. Many extensions of the Standard Model predict additional strongly interacting sectors, often motivated by attempts to explain phenomena such as electroweak symmetry breaking, dark matter, or the hierarchy problem. These BSM theories frequently involve new gauge dynamics whose low-energy behaviour is inherently non-perturbative. In summary, lattice field theory has become a mature computational discipline for obtaining quantitative predictions for strongly interacting systems directly from the underlying quantum field theory, while also providing a versatile first-principles framework for interpreting present experimental data and exploring possible new physics scenarios beyond the Standard Model.

1.1. Lattice formulation

The lattice formulation replaces continuous space–time with a discrete grid characterised by a lattice spacing a and a finite physical extent L. The lattice spacing acts as an ultraviolet regulator of the theory, while the finite volume provides an infrared cutoff.

Physical observables are computed as expectation values of operators defined on the lattice. In practice, these expectation values are estimated by averaging over ensembles of gauge-field configurations. Each configuration can be interpreted as a snapshot of the quantum fields over the entire space–time lattice. Since quantum fluctuations are intrinsic to the theory, a single configuration does not carry direct physical meaning. Instead, reliable predictions emerge only after sampling a sufficiently large number of configurations distributed according to the probability measure defined by the lattice action.

Markov-chain Monte Carlo algorithms produce a sequence of such field configurations. Observables are then evaluated on each configuration ("measurements") and averaged over the ensemble. The statistical uncertainty associated with this procedure decreases as more configurations are included, allowing one to assign predictive errors to the final results. Controlling these statistical uncertainties requires careful treatment of correlations between configurations and different observables on a given ensemble.

In addition to statistical uncertainties, lattice calculations must address systematic effects introduced by the discretisation of space–time. The physical theory corresponds to the continuum limit, which is obtained by extrapolating results computed at several lattice spacings towards $a \to 0$, a procedure called "continuum limit". Achieving control over this extrapolation is challenging and requires simulations at multiple lattice spacings together with careful analysis of discretisation effects. At the same time, the lattice volume must be large enough to suppress finite-volume

[1] https://europeanstrategy.cern/
[2] https://www.nupecc.org/

artefacts, and the parameters of the theory must be tuned to reproduce some observables of reference.

Reducing the lattice spacing, increasing the lattice volume, and approaching physical quark masses all lead to rapidly increasing computational costs. As a result, lattice QCD simulations require extremely large computational resources and have historically been among the most demanding applications in high-performance computing.

1.2. Lattice field theory community

A key feature of lattice field theory is that physical results, once extrapolated to the continuum limit, do not depend on the specific lattice formulation used. Different discretisations of the theory may differ at finite lattice spacing, but they all converge to the same continuum physics. This property allows for independent calculations of the same observables using different lattice formulations. In practice, this provides an important and, in fact, scientifically necessary form of cross-validation of the method: results obtained by different groups using different discretisations can be compared and tested against each other. In this respect, lattice QCD resembles large experimental programmes such as those at CERN, where multiple experiments independently measure the same quantities using different detectors to ensure robustness and reliability of the results.

The choice of a particular lattice formulation often shapes the organisation of the research collaborations that carry out the simulations. Over time, communities have formed around specific discretisations of the theory, together with the software frameworks and algorithmic strategies required to implement them efficiently. These collaborations typically involve multiple research institutions and national computing infrastructures, pooling expertise in physics, numerical algorithms, and high-performance computing.

The lattice field theory community, therefore, operates through large collaborative projects that generate and analyse ensembles of gauge-field configurations. Producing such ensembles requires extensive and sustained access to high-performance computing resources and is frequently coordinated across several computing centres. Once generated, these configurations become long-lived scientific assets within an open-science framework, shared across collaborations, and reused by multiple groups for diverse physics analyses, retaining and even increasing their value over more than a decade.

Because the same ensembles support many independent studies, the community has historically developed a strongly interconnected structure. This model maximises the scientific return of large computational investments and has fostered a culture of data sharing, common standards, and joint development of software tools that underpin modern lattice field theory research. It also enables the community to scrutinise results obtained with different lattice formulations and to provide validated input to the wider community, in particular flavour physics, where lattice QCD supplies the non-perturbative hadronic quantities needed to interpret precision measurements from LHCb, Belle II, BESIII, kaon experiments, and related programmes. Decay constants, form factors, bag parameters, and mixing matrix elements enter directly into CKM phenomenology, unitarity tests, and searches for physics beyond the Standard Model. The Flavour Lattice Averaging Group (FLAG) provides a central example of this community-wide validation process: it reviews lattice results, applies agreed quality criteria, and provides reference averages or estimates for the wider particle-physics community [4, 5, 6].

1.3. EuroLFT

The European lattice field theory community has recently established the EuroLFT initiative (https://eurolft.github.io/) as a platform to represent the interests and needs of the European lattice community towards European institutions. EuroLFT acts as a point of reference for institutions seeking to engage with the lattice field theory community and facilitates dialogue on topics relevant to research, computing infrastructure, and long-term scientific strategy. Similar initiatives exist in other regions of the world; for example, in the United States the lattice community is organised through the USQCD collaboration, which coordinates community activities and interactions with national laboratories and funding agencies (https://www.usqcd.org).

EuroLFT aims to strengthen connections within the European lattice community and to improve interactions between research groups, computing centres, and policy bodies. By serving as a representative forum for the community, it articulates the computational requirements of the field, provides a collective perspective on future developments in computing architectures, and ensures that the specific needs of lattice field theory are visible in broader strategic discussions concerning high-performance computing and scientific data infrastructures in Europe.

1.4. International Lattice Data Grid

The lattice field theory community has historically embraced the principles of open science. A central element of this approach is the sharing of gauge-field configurations, which represent a major fraction of the consumed computer time of large-scale lattice simulations. Once generated, these configurations can be reused by many research groups to study a wide range of physical observables. This practice greatly amplifies the scientific return of large computational investments and is consistent with the culture of collaboration and data sharing across the field.

To support this model, the community established the International Lattice Data Grid (ILDG) [7, 8, 9, 10], a distributed service to share gauge-field ensembles produced by lattice collaborations worldwide. The ILDG connects multiple regional storage nodes through a common metadata catalogue, enabling researchers to discover, access, and reuse configurations generated on major high-performance computing systems. The ILDG represented, and still represents, an implementation of FAIR principles [11] for the precious lattice gauge-field ensembles.

More recently, the European part of the community has launched the ILDG 2.0 initiative [12], aimed at revitalising this infrastructure by adapting it to modern data management practices and evolving computing environments. This effort seeks to improve interoperability with contemporary storage technologies and data services while preserving the core mission of enabling open access to lattice simulation data. In addition, ILDG 2.0 strives to extend its service to additional data types.

2. Computational profile

The computational workflow of lattice QCD spans several stages, each characterised by different computational patterns. At its core lies the generation of gauge-field configurations using large-scale Monte Carlo simulations. These configurations form ensembles that are subsequently used for a wide range of physics studies.

Once generated, ensembles often remain scientifically valuable for many years. New analysis techniques are continuously applied to existing configurations and novel observables measured, meaning that the computational investment in ensemble generation continues to produce results over long timescales.

2.1. Typical workflows

Lattice workflows can involve a wide variety of computational tasks and analysis steps, often combining several software frameworks and numerical techniques. In practice, the full scientific pipeline may include configuration generation, "measurements", and increasingly sophisticated analysis procedures. In this section we provide a simplified overview of these workflows, focusing on the main components that dominate the overall computational cost.

Configuration generation. Gauge configurations are generated using the Hybrid Monte Carlo (HMC) algorithm [13] and related methods that sample the probability distribution defined by the lattice action as a Markov chain. In practice, the simulation evolves a gauge field configuration through a sequence of molecular-dynamics trajectories, with a Metropolis accept/reject step after each individual trajectory. The size of the lattice is determined by two main physical requirements. The lattice spacing a must be small enough to control discretisation effects, while the physical volume $\propto L^4$ must be large enough to accommodate the relevant hadronic states without significant finite-volume distortions. These constraints are closely related to the characteristic physical scales that must be resolved in the simulation. At long distances, the box size must be large enough to accommodate the lightest hadronic state in the theory, the pion. At the same time, the lattice spacing must be small enough to resolve the shortest physical scales relevant for the observables under investigation, which are typically set by the heaviest states or largest momenta entering the problem.

For present-day calculations with close to physical quark masses, this typically leads to lattice spacings in the range $a \sim 0.04$–0.1 fm and spatial volumes with linear sizes of several fermis (1 fm = 10^{-15} m). As a result, typical lattice grids used in modern simulations contain between 10^7 and 10^9 lattice sites, corresponding, for example, to lattices of size $64^3 \times 128$, $96^3 \times 192$, or $128^3 \times 256$.

The computational cost of this procedure is dominated by repeatedly solving large sparse linear systems involving the lattice Dirac operator, which describes the propagation of quarks in the background gauge field. These linear systems typically involve matrices with dimensions proportional to the number of lattice sites multiplied by internal spin and colour degrees of freedom. For the lattice volumes quoted above this amounts to order 10^{10} degrees of freedom in the linear system, or more.

In addition, as the lattice spacing is decreased, autocorrelation times grow rapidly [14, 15] and increasingly long Markov chains need to be generated in order to obtain a sufficiently large number of statistically independent configu-

rations. This effect, combined with large lattice volumes required to control finite-volume effects, makes configuration generation one of the most computationally demanding components of lattice QCD simulations and of workloads in high-performance computing in general.

While the discussion above provides a useful estimate of the computational cost of lattice QCD simulations, it represents only a simplified picture. Over the past decades the lattice community has continuously worked to overcome these limitations through the development of improved numerical algorithms alongside advances in computing hardware.

Key developments include refinements of the Hybrid Monte Carlo algorithm [13], such as mass-preconditioning [16], domain-decomposition techniques [17], deflation methods [18], and, more recently, multigrid solvers for the lattice Dirac operator [19]. These innovations have significantly reduced the cost of the most demanding parts of the calculation and have enabled simulations at or near the physical quark masses, at small lattice spacings.

Measurements of observables. Once gauge-field configurations have been generated, physical observables are extracted through "measurement calculations" performed on these ensembles. Most observables in lattice QCD are obtained from correlation functions constructed from quark and gluon fields. In practice, computing these correlation functions requires solving additional linear systems involving the Dirac operator, similar to those encountered during configuration generation. These measurements typically require many independent inversions of the Dirac operator for each configuration. To improve statistical precision, the calculations are repeated for multiple source locations on the lattice and often for several operator structures.

Observables of interest such as hadron masses, matrix elements, or form factors are extracted from correlation functions at large Euclidean times. A central challenge in these calculations is the signal-to-noise problem. For many observables, particularly those involving baryons or multi-hadron systems, the signal contained in correlation functions decreases exponentially with the Euclidean time separation, while the statistical noise decreases much more slowly. As a result, the ratio between signal and noise deteriorates exponentially at large time separations, making it increasingly difficult to extract the desired physical information.

Controlling this effect requires ensembles of large numbers of configurations and a significant number of measurements per configuration. Various strategies are used to mitigate the problem, including improved operator constructions, variance-reduction techniques, and stochastic estimation methods. Nevertheless, signal-to-noise degradation remains a fundamental limitation that often drives the overall computational cost of many lattice QCD studies.

Machine-learning assisted workflows. Recent developments have introduced machine-learning techniques into several aspects of lattice QCD workflows. These approaches are being explored for tasks along the entire pipeline of a lattice QCD simulation: from the tuning of the action parameters, via the generation of ensembles and the computation of observables, to the data analysis [20]. As an example, it is now possible to machine-learn a renormalisation–group-improved lattice gauge action ("perfect action") [21] via the use of *lattice gauge equivariant convolutional neural networks* [22]. Variance or noise reduction in correlation functions can be achieved via the use of (automated) differentiation of lower n-point functions [23, 24].

A particularly active area of research concerns the use of generative models to improve sampling in Monte Carlo simulations. In this context, methods in generative AI are being adopted as a way to learn transformations that map simple probability distributions to the highly structured distributions describing lattice field configurations. A distinction has to be drawn between approaches that learn from the action, such as normalising flow [25], and those that learn from data, such as diffusion models [26], and hence rely on large-scale ensembles generated using standard (HMC) approaches. Once trained, such models can potentially generate candidate configurations with reduced autocorrelations, or provide improved proposal distributions for Monte Carlo algorithms. These approaches may help mitigate some of the algorithmic challenges associated with critical slowing down, which increasingly affect simulations at fine lattice spacings. While there are some attempts to go to four dimensions [27], so far the emphasis is on lower-dimensional models. Importantly, there are deep links between methods proposed in the machine-learning literature and those developed in lattice field theory, which provides a fruitful path for future interaction [28, 29]. A strategic roadmap, based on a survey across particle and nuclear physics, is now available [30].

Machine-learning techniques are also being explored as surrogate models capable of approximating expensive components of lattice calculations, thereby reducing the number of costly solver applications required in certain analysis tasks. While these approaches remain at an exploratory stage and do not yet replace the established numerical algorithms used in lattice simulations, they offer promising avenues for improving efficiency and scalability. As modern HPC architectures increasingly integrate AI-oriented hardware and software ecosystems, machine-learning-

assisted methods may become an important complement to traditional lattice QCD workflows.

2.2. Main hardware bottlenecks

Modern lattice QCD codes are dominated by repeated applications of sparse matrices, such as the Dirac operator, within iterative linear solvers. While the computational effort is perfectly balanced over the space-time grid, the corresponding kernels have relatively low arithmetic intensity and are, therefore, often limited by memory bandwidth and communication rather than peak floating-point performance. Modern multigrid and deflated solvers further amplify this effect, as key components of these algorithms reduce the local arithmetic workload and make performance increasingly dependent on low-latency, high-bandwidth network communication. Thus, high-bandwidth, low-latency interconnects and large local CPU/GPU memory are key requirements, especially for measurement applications where solutions of the Dirac equation are reused extensively.

At the same time, increasing floating-point precision is becoming an important requirement. Although mixed-precision algorithms are often used in inner iterations of solvers to maximise performance, many observables require high numerical accuracy and strict control of rounding errors. This requirement becomes more pronounced as lattice volumes grow, where maintaining numerical stability over large configuration volumes becomes essential. In this context, current trends in GPU hardware, which increasingly prioritise lower-precision arithmetic to maximise throughput, could be suboptimal for lattice workloads, where sustained and reliable double-precision performance remains critical. A potential solution may be the use of emulated high-precision arithmetic. This would, however, require a reliable and verified implementation of the emulation for simulation results to be trustworthy.

2.3. Evolution of computational effort

Over the past decades the computational scale of lattice QCD simulations has grown dramatically. Early simulations were limited to comparatively small lattice volumes, using computing resources that were state of the art at the time but modest by present-day standards. Modern simulations routinely employ volumes exceeding $64^3 \times 192$ with near-physical quark masses [31, 32] and require large statistics [33]. A non-exhaustive list of large collaborations based in Europe, or with a significant European component, includes CLS, CSSM/QCDSF/UKQCD, ETMC, FASTSUM, HotQCD, openLAT, RBC/UKQCD, RC*, TELOS, and TWEXT. These collaborations coordinate ensemble production campaigns across multiple HPC systems and the subsequent sharing of the resulting ensembles.

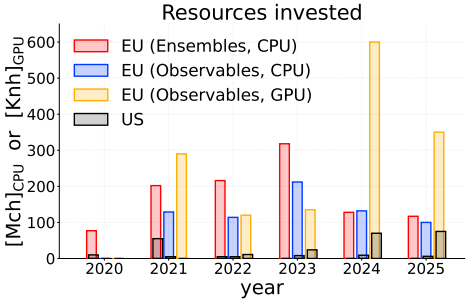


Figure 1: Evolution of computational resources invested in the OpenLAT collaboration, separated into configuration generation and observable measurements.

As an illustrative example, fig. 1 shows the evolution of computational resources within the OpenLAT collaboration, one of the youngest large-scale initiatives. Although this is a single case, it reflects a general pattern: the upfront cost of configuration generation supports a sustained physics programme through community reuse. In addition, a constant effort to develop novel algorithms and new software kernels is required to utilise state-of-the-art machines [34, 35]. An example for this ongoing effort is the development of `tmLQCD`, see fig. 2a, which could substantially speed up simulations using multi-grid solvers [36], as well as lately offloading solver and force computations to the `QUDA` library [37, 32]. Another example is the recent porting of `HiRep` to GPU architectures [38], which demonstrates excellent strong-scaling performance, see fig. 2b.

3. Main requirements for the future

3.1. Core hardware requirements for lattice investigations

The performance of lattice applications is primarily determined by a combination of memory bandwidth, communication efficiency, and the ability to sustain performance across large parallel systems. The dominant computational kernel, namely the application of the Dirac operator within iterative solvers, is inherently bandwidth-bound. This situation has become even more pronounced with the adoption of novel algorithms, which reduce the overall floating-point workload while increasing pressure on memory access and data movement.
As a consequence:

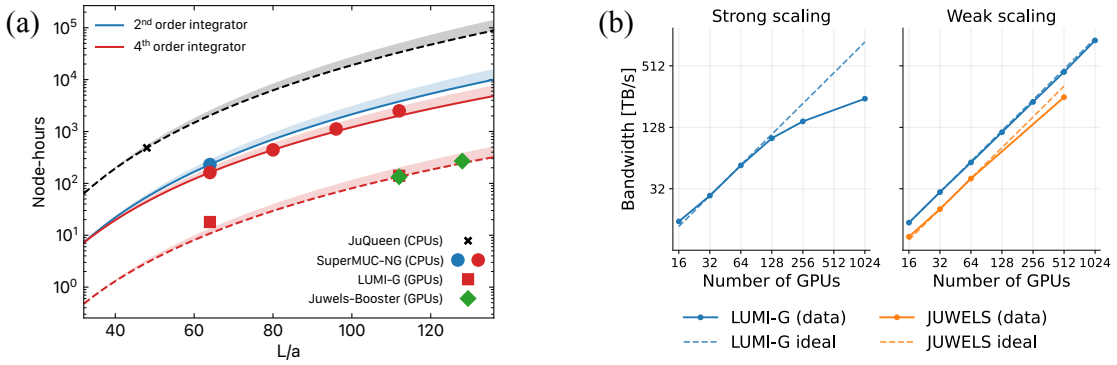


Figure 2: (a): `tmLQCD`: Cost per molecular dynamics unit for a simulation with light, strange and charm quarks tuned to physical masses. Utilising multi-grid solvers accelerated the simulations by a factor 10. Using GPUs decreases the required node-hours by another factor 10. (b): `HiRep`: Strong and weak scaling of the Dirac operator on LUMI-G and JUWELS Booster with all dimensions parallelised up to 1024 GPUs.

- **High memory bandwidth** is a critical requirement for future architectures. Equally important is the availability of low-latency, high-bandwidth interconnects, as lattice simulations rely heavily on nearest-neighbour communication patterns combined with frequent global reductions. Achieving efficient strong scaling to large fractions of modern supercomputers therefore depends both on network performance and the ability of the runtime system to overlap communication and computation effectively.
- **Large on-device memory**, such as high-bandwidth memory on accelerators, is also of central importance. Many applications require storing multiple fields and propagators simultaneously, particularly during measurement phases where solutions of the Dirac equation are reused extensively.
- In this context, a **balanced ratio between compute capability and memory bandwidth** is essential. Peak floating-point performance alone is not a reliable indicator of application performance; instead, sustained performance for stencil-like kernels with regular memory access patterns is the relevant metric for lattice workloads.
- **High and stable double precision performance** remains essential to ensure numerical accuracy, in particular for global reductions and long molecular-dynamics trajectories where rounding errors may accumulate. In selected cases, support for extended precision may also be required, for example in the presence of ill-conditioned systems or long summations. In the past this was usually realised in hardware, however, emulated high-precision arithmetic may be feasible, if performant and sufficiently verified.
- Support for **mixed-precision computations** is increasingly important. Modern solvers often exploit lower precision (such as single or half precision) in inner iterations to improve performance, while maintaining double precision for outer corrections and for the computation of physical observables.
- Lattice QCD simulations require efficient exploitation of **massive parallelism**. Architectures based on GPUs or many-core processors are well suited to this task, provided that lattice domains can be mapped efficiently onto the available parallel resources. Strong scaling to very large system sizes remains a key requirement, especially for large-volume simulations and long Markov chains. This, in turn, depends critically on both hardware capabilities and software support, including efficient MPI implementations and runtime systems. Given the regular structure of lattice discretisations, topology-aware network designs—such as **hyper-torus or locality-aware networks** offer clear advantages by minimising communication overhead and preserving locality in nearest-neighbour exchanges.

3.2. Software

Given the high demand in computing resources, the lattice QCD community has a long and successful history in self-developing highly optimised software, often including hand-optimised code targeting specific HPC architectures. For instance, the lattice community had already realised and applied the power of GPUs for scientific applications at a time when GPUs were mostly used for gaming [39]. This has led to the development of a handful of simulation software suites, maintained by small groups of physicists often connected to one of the collaborations. Examples are `Chroma`, `Grid`, `HiRep`, `MILC-QCD`, `openQCD`, `QUDA`, `SIMULATeQCD`, and `tmLQCD`.

With the increasing complexity of modern computer architectures, performance portability has become a major challenge, as writing hand-optimised code for each platform is increasingly demanding. This has also sparked the

development of novel approaches, for example implemented in `Kokkos`, a performance portability library that provides abstractions for parallel execution and data management. By expressing parallelism and memory access patterns at a high level, these approaches aim to enable compilers and backends to map the same code efficiently onto different architectures, such as multi-core CPUs and GPU accelerators, from a single source code base.

Maintaining such code suites requires continuous effort: continuous integration systems and automated performance regression tests are becoming increasingly important for sustaining reliability, reproducibility, and numerical efficiency. Moreover, these software packages require not only a stable and well maintained software stack, but also well-trained and sufficient personnel at the relevant computing centres: lattice QCD software often tests the boundaries of the hardware, and relies on the most recent features of the development toolchain. Therefore, small updates in the software stack regularly break the build chain of lattice QCD software, or lead to crashes. To address this, lattice QCD codes should be included into continuous testing pipelines of the high-performance computing centres. This is also useful for detecting hardware and compiler/module stack issues.

3.3. Human resources

Efficient utilisation of modern HPC systems requires close and sustained collaboration between research groups and computing centres. Performance analysis, architecture optimisation, and management of increasingly complex simulation workflows demand specialised expertise that goes beyond the traditional skills of individual research groups.

Dedicated support personnel with expertise in parallel programming, GPU optimisation, and performance engineering are crucial for translating algorithmic requirements into efficient implementations on evolving HPC architectures. Such roles are particularly critical given that they require a combination of in-depth knowledge of numerical algorithms, hardware architectures, and software engineering practices, while at the same time being highly attractive for the broader job market. As a result, maintaining and developing this expertise within academia remains a structural challenge for many scientific communities, including lattice field theory, where software development is still largely distributed across individual groups and lacks a coherent, long-term European framework.

A successful model to address these challenges is provided by the Swiss Platform for Advanced Scientific Computing (PASC, `https://pasc-ch.org/about/index.html`). PASC fosters close collaboration between universities and the Swiss National Supercomputing Centre, with a strong focus on application-driven software development and co-design between domain scientists and HPC experts. Extending similar frameworks at the European level would significantly strengthen the ability of lattice field theory and related disciplines to exploit current and future HPC systems effectively. In this context, the EuroHPC Centres of Excellence could represent a decisive step forward in establishing such coordinated efforts at the European scale.

3.4. Critical view of present trends in EuroHPC

European initiatives such as EuroHPC provide unprecedented computational capabilities and represent a major strategic investment for European science and technology. The deployment of large-scale systems across Europe has significantly expanded access to leadership-class computing resources. At the same time, several emerging trends offer new opportunities to further strengthen the impact of these infrastructures across a broad range of scientific domains.

- An important opportunity lies in broadening the criteria used to guide hardware procurement decisions. While current approaches capture peak floating-point performance and accelerator-driven workloads, complementing them with community-specific benchmarks would ensure a more balanced representation of scientific needs. This is not only relevant for lattice QCD, but also for other numerical approaches to problems in science and engineering, including hydrodynamics and numerical gravity [40]. These applications share demanding requirements in terms of communication costs, memory bandwidth, sustained strong scaling, network topology, latency, floating-point precision, and sustained bandwidth. Incorporating representative workloads from these communities into benchmarking suites would therefore help ensure that future systems suit a broader range of scientific applications.
- A further strategic opportunity is the development of a clear long-term European roadmap towards post-exascale computing, covering both an intermediate five-year horizon and a longer ten-year perspective. Building on the strong foundation established by current EuroHPC systems, increased visibility on future architectures and infrastructure evolution would provide valuable guidance to scientific communities engaged in long-term code development and optimisation. Establishing such a roadmap, in alignment with other international efforts, would strengthen Europe's position in the global HPC landscape and support sustained scientific leadership.
- The rapid growth of AI-driven workloads represents a major and welcome evolution in the HPC landscape, opening

new scientific and technological opportunities. At the same time, this trend should be accompanied by a balanced allocation of resources that continues to support other computational science domains.

- Established fields such as lattice field theory have historically been key drivers of innovation, producing advances in algorithms, software, and hardware whose broader impact is often difficult to anticipate but frequently transformative. Maintaining strong support for these research directions alongside emerging AI workloads will help preserve a diverse and resilient scientific ecosystem, maximising both immediate and long-term societal benefits.

A balanced approach to procurement, access policies, and long-term planning is essential for communication-bound applications such as lattice field theory.

4. Conclusions

Researchers in lattice field theory are among the largest and most efficient users of high-performance computing resources in fundamental science. The computational workloads place sustained demands on memory bandwidth, communication, and scalability, providing stringent benchmarks for current and future HPC architectures. At the same time, lattice field theory illustrates the broader impact of curiosity-driven research. Its first-principles description of strongly interacting systems has driven methodological advances beyond its original domain, including the development of algorithms such as Hybrid Monte Carlo (Hamiltonian Monte Carlo) [13] and contributions to the co-design of HPC systems, exemplified, e.g., by IBM's BlueGene architectures. These developments have had a lasting impact on computational science and data-driven methodologies.

The field remains one of the most computationally demanding areas of computational science, with progress tightly coupled to advances in algorithms, computing infrastructures and, potentially, machine learning. Looking beyond the exascale era, continued progress will require balanced HPC architectures capable of sustaining memory bandwidth and communication performance, together with robust and sustainable software ecosystems.

A clear vision for post-exascale computing is therefore essential. This must ensure balanced support for fundamental research domains, such as lattice field theory, that drive conceptual advances and technological innovation, along with emerging trends.

Sustained investment in human expertise is equally critical. The increasing complexity of hardware and software demands dedicated support at the interface of physics, numerical methods, and HPC. Establishing stable career paths for research software engineers and strengthening collaboration between scientific communities and computing centres are essential components of an effective long-term strategy.

Acknowledgements

The authors thank L. Del Debbio, A. Francis, B. Kostrzewa, C. Pica, S. Ryan, and S. Schaefer for valuable discussions and input. We also thank the OpenLAT collaboration for sharing information on their computational resource allocations. GA is supported by STFC grant ST/X000648/1 and a Royal Society Leverhulme Trust Senior Research Fellowship. JF received funding from the European Research Council via the project "LEEX" (GA 101170304). CU and SK were supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the CRC 1639 NuMeriQS – project no. 511713970 and under Germany's Excellence Strategy – EXC 3107 – Project-ID 533766364 as part of the Cluster of Excellence Color-meets-Flavor. GB, SK and CU are supported in part by DFG project 460248186 (PUNCH4NFDI). We acknowledge EuroHPC JU for awarding access under the projects EHPC-EXT-2025E01-079 (222 knh on LUMI-G), EHPC-EXT-2023E01-010 (111 Mch on LUMI-C), EHPC-EXT-2025E01-091 (JUPITER), EHPC-EXT-2022E01-096 (120 Mch on LUMI-C), and EHPC-EXT-2023E01-018 (462 knh on Leonardo and 120 Mch on LUMI-C), which contributed to the studies presented.

References

[1] R. Aliberti, et al., The anomalous magnetic moment of the muon in the Standard Model: an update, Phys. Rept. 1143 (2025) 1–158. doi:10.1016/j.physrep.2025.08.002.

[2] J. de Blas, M. Dunford, E. Bagnaschi, A. Freitas, P. P. Giardino, et al., Physics Briefing Book: Input for the 2026 update of the European Strategy for Particle Physics doi:10.17181/CERN.35CH.2O2P.

[3] G. Aarts, et al., Phase Transitions in Particle Physics: Results and Perspectives from Lattice Quantum Chromo-Dynamics, Prog. Part. Nucl. Phys. 133 (2023) 104070. doi:10.1016/j.ppnp.2023.104070.

[4] S. Aoki, et al., FLAG Review 2019: Flavour Lattice Averaging Group (FLAG), Eur. Phys. J. C 80 (2) (2020) 113. doi:10.1140/epjc/s10052-019-7354-7.

[5] Y. Aoki, et al., FLAG Review 2021, Eur. Phys. J. C 82 (10) (2022) 869. doi:10.1140/epjc/s10052-022-10536-1.

[6] Y. Aoki, et al., FLAG review 2024, Phys. Rev. D 113 (1) (2026) 014508. doi:10.1103/nfzp-p5dn.

[7] C. T. H. Davies, A. C. Irving, R. D. Kenway, C. M. Maynard, International lattice data grid, Nucl. Phys. B Proc. Suppl. 119 (2003) 225–226. doi:10.1016/S0920-5632(03)01509-3.

[8] A. C. Irving, R. D. Kenway, C. M. Maynard, T. Yoshie, Progress in building an International Lattice Data Grid, Nucl. Phys. B Proc. Suppl. 129 (2004) 159–163. doi:10.1016/S0920-5632(03)02517-9.

[9] A. Ukawa, Status of international lattice data grid: An Overview, Nucl. Phys. B Proc. Suppl. 140 (2005) 207–208. doi:10.1016/j.nuclphysbps.2004.11.251.

[10] M. G. Beckett, B. Joo, C. M. Maynard, D. Pleiter, O. Tatebe, T. Yoshie, Building the International Lattice Data Grid, Comput. Phys. Commun. 182 (2011) 1208–1214. doi:10.1016/j.cpc.2011.01.027.

[11] M. D. Wilkinson, et al., The FAIR Guiding Principles for scientific data management and stewardship, Sci. Data 3 (1) (2016) 160018. doi:10.1038/sdata.2016.18.

[12] F. Karsch, H. Simma, T. Yoshie, The International Lattice Data Grid – towards FAIR data, PoS LATTICE2022 (2023) 244. doi:10.22323/1.430.0244.

[13] S. Duane, A. D. Kennedy, B. J. Pendleton, D. Roweth, Hybrid Monte Carlo, Phys. Lett. B 195 (1987) 216–222. doi:10.1016/0370-2693(87)91197-X.

[14] L. Del Debbio, G. M. Manca, E. Vicari, Critical slowing down of topological modes, Phys. Lett. B 594 (2004) 315–323. doi:10.1016/j.physletb.2004.05.038.

[15] S. Schaefer, R. Sommer, F. Virotta, Critical slowing down and error analysis in lattice QCD simulations, Nucl. Phys. B 845 (2011) 93–119. doi:10.1016/j.nuclphysb.2010.11.020.

[16] M. Hasenbusch, Speeding up the hybrid Monte Carlo algorithm for dynamical fermions, Phys. Lett. B 519 (2001) 177–182. doi:10.1016/S0370-2693(01)01102-9.

[17] M. Lüscher, Solution of the Dirac equation in lattice QCD using a domain decomposition method, Comput. Phys. Commun. 156 (2004) 209–220. doi:10.1016/S0010-4655(03)00486-7.

[18] M. Lüscher, Local coherence and deflation of the low quark modes in lattice QCD, JHEP 07 (2007) 081. doi:10.1088/1126-6708/2007/07/081.

[19] R. Babich, J. Brannick, R. C. Brower, M. A. Clark, T. A. Manteuffel, S. F. McCormick, J. C. Osborn, C. Rebbi, Adaptive multigrid algorithm for the lattice Wilson-Dirac operator, Phys. Rev. Lett. 105 (2010) 201602. doi:10.1103/PhysRevLett.105.201602.

[20] D. Boyda, et al., Applications of Machine Learning to Lattice Quantum Field Theory, in: Snowmass 2021, 2022.

[21] K. Holland, A. Ipp, D. I. Müller, U. Wenger, Machine-Learned Renormalization-Group-Improved Gauge Actions and Classically Perfect Gradient Flows, Phys. Rev. Lett. 136 (3) (2026) 031901. doi:10.1103/k41k-2pnc.

[22] M. Favoni, A. Ipp, D. I. Müller, D. Schuh, Lattice Gauge Equivariant Convolutional Neural Networks, Phys. Rev. Lett. 128 (3) (2022) 032003. doi:10.1103/PhysRevLett.128.032003.

[23] G. Catumba, A. Ramos, Stochastic automatic differentiation and the signal to noise problem, Eur. Phys. J. C 85 (9) (2025) 1037. doi:10.1140/epjc/s10052-025-14690-0.

[24] R. Abbott, D. Boyda, Y. Fu, D. C. Hackett, G. Kanwar, F. Romero-López, P. E. Shanahan, J. M. Urban, Variance reduction in lattice QCD observables via normalizing flows (2026). arXiv:2603.02984.

[25] K. Cranmer, G. Kanwar, S. Racanière, D. J. Rezende, P. E. Shanahan, Advances in machine-learning-based sampling motivated by lattice quantum chromodynamics, Nature Rev. Phys. 5 (9) (2023) 526–535. doi:10.1038/s42254-023-00616-w.

[26] G. Aarts, D. E. Habibi, A. Ipp, D. I. Müller, T. R. Ranner, L. Wang, W. Wang, Q. Zhu, Generalizable Equivariant Diffusion Models for Non-Abelian Lattice Gauge Theory (2026). arXiv:2601.19552.

[27] R. Abbott, D. Boyda, G. Kanwar, F. Romero-López, D. C. Hackett, P. E. Shanahan, J. M. Urban, Progress in Normalizing Flows for 4d Gauge Theories, PoS LATTICE2024 (2025) 066. doi:10.22323/1.466.0066.

[28] M. Caselle, E. Cellini, A. Nada, M. Panero, Stochastic normalizing flows as non-equilibrium transformations, JHEP 07 (2022) 015. doi:10.1007/JHEP07(2022)015.

[29] L. Wang, G. Aarts, K. Zhou, Diffusion models as stochastic quantization in lattice field theory, JHEP 05 (2024) 060. doi:10.1007/JHEP05(2024)060.

[30] S. Caron, et al., Strategic white paper on AI infrastructure for particle, nuclear, and astroparticle physics: insights from JENA and EuCAIF, Mach. Learn. Sci. Tech. 7 (1) (2026) 013002. doi:10.1088/2632-2153/ae35cd.

[31] K. K. Szabo, L. Lellouch, Z. Fodor, F. Stokes, B. C. Toth, G. Wang, New physics in the muon magnetic moment?, Procedia Comput. Sci. 240 (2024) 91–98. doi:10.1016/j.procs.2024.07.012.

[32] C. Alexandrou, et al., Large-scale simulations of lattice QCD for nucleon structure using Nf=2+1+1 flavors of twisted mass fermions, Procedia Comput. Sci. 267 (2025) 92–101. doi:10.1016/j.procs.2025.08.236.

[33] S. Borsányi, Z. Fodor, J. N. Guenther, P. Parotto, A. Pásztor, L. Pirelli, K. K. Szabó, C. H. Wong, The QCD crossover line in a finite volume, Procedia Comput. Sci. 267 (2025) 4–13. doi:10.1016/j.procs.2025.08.229.

[34] J. Finkenrath, Review on Algorithms for dynamical fermions, PoS LATTICE2022 (2023) 227. doi:10.22323/1.430.0227.

[35] J. Finkenrath, Future trends in lattice QCD simulations, PoS EuroPLEx2023 (2024) 009. doi:10.22323/1.451.0009.

[36] 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. doi:10.1103/PhysRevD.94.114509.

[37] 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. doi:10.22323/1.430.0340.

[38] V. Drach, S. Martins, C. Pica, A. Rago, High-performance simulations of higher representations of wilson fermions, Computer Physics Communications 322 (2026) 110061. doi:https://doi.org/10.1016/j.cpc.2026.110061.

[39] G. I. Egri, Z. Fodor, C. Hoelbling, S. D. Katz, D. Nogradi, K. K. Szabo, Lattice QCD as a video game, Comput. Phys. Commun. 177 (2007) 631–639. doi:10.1016/j.cpc.2007.06.005.

[40] The scientific and innovation case for computing in europe 2026–2034, iSBN: 9789464597738 (Apr. 2026). URL https://prace-ri.eu/scientific-case/the-scientific-and-innovation-case-for-computing-in-europe-2026-2034/

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 34–47

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

LBFAST: A Lightweight Moment-Represented Lattice Boltzmann Solver for Multi-GPU Architectures

Marco Lauricella[a,*], Andrea Montessori[b], Giorgio Amati[c], Filippo Spiga[d], Adriano Tiribocchi[a], Massimo Bernaschi[a], Sauro Succi[a,e,f]

[a]*Istituto per le Applicazioni del Calcolo, Consiglio Nazionale delle Ricerche, Via dei Taurini 19, Rome, 00185, Italy*
[b]*Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University, Via Vito Volterra, Rome, 00146, Italy*
[c]*HPC Department, CINECA, Rome 00185, Italy*
[d]*NVIDIA Development UK Ltd, Milton Hall, Ely Rd, Milton, Cambridge CB24 6WZ, United Kingdom*
[e]*Center for Life Nano- & Neuro-Science, Fondazione Istituto Italiano di Tecnologia, Viale Regina Elena 295, Rome, 00161, Italy*
[f]*Department of Physics, Harvard University, 17 Oxford St., Cambridge, 02138, MA, USA*

Abstract

We present *LBFAST*, a GPU-oriented lattice Boltzmann solver based on a lightweight moment-represented formulation, in which post-collision populations are reconstructed on the fly from a reduced set of moments rather than stored explicitly. This approach significantly lowers the memory footprint, enabling large three-dimensional simulations within the constraints of modern accelerator architectures, where VRAM capacity and bandwidth are critical resources. The method is assessed through standard single- and two-component benchmarks demonstrating good accuracy and stability. Extensive scaling experiments on multi-GPU systems show near-ideal weak scaling up to 512 GPUs and sustained performance across different velocity sets. The combination of reduced memory usage, competitive throughput, and stable energy efficiency makes the proposed formulation a practical route for large-scale lattice Boltzmann simulations on current and emerging HPC platforms.

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026

Keywords: Computational fluid dynamics; Lattice Boltzmann; HCP GPU computing; Multiphase flows

1. Introduction

Over the last three decades, the lattice Boltzmann method (LBM) has become a well-established tool for computational fluid dynamics, thanks to its kinetic foundation, algorithmic simplicity, and remarkable suitability for parallel computing [39, 37, 17, 19]. In its standard form, LBM stores the full set of discrete particle populations at every lattice site. This makes the method inherently memory intensive, especially for three-dimensional simulations on

* Corresponding author.
E-mail address: marco.lauricella@cnr.it

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.017

modern GPU-based architectures [41]. This issue has long been recognized as one of the main practical limitations of large-scale LB simulations [25, 38].

The idea that the full set of populations may not always need to be stored explicitly is not entirely new. Early reflections in this direction can already be found in the work of Ladd and Verberg [20], who pointed out that substantial memory savings might be achieved by exploiting the relation between distribution functions and hydrodynamic fields. In a broader sense, this observation is rooted in the very structure of the lattice Boltzmann method: the discrete populations are not arbitrary variables, but a finite kinetic representation whose physically relevant content is encoded in a limited set of moments, namely density, momentum, and the tensors governing the stress and higher-order non-equilibrium contributions [17].

A decisive theoretical step came with the Hermite-based reformulation of lattice Boltzmann models and with the development of regularized schemes [21, 22, 31]. In the work of Latt and Chopard [21], the method was recast so as to emphasize macroscopically relevant variables rather than the microscopic populations themselves, thereby clarifying which degrees of freedom are essential to recover the Navier–Stokes dynamics. This viewpoint was later strengthened by regularization procedures, where the non-equilibrium part of the distribution is projected onto a controlled Hermite subspace, filtering out spurious high-order contributions and improving both stability and accuracy [46, 45, 10, 32, 15, 29, 3, 27]. In this context, the Hermite representation provides a compact way of organizing the kinetic hierarchy in terms of moments. For the athermal case, this structure becomes especially powerful because the equilibrium Hermite coefficients satisfy a simple recursive relation, allowing higher-order coefficients to be reconstructed from lower-order moments in a compact and physically consistent way [36, 27].

These developments opened the way to what is now commonly called *moment-represented* or *lightweight* lattice Boltzmann. The central idea is simple: instead of storing all lattice populations, one stores only the hydrodynamically relevant moments—typically density, momentum, and the second-order tensor associated with the non-equilibrium stress—while reconstructing the populations on the fly whenever they are needed for collision and propagation. In three dimensions, this reduces the number of stored variables per lattice site from, for example, 19 populations in a D3Q19 formulation to only 10 moment variables. Since contemporary high-performance implementations of LBM are often limited more by memory bandwidth than by floating-point throughput [25, 38], such a reduction can translate into a substantial performance gain.

The modern formulation of moment-represented LBM has been developed primarily in the context of regularized schemes. Vardhan *et al.* [43] demonstrated that storing only moment-based data can significantly reduce memory footprint and data motion on massively parallel architectures, while preserving the accuracy of regularized LBM for fluid-dynamical applications. This line of work was further extended by Gounley *et al.* [12], who introduced a dedicated propagation pattern for the moment representation, showing that careful control of cache reuse and data movement is essential to fully exploit the potential of the compressed kinetic description. More recently, Ferrari *et al.* [11] provided a three-dimensional GPU implementation of the moment representation, confirming that the reduction in global memory traffic can yield substantial speedups with no appreciable loss of accuracy, especially in single precision.

Within this line of research, the work by Tiribocchi *et al.* [40] offers a particularly clear synthesis of both the historical motivations and the computational implications of the moment representation strategy, also called *lightweight* lattice Boltzmann. The formulation in Ref. [40] revisits earlier theoretical ideas in light of modern hardware constraints and shows that, in regularized multicomponent lattice Boltzmann models, reconstructing populations from hydrodynamic information may lead to large savings in memory occupation and data-access costs without compromising the underlying physics. In this sense, lightweight LBM is not merely a technical optimization, but the natural outcome of a mature understanding of which kinetic degrees of freedom are genuinely needed by the numerical scheme and which can instead be reconstructed when required.

In this work, we present *LBFAST*, a high-performance lattice Boltzmann solver specifically developed to implement the *lightweight* lattice Boltzmann approach on modern GPU-accelerated supercomputers. Within this framework, *LBFAST* follows the *moment-represented* lattice Boltzmann paradigm, in which the post-collision populations are reconstructed on the fly from a reduced set of hydrodynamic and kinetic variables and immediately streamed to neighboring lattice nodes to update the reduced set of variables at the new time step, without ever being stored as a full population array in global device memory [40, 43, 12, 11]. Compared with the previous lightweight implementation of Tiribocchi et al. [40], the present work introduces two main advances. First, *LBFAST* extends the

population-free, moment-represented strategy to a distributed-memory multi-GPU solver supporting one-, two-, and three-dimensional MPI domain decompositions. Second, the underlying multiphase formulation is changed from the weakly compressible color-gradient setting, in which two component distribution functions are reconstructed separately, to an incompressible velocity–pressure formulation coupled to a conservative Allen–Cahn equation for the phase field [23]. In this setting, density and viscosity are reconstructed from the phase field, while pressure and velocity are evolved through the lightweight moment-represented flow solver. Further, the present work extends this strategy to a distributed-memory multi-GPU setting and assesses it on a modern GPU-accelerated supercomputer through numerical validation and strong- and weak-scaling measurements using up to 512 GPUs.

This strategy is particularly well suited to modern accelerator-based architectures, where lattice Boltzmann performance is typically limited by both memory bandwidth and memory footprint. In this context, the moment-represented, *lightweight* LBM formulation implemented in *LBFAST* provides a versatile framework for next-generation high-performance fluid dynamics solvers, especially in applications involving multicomponent physics, large three-dimensional domains, and GPU acceleration, where the available device memory is a key constraint.

2. Methods

LBFAST is formulated within the velocity–pressure framework already introduced in Refs. [30, 24, 7, 8, 48] , in which the flow solver evolves the velocity field u_α and the pressure p, while the density is not obtained from the zeroth moment of the flow populations. Instead, both density and viscosity are reconstructed from the phase field ϕ, which is transported by a conservative Allen–Cahn equation. In this setting, the governing equations read

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

$$\partial_t \left(\rho(\phi) u_\alpha\right) + \partial_\beta \left(\rho(\phi)\, u_\alpha u_\beta\right) = -\partial_\alpha p + \partial_\beta \left[\rho(\phi)\, \nu(\phi) \left(\partial_\beta u_\alpha + \partial_\alpha u_\beta\right)\right] + F_\alpha. \tag{1b}$$

where ν denotes the kinematic viscosity, ρ the density and F_α is the total force including the surface tension. Greek indices denote Cartesian tensor components, and summation over repeated indices is implied.

The local density and kinematic viscosity are interpolated from the phase field through linear blending laws, $\rho(\phi) = \rho_L + \phi\,(\rho_H - \rho_L)$, and $\nu(\phi) = \nu_L + \phi\,(\nu_H - \nu_L)$, where $\phi = 1$ and $\phi = 0$ correspond to the heavy and light phases, respectively.

The interface dynamics is described by a conservative Allen–Cahn equation,

$$\partial_t \phi + u_\alpha \partial_\alpha \phi = D\, \partial_\alpha \partial_\alpha \phi - \lambda\, \partial_\alpha (\phi(1-\phi) n_\alpha), \tag{2}$$

where D is the interface diffusivity and $\lambda = 4D/W$, with W denoting the interface thickness and the term $-\lambda\, \partial_\alpha(\phi(1-\phi)n_\alpha)$ acting as an interface compression contribution, counterbalancing numerical diffusion and preserving a sharp interface profile [24]. The unit normal to the interface is defined as $n_\alpha = \partial_\alpha \phi / |\nabla \phi|$.

In the lightweight moment-represented Lattice Boltzmann (LLB) approach, Eqs 1a and 1b are indirectly solved integrating the Lattice Boltzmann equation [10, 9]:

$$f_i(x_\alpha, t + \Delta t) = f_i^{eq}(x_\alpha - c_{i\alpha}\Delta t, t) + (1-\omega) f_i^{neq}(x_\alpha - c_{i\alpha}\Delta t, t) + \frac{1}{2} S_i(x_\alpha - c_{i\alpha}\Delta t, t), \tag{3}$$

where collision and streaming are combined into a single update step within a pulled scheme, S_i denotes the forcing term and $f_i^{neq} = f_i - f_i^{eq} + \frac{1}{2} S_i$ ensures the second-order accuracy in time. The relaxation time $\tau = 1/\omega$ determines the kinematic viscosity through $\nu = c_s^2(\tau - 0.5)$ with c_s the lattice speed of sound [37, 17]. However, rather than evolving the full set of populations, in LLB the distribution functions f_i are reconstructed on the fly as a truncated Hermite expansion [36]:

$$f_i^{eq} = w_i \sum_n \frac{1}{c_s^{2n} n!} \mathcal{H}^{(n)}_{i\alpha_1 \ldots \alpha_n} a^{(n)}_{eq,\alpha_1 \ldots \alpha_n}, \tag{4a}$$

$$f_i^{neq} = w_i \sum_n \frac{1}{c_s^{2n} n!} \mathcal{H}^{(n)}_{i\alpha_1 \ldots \alpha_n} a^{(n)}_{neq,\alpha_1 \ldots \alpha_n}, \tag{4b}$$

with the Hermite coefficients obtained by projection:

$$a^{(n)}_{eq,\alpha_1,\ldots,\alpha_n} = \sum_i \mathcal{H}^{(n)}_{i\alpha_1,\ldots,\alpha_n} f_i^{eq}, \tag{5a}$$

$$a^{(n)}_{neq,\alpha_1,\ldots,\alpha_n} = \sum_i \mathcal{H}^{(n)}_{i\alpha_1,\ldots,\alpha_n} f_i^{neq}. \tag{5b}$$

In practice, the expansion is truncated at second order, $n = 2$, obtaining the total Hermite coefficient as $a^{(n)}_{\alpha_1,\ldots,\alpha_n} = a^{(n)}_{eq,\alpha_1,\ldots,\alpha_n} + a^{(n)}_{neq,\alpha_1,\ldots,\alpha_n}$, consistently with the recovery of the Navier–Stokes equations. The discrete Hermite basis reads $H_i^{(0)} = 1$, $H_{i\alpha}^{(1)} = c_{i\alpha}$, $H_{i\alpha\beta}^{(2)} = (c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})$. Under this truncation, and within a velocity-based formulation [24, 7, 8, 48], the equilibrium coefficients are explicitly expressed in terms of the primitive variables:

$$a^{(0)}_{eq} = p^*, \quad a^{(1)}_{eq,\alpha} = u_\alpha, \quad a^{(2)}_{eq,\alpha\beta} = u_\alpha u_\beta, \tag{6a}$$

$$a^{(0)}_{neq} = 0, \quad a^{(1)}_{neq,\alpha} = 0, \quad a^{(2)}_{neq,\alpha\beta} = -\frac{c_2^s}{\omega}\left(\partial_\beta u_\alpha + \partial_\alpha u_\beta\right) + \frac{1}{2\rho}\left(F_\alpha u_\beta + F_\beta u_\alpha\right), \tag{6b}$$

where p^* denotes a rescaled pressure variable, defined as $p^* = p/(\rho(\phi)c_s^2)$, allowing the equilibrium to be formulated directly in terms of the primitive variables, i.e., the velocity and the pressure fields (rather than the momentum), and the non-equilibrium contribution is entirely contained in the second-order tensor $a^{(2)}_{neq,\alpha\beta}$, related to the viscous stress [27, 18] and the force term arising from the shifted definition $f_i^{neq} = f_i - f_i^{eq} + \frac{1}{2}S_i$. This projection effectively filters out non-hydrodynamic degrees of freedom, a key ingredient of regularized formulations [21].

In the LLB approach [40], exploiting Eqs 4 and 5, the lattice Boltzmann Eq. 3 is recast as:

$$\begin{aligned} a^{*(n)}_{\alpha_1,\ldots,\alpha_n}(x_\alpha, t+\Delta t) = \sum_i \mathcal{H}^{(n)}_{i\alpha_1,\ldots,\alpha_n} \Bigg[& w_i \sum_m \frac{1}{c_s^{2m} m!} \mathcal{H}^{(m)}_{i\alpha_1,\ldots,\alpha_m} \\ & \times \left(a^{(m)}_{eq,\alpha_1,\ldots,\alpha_m}(x_\alpha - c_{i\alpha}\Delta t, t) + (1-\omega)\, a^{(m)}_{neq,\alpha_1,\ldots,\alpha_m}(x_\alpha - c_{i\alpha}\Delta t, t) \right) \\ & + \frac{1}{2} S_i(x_\alpha - c_{i\alpha}\Delta t, t) \Bigg]. \end{aligned} \tag{7}$$

where i runs over all discrete lattice directions of the adopted velocity set, while m spans the Hermite expansion order from 0 to 2. In order to remain consistent with the definition $f_i^{neq} = f_i - f_i^{eq} + \frac{1}{2}S_i$, the Hermite coefficients are updated as:

$$a^{(0)} = a^{*(0)}, \quad a^{(1)}_\alpha = a^{*(1)}_\alpha + \frac{F_\alpha}{2\rho}, \quad a^{(2)}_{\alpha\beta} = a^{*(2)}_{\alpha\beta} + \frac{1}{2\rho}\left(F_\alpha u_\beta + F_\beta u_\alpha\right), \tag{8}$$

with the force term computed at the updated time, $t + \Delta t$.

Thus, the LLB approach consists of integrating in time Eq. 7 and applying the correction in Eq. 8, which depends only on the Hermite coefficients at the previous time step. The discrete forcing term S_i is defined according to the Guo scheme [13]:

$$S_i = w_i \left(\frac{c_{i\alpha} - u_\alpha}{c_s^2} + \frac{c_{i\beta}u_\beta}{c_s^4} c_{i\alpha} \right) \frac{F_\alpha}{\rho}, \tag{9}$$

where F_α denotes the total force per unit volume. In particular, the vector F_α can be split into four contributions, namely $F_\alpha = F_\alpha^s + F_\alpha^b + F_\alpha^p + F_\alpha^\nu$. Following Refs. [24, 7, 16], the capillary force is expressed as $F_\alpha^s = \mu_\phi\, \partial_\alpha\phi$, where $\mu_\phi = 4\beta\,\phi(\phi-1)\left(\phi - \frac{1}{2}\right) - \kappa\nabla^2\phi$ is the chemical potential, with the coefficients chosen so as to recover the prescribed surface tension σ and interface thickness W, yielding $\beta = 12\sigma/W$ and $\kappa = 3\sigma W/2$. The symbol F_α^b stands for an external body force, whereas F_α^p denotes the pressure force [24] equal to $F_\alpha^p = -p^* c_s^2 \partial_\alpha \rho$. This choice allows reconstructing $\partial_\alpha p = \rho c_s^2 \partial_\alpha p^* + p^* c_s^2 \partial_\alpha \rho$ in Eq. 1b with the first term already embedded in the LB equation [24]. Lastly, the contribution arising from viscosity gradients across the interface $F_\alpha^\nu = \nu\,\omega\, c_s^2\, a^{*(2)}_{\mathrm{neq},\alpha\beta}\, \partial_\beta\rho$ accounts for the non-equilibrium stress-based forcing term, as in Refs. [23, 24].

It is worth noting that Eq. 7 is formulated in a way that is independent of the specific choice of the discrete velocity set, and can therefore be readily implemented with different quadrature rules. In the present work, we consider

the D3Q19 and D3Q27 models with the Hermite expansion truncated at second order, together with a higher-order D3Q27h formulation based on the complete Hermite basis. In the latter case, the Hermite coefficients are reconstructed through the recursive closure relations of Shan et al. [36], consistently with the high-order Hermite framework discussed by Malaspinas [27].

Spatial derivatives entering both the forcing terms and the phase-field equation are computed using lattice-based stencils constructed from the weights of the D3Q27 velocity set. In particular, the first- and second-order derivatives are approximated as $\partial_\alpha \Psi = \frac{1}{c_s^2} \sum_i w_i \Psi(x_\alpha + c_{i\alpha})\, c_{i\alpha}$, and $\partial_\alpha \partial_\beta \Psi = \frac{1}{c_s^2} \left(\sum_{i \neq 0} w_i \Psi(x_\alpha + c_{i\alpha}) - w_0 \Psi(x_\alpha) \right)$.

Given the small time step imposed by the LBM, the conservative Allen–Cahn Eq. 2 is discretized using a forward-time, centered-space (FTCS) scheme. This finite-difference approach ensures stability while reducing computational cost and memory usage.

3. Implementation

The code is implemented in `CUDA Fortran`, with additional `OpenACC` directives, and uses MPI for distributed-memory parallel execution across multiple GPUs. The domain can be decomposed along one, two, or three directions, allowing flexibility in the mapping of large three-dimensional problems while keeping the collision step strictly local and the streaming step based on the on-the-fly reconstruction.

Inter-process communication is carried out using non-blocking MPI primitives. At each time step, halo layers are first assembled to gather the Hermite coefficients required by neighboring subdomains. Asynchronous send and receive calls are issued, and the computation proceeds on the interior of the subdomains while data transfers are in progress. A final synchronization through `MPI_Waitall` ensures completion of communications before updating the boundary nodes. This overlap between communication and computation reduces idle time and improves scalability.

The Hermite coefficients are stored using a block-structured layout. Lattice nodes are grouped into compact three-dimensional tiles mapped onto CUDA thread blocks, and the data are arranged as

$$n_{\text{tot}} = TILE\_DIM_x \, TILE\_DIM_y \, TILE\_DIM_z \, n_{\text{hfields}} \, n_{\text{blocks}} \tag{10}$$

Two arrays, `hfields_flip` and `hfields_flop`, are used in a double-buffering scheme. The number of fields is fixed to $n_{\text{hfields}} = 10$, corresponding to the Hermite expansion truncated at second order: one scalar moment $a^{(0)}$, three velocity components $a_\alpha^{(1)}$, and six independent components of the symmetric second-order tensor $a_{\alpha\beta}^{(2)}$. This layout is consistent with previous GPU-oriented implementations based on moment representations [11].

Within each tile, threads access data by fixing the local indices (i, j, k), leading to coalesced memory accesses. The non-locality of the streaming step is handled on the fly by reconstructing populations and storing them temporarily in shared memory. The shared buffer includes halo layers and is defined at compile time (e.g., $8 \times 8 \times 8$), providing low-latency access and limiting global-memory traffic.

The main kernel follows a fused collide-and-stream approach in a pull formulation. For each lattice site, the Hermite coefficients are read, the populations are reconstructed locally, and their contributions are accumulated directly into the output moments following Eq. 7. These output moments (Hermite coefficients) act as local accumulators, so that no full set of populations is stored in global memory. This reduces memory footprint and increases arithmetic intensity.

In this formulation, the macroscopic dynamics is encoded in a reduced set of low-order moments, while higher-order contributions are either filtered, as in regularized approaches [21], or consistently reconstructed from the retained moments through Hermite-based closures [27, 36]. This separation enables a substantial reduction in memory footprint and allows for a reorganization of the algorithm that is well suited to GPU architectures. The resulting implementation combines a compact data representation with an execution pattern tailored for large-scale simulations. To make the memory saving more explicit, let us consider a cubic domain of size 512^3, for which the total number of lattice sites is $N = 512^3$. In the present lightweight formulation, the solver stores only the ten second-order Hermite fields and uses a double-buffering strategy, so that the total storage amounts to $2 \times 10 = 20$ scalar fields per lattice site. By contrast, a standard population-based implementation with flip–flop storage requires $2 \times 19 = 38$ scalar fields for a D3Q19 stencil and $2 \times 27 = 54$ scalar fields for a D3Q27 stencil. Hence, the total device memory required by the main lattice arrays is $M = N \, n_{\text{fields}} \, b$, where n_{fields} is the number of stored scalar fields per site and b is the size in bytes of a scalar.

In double precision ($b = 8$ bytes), this gives exactly 20 GB for the present moment-represented implementation, compared with 38 GB for a standard D3Q19 population scheme and 54 GB for a standard D3Q27 one in a double-buffering approach. Therefore, the lightweight approach reduces the memory occupation by about 47.4% with

respect to D3Q19 and by about 63.0% with respect to D3Q27. In single precision, the corresponding values become 10 GB, 19 GB, and 27 GB, respectively, with the same percentage reductions. These figures refer only to the main lattice-resident flow arrays and clearly illustrate the advantage of the moment-represented formulation for large three-dimensional simulations on GPU architectures.

4. Validation and Benchmarking

We first consider a monocomponent benchmark in order to test the accuracy of the solver in a simple and controlled setting. To this end, we study the viscous decay of the three-dimensional Taylor–Green vortex in a fully periodic cubic box of size 512^3, at fixed density $\rho = 1.0$ and kinematic viscosity $\nu = 0.04074$. With the characteristic length defined as $L = N/(2\pi)$ and the initial velocity equal to $U_0 = 0.04$, this choice corresponds to a Reynolds number $Re \simeq 80$.

The initial condition is taken as in Ref. [44], consistently adapted to the pressure-based formulation [23, 7], where the hydrodynamic pressure is defined up to an arbitrary constant and taken with zero mean. The velocity field is prescribed as

$$u_x(x, y, z) = U_0 \sin(x) \cos(y) \cos(z), \tag{11a}$$

$$u_y(x, y, z) = -U_0 \cos(x) \sin(y) \cos(z), \tag{11b}$$

$$u_z(x, y, z) = 0, \tag{11c}$$

$$p(x, y, z) = \frac{U_0^2}{16} [\cos(2x) + \cos(2y)] [2 + \cos(2z)], \tag{11d}$$

where the dimensionless coordinates are defined as $x = 2\pi(i-1)/N$, $y = 2\pi(j-1)/N$, $z = 2\pi(k-1)/N$, and U_0 denotes the initial velocity amplitude.

As a diagnostic quantity, we monitor the volume-averaged kinetic energy $E(t) = \langle u_\alpha u_\alpha \rangle / 2$, and compare its evolution with the analytical viscous decay of the fundamental mode, $E(t)/E(0) = \exp\left[-6\nu k^2 t\right]$, being $k = 2\pi/N$.

The same test is repeated with three different discrete velocity sets: D3Q19, D3Q27 with Hermite expansion truncated at second order, and a high-order D3Q27h model based on the full Hermite basis.

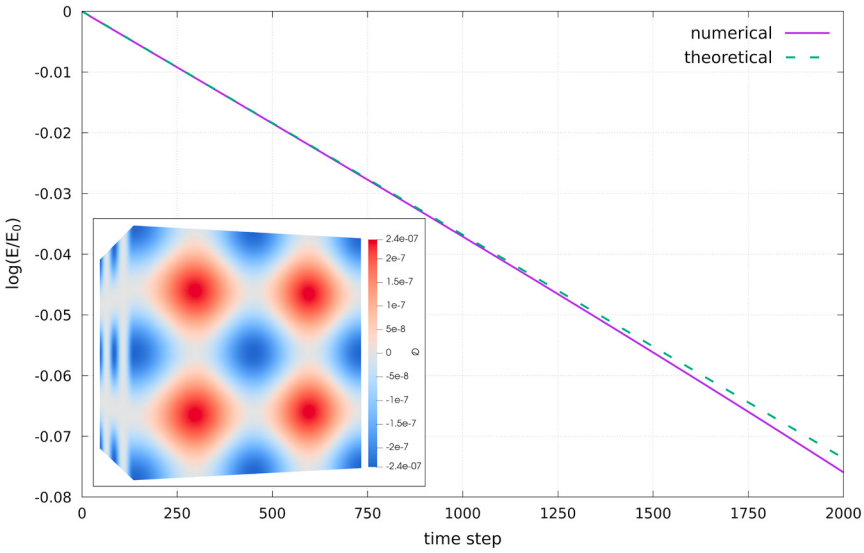


Fig. 1. Decay of the normalized kinetic energy in the Taylor–Green vortex for the D3Q27 scheme. Numerical results are in close agreement with the analytical prediction, yielding $\nu_{\text{fit}} = 0.04079$ with a relative error of $\sim 1.3 \times 10^{-3}$. Inset: the initial Q-criterion scalar field is mapped onto the surface of the computational domain, with $Q = \frac{1}{2}(\|\mathbf{\Omega}\|^2 - \|\mathbf{S}\|^2)$, where $\mathbf{\Omega}$ and $\mathbf{S}$ are the antisymmetric and symmetric parts of the velocity-gradient tensor, respectively.

Figure 1 reports the temporal evolution of the normalized kinetic energy, $E(t)/E(0)$, for the D3Q27 scheme. The numerical data follow closely the expected exponential decay, with only minor deviations at longer times, where the

numerical curve exhibits a slightly faster decay than the theoretical prediction. The slight late-time departure from the theoretical exponential decay is attributed to the accumulation of numerical dissipation and finite-resolution errors, a behavior commonly monitored in vortex-decay benchmarks as a sensitive indicator of the effective viscosity and numerical accuracy of the scheme [28].

A linear fit of $\log(E/E_0)$ over the initial decay range yields an effective kinematic viscosity $\nu_{\text{fit}} = 0.04079$, in very good agreement with the input value, with a relative deviation of approximately 1.3×10^{-3}. This confirms that both the kinetic model and its implementation accurately reproduce the viscous decay of the fundamental mode. An inset shows the initial Q-criterion field at $t = 0$, highlighting the coherent vortical structures of the Taylor–Green configuration, where $Q = \frac{1}{2}\left(\Omega_{\alpha\beta}\Omega_{\alpha\beta} - S_{\alpha\beta}S_{\alpha\beta}\right)$ denotes the second invariant of the velocity gradient tensor, with $\Omega_{\alpha\beta}$ and $S_{\alpha\beta}$ the antisymmetric and symmetric parts, respectively.

Finally, for completeness, we report that analogous fits yield $\nu_{\text{fit}} = 0.040794$ for D3Q19 and $\nu_{\text{fit}} = 0.040791$ for the high-order D3Q27h model, with relative deviations of $\sim 1.3 \times 10^{-3}$ and $\sim 1.2 \times 10^{-3}$, respectively.

We consider the layered Poiseuille flow as a two-phase benchmark with viscosity contrast, to assess the accuracy of the solver in the presence of a diffuse but stationary interface. This configuration provides a standard validation test for binary-fluid lattice Boltzmann models, since an analytical steady solution exists for two immiscible layers under laminar conditions [48]. The flow is driven by a constant body acceleration a_z along z, with no-slip walls at the boundaries normal to x and periodicity in y and z. The domain size is $L_x \times L_y \times L_z = 512 \times 8 \times 1024$, and $a_z = 1.0 \times 10^{-7}$.

The two fluids are initially arranged in layers across the channel: the less viscous phase occupies $1 \leq x \leq L_x/2$, while the more viscous one fills the remaining half. Their kinematic viscosities are $\nu_1 = 0.05$ and $\nu_2 = 0.4$, with densities $\rho_1 = 0.5$ and $\rho_2 = 1.5$. Density and viscosity are reconstructed from the order parameter ϕ, and the interface is initialized as a diffuse layer of thickness $W = 4$ with surface tension $\sigma = 0.03$. Capillary effects do not affect the steady velocity profile due to the planar interface.

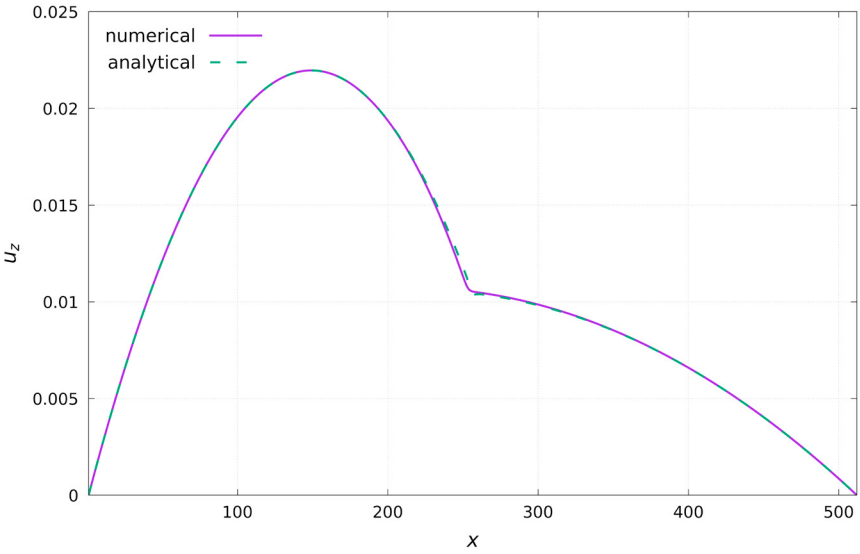


Fig. 2. Comparison between the numerical and analytical velocity profiles for the layered Poiseuille flow with viscosity contrast for the D3Q27 scheme. Excellent agreement is observed across the entire channel, including the interfacial region where the slope changes due to the viscosity jump.

Figure 2 compares the numerical velocity profile obtained from the D3Q27 scheme with the analytical solution reported in [48]. The agreement is very good across the entire channel. A small deviation is observed in the interfacial region, where the slope changes due to the viscosity jump: this discrepancy is expected, as the analytical solution assumes a sharp interface, while the present model employs a diffuse-interface formulation. In particular, the largest relative error is observed at the interface, $x = 251$, and amounts to 4.18×10^{-2}, 4.18×10^{-2}, and 3.97×10^{-2} for the D3Q19, standard D3Q27, and high-order D3Q27h schemes, respectively. Overall, this benchmark confirms that the

Table 1. Strong-scaling with fixed total domain cubic box of side 512 lattice points: GLUPS values for one-component (Taylor–Green benchmark) and two-component (Laplace benchmark) systems in single and double precision.

GPU	decomp	one-component						two-component					
		single precision			double precision			single precision			double precision		
		D3Q19	D3Q27	D3Q27h	D3Q19	D3Q27	D3Q27h	D3Q19	D3Q27	D3Q27h	D3Q19	D3Q27	D3Q27h
1	1x1x1	2.3	1.8	1.0	1.4	1.0	0.4	1.6	1.4	0.8	0.8	0.7	0.2
2	1x1x2	4.4	3.5	1.9	2.6	2.0	0.8	3.0	2.6	1.6	1.5	1.2	0.5
4	1x1x4	8.2	6.6	3.6	5.0	3.9	1.6	5.7	4.8	3.0	2.9	2.5	0.9
4	1x2x2	8.2	6.6	3.6	4.9	3.9	1.6	5.6	4.8	3.0	2.9	2.4	0.9
8	1x2x4	14.1	11.6	6.8	8.7	7.0	3.1	9.3	8.2	5.4	5.1	4.3	1.8
8	2x2x2	13.8	11.4	6.7	8.6	6.9	3.0	9.1	8.0	5.3	5.0	4.3	1.7
16	1x4x4	22.3	19.1	12.0	14.7	12.1	5.8	14.1	12.6	9.1	8.4	7.4	3.3
16	2x2x4	22.2	18.8	11.9	14.5	12.0	5.7	13.5	12.4	8.9	8.4	7.2	3.2
32	1x4x8	31.5	28.2	19.6	21.4	18.5	10.0	18.7	16.9	13.3	11.8	10.8	5.6
32	2x4x4	32.4	28.5	19.8	22.3	19.2	10.2	18.6	17.2	13.4	12.3	11.1	5.7
64	1x8x8	41.7	39.0	29.0	29.9	26.4	16.8	21.9	21.6	14.9	15.7	14.5	9.1
64	4x4x4	39.0	40.1	27.7	30.2	26.3	17.2	19.7	20.5	17.5	14.4	14.5	9.2
128	1x8x16	49.5	47.6	41.5	38.5	35.4	26.4	24.4	23.6	23.4	19.3	16.5	13.2
128	4x4x8	41.4	42.2	40.4	37.9	33.8	26.5	19.8	22.0	21.6	17.8	14.8	13.1
256	1x16x16	44.8	58.5	47.6	46.1	43.9	34.3	21.6	26.1	24.7	19.5	21.3	17.0
256	4x8x8	44.3	48.5	45.1	45.1	40.7	33.8	19.1	23.9	19.0	18.1	18.0	16.8
512	1x16x32	62.3	55.9	55.1	56.1	53.9	46.9	27.3	24.8	24.5	23.4	23.2	20.7
512	8x8x8	48.1	46.0	44.5	48.3	47.4	42.7	21.8	19.4	21.5	20.4	19.0	20.5

implementation accurately reproduces the layered Poiseuille solution and correctly captures viscosity contrasts within the diffuse-interface framework.

In addition to the layered Poiseuille configuration, we performed a Laplace test to assess the accuracy of surface tension and isotropy for the different velocity sets. A spherical droplet of radius $R = 64$ lattice units is initialized at the center of a cubic domain of size 512^3, ensuring negligible finite-size effects. The two phases are characterized by equal densities and viscosities, $\rho_1 = \rho_2 = 1$ and $\nu_1 = \nu_2 = 0.1$, while the interface is described by a diffuse profile of thickness $W = 4$ and surface tension $\sigma = 3.000 \times 10^{-2}$.

At equilibrium, the pressure jump across the interface is expected to follow the Laplace law, $\Delta P = 2\sigma/R$. The numerical results obtained with the three stencils are in very good agreement with the theoretical prediction. In particular, the measured surface tension is $\sigma = 2.796 \times 10^{-2} \pm 9.976 \times 10^{-4}$, $\sigma = 2.796 \times 10^{-2} \pm 9.979 \times 10^{-4}$, and $\sigma = 2.796 \times 10^{-2} \pm 9.979 \times 10^{-4}$ for the D3Q19, standard D3Q27, and high-order D3Q27h schemes, respectively. The corresponding pressure jumps and radii are also consistent across all cases, with only negligible differences within statistical uncertainty. These results indicate that all three discretizations provide a consistent estimate of the macroscopic surface tension within statistical uncertainty for this static configuration.

5. Performance Analysis

In this section, we assess the performance and energy efficiency of the present implementation on large-scale GPU systems. All simulations have been carried out on the *Leonardo* supercomputer at CINECA [42]. Each Leonardo compute node is equipped with four NVIDIA A100 GPUs, each with 64 GB of HBM2 memory; hence, the largest calculations reported in this work, using 512 GPUs, correspond to 128 compute nodes. The code is compiled with the NVIDIA HPC SDK and executed using CUDA-aware MPI.

Performance is quantified in terms of GLUPS (Giga Lattice Updates Per Second), defined as the total number of lattice updates performed per second, expressed in billions.

$$\text{GLUPS} = \frac{L_x L_y L_z}{10^9 t_s}, \tag{12}$$

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. This metric provides a direct measure of the solver throughput and is used throughout this section to compare

Table 2. Weak-scaling with fixed sub-domain cubic box of side 512 lattice points: GLUPS values for one-component (Taylor–Green benchmark) and two-component (Laplace benchmark) systems in single and double precision.

GPU	decomp	one-component						two-component					
		single precision			double precision			single precision			double precision		
		D3Q19	D3Q27	D3Q27h	D3Q19	D3Q27	D3Q27h	D3Q19	D3Q27	D3Q27h	D3Q19	D3Q27	D3Q27h
1	1x1x1	2.3	1.8	1.0	1.3	1.0	0.4	1.6	1.4	0.8	0.8	0.7	0.2
2	1x1x2	4.5	3.6	1.9	2.6	2.1	0.8	3.2	2.7	1.6	1.6	1.3	0.5
4	1x1x4	9.0	7.1	3.8	5.3	4.1	1.7	6.4	5.3	3.2	3.1	2.6	0.9
4	1x2x2	9.0	7.0	3.8	5.2	4.1	1.6	6.3	5.3	3.2	3.1	2.6	0.9
8	1x1x8	17.7	14.0	7.5	10.3	8.1	3.3	12.5	10.5	6.3	6.2	5.1	1.9
8	1x2x4	17.4	13.8	7.4	10.2	8.0	3.3	12.2	10.2	6.3	6.0	5.0	1.8
8	2x2x2	17.4	13.8	7.4	10.1	8.0	3.3	12.1	10.2	6.3	6.0	5.0	1.8
16	1x1x16	35.3	27.8	14.9	20.7	16.2	6.6	24.9	20.8	12.7	12.3	10.2	3.7
16	1x4x4	34.7	27.6	14.8	20.2	15.9	6.5	24.3	20.3	12.5	12.0	10.0	3.6
16	2x2x4	33.6	26.7	14.6	19.4	15.4	6.4	23.3	19.8	12.3	11.5	9.6	3.6
32	1x1x32	70.9	55.9	29.9	41.2	32.4	13.1	49.6	41.6	25.2	24.6	20.3	7.4
32	1x4x8	67.9	54.2	29.5	39.3	31.2	12.7	47.3	39.9	24.7	23.4	19.5	7.1
32	2x4x4	66.7	53.3	29.2	38.4	30.6	12.8	46.1	39.1	24.3	22.9	19.2	7.1
64	1x1x64	140.7	111.7	59.9	82.3	64.5	26.0	98.3	83.6	50.6	48.9	40.8	14.8
64	1x8x8	134.8	107.7	58.7	78.4	62.3	25.7	93.8	79.6	49.3	46.7	39.1	14.3
64	4x4x4	132.2	106.2	58.3	76.7	60.8	25.3	91.2	78.2	48.6	45.7	37.9	14.1
128	1x1x128	278.8	223.2	119.6	164.8	128.9	52.1	196.7	165.9	100.8	97.8	81.0	29.7
128	1x8x16	268.0	215.9	117.7	157.2	124.4	51.5	187.4	157.9	98.3	93.0	77.2	28.5
128	4x4x8	256.8	208.7	115.2	150.2	119.9	50.7	179.2	152.4	95.9	88.8	74.8	28.3
256	1x1x256	564.3	447.5	239.5	329.8	258.7	104.4	393.8	332.2	201.9	195.1	162.0	59.3
256	1x16x16	536.0	428.7	234.7	311.1	247.5	102.4	370.9	317.6	196.2	184.9	154.4	56.9
256	4x8x8	512.6	413.1	230.3	296.1	237.5	101.1	351.7	303.1	191.2	176.7	148.7	57.5
512	1x1x512	1131.4	890.2	479.2	658.9	517.3	209.4	767.8	641.1	386.3	379.3	312.6	115.6
512	1x16x32	1001.5	797.7	430.5	582.2	457.3	191.5	709.3	596.2	362.7	351.5	290.8	111.2
512	8x8x8	960.5	770.7	421.1	556.2	440.6	187.1	674.2	566.5	353.4	334.3	277.2	108.2

different configurations. To evaluate scalability, we also introduce the parallel efficiency, defined as

$$\mathrm{PE}(N) = \frac{\mathrm{GLUPS}(N)}{N\,\mathrm{GLUPS}(1)}, \tag{13}$$

where GLUPS(N) is the performance obtained using N GPUs.

A direct performance comparison with the standard population-based formulation was already reported in our previous lightweight LBM study [40]. There, a two-component D3Q19 color-gradient benchmark in single precision on an NVIDIA Tesla V100 was used to compare the population-free lightweight implementation with the corresponding standard CGLB formulation. The lightweight version [40] reached about 0.622 GLUPS, compared with 0.277 GLUPS for the population-based implementation, yielding a speed-up of about 2.25. The same test showed a reduction of the GPU memory footprint from 1.1411 GB to 0.6782 GB, corresponding to about 40% memory saving.

The present work builds on those single-GPU results and addresses the complementary problem of deploying the lightweight moment-represented strategy in a distributed-memory multi-GPU setting. In *LBFAST*, the reduced moment fields are stored using a flip–flop double-buffering layout, while full population arrays are never stored in global device memory. The analysis below therefore focuses on numerical validation, MPI communication, one-, two-, and three-dimensional domain decomposition, and strong and weak scaling up to 512 GPUs.

Strong- and weak-scaling results are reported in Tables 1 and 2 for both the single-component (Taylor–Green) and two-component (Laplace) benchmarks as introduced in Section 4, considering different velocity sets and numerical precisions.

In strong scaling, the solver exhibits good parallel efficiency up to several hundred GPUs, with a gradual saturation at the largest scales. This behavior is mainly due to the increasing impact of communication costs as the local sub-domain size decreases. The choice of the velocity set significantly affects performance: D3Q19 achieves the highest GLUPS, followed by D3Q27, while D3Q27h is consistently more expensive due to the larger number of degrees of

freedom and reconstruction operations. The same trend is observed in both single- and two-component systems and in both precisions.

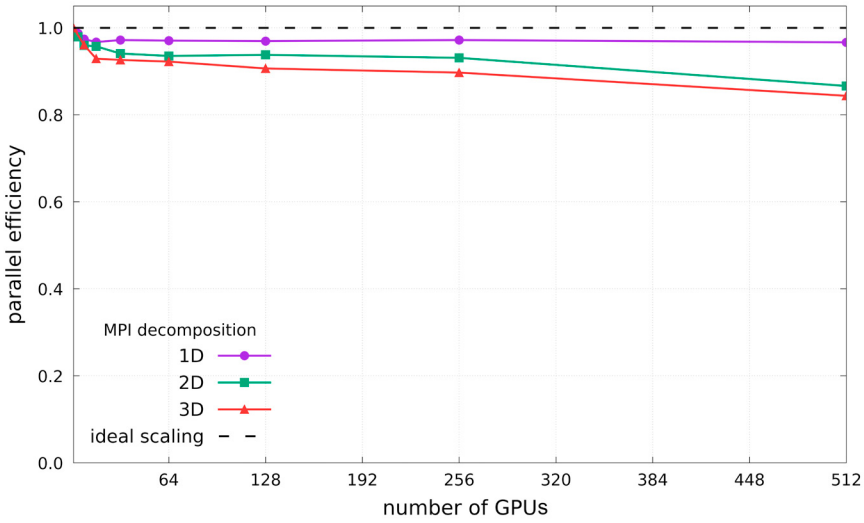


Fig. 3. Weak scaling parallel efficiency for the single-component Taylor–Green benchmark (Section 4) using the D3Q27 velocity set. Results are shown for different MPI decompositions (1D, 2D, 3D), reporting for each GPU count the best-performing MPI configuration for the 2D and 3D decompositions. The dashed line indicates ideal scaling.

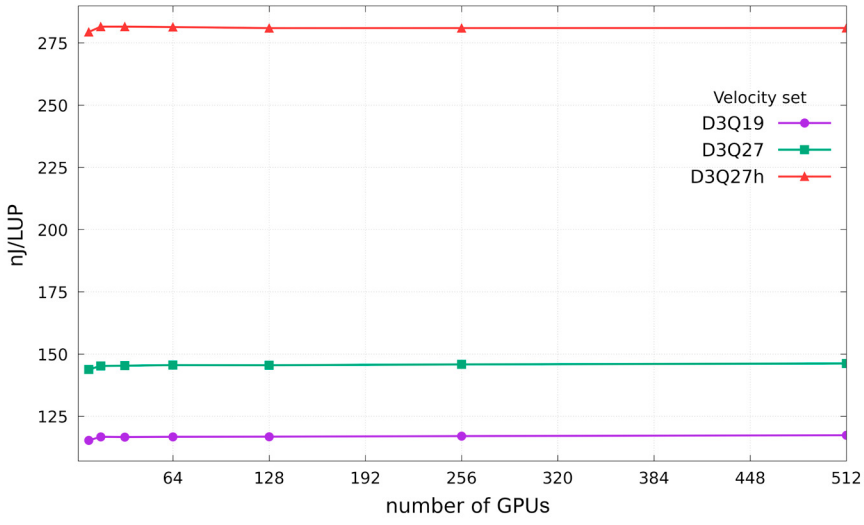


Fig. 4. Energy per lattice update (nJ/LUP) in weak scaling for the single-component Taylor–Green benchmark (Section 4), using a 3D MPI decomposition. Comparison among different velocity sets (D3Q19, D3Q27, and D3Q27h).

In weak scaling, the code remains close to ideal up to 512 GPUs for all configurations, with only a modest loss of efficiency at the largest scales, suggesting that communication is largely hidden by computation. Figure 3 shows the parallel efficiency for the D3Q27 case across different MPI decompositions. The 1D layout performs slightly better over the whole range, while 2D and 3D decompositions show a more visible, though still limited, degradation as the number of GPUs increases. This trend reflects the growing communication burden associated with higher-dimensional layouts: in the present setup, each subdomain exchanges data with two neighbors in 1D, eight in 2D, and up to twenty-six in 3D, resulting in progressively higher communication costs.

The comparison between one- and two-component systems reveals a systematic reduction in GLUPS for the latter. This is expected, as the phase-field evolution and the computation of interfacial forces introduce additional memory accesses and arithmetic operations. The performance gap becomes more pronounced in double precision, where memory bandwidth limitations are more severe.

Energy consumption has been measured using the NVIDIA Management Library (NVML), by computing the difference of the cumulative energy counter provided by `nvmlDeviceGetTotalEnergyConsumption` between the beginning and the end of each run. Based on this, we define the energy per lattice update as

$$\mathrm{nJ/LUP} = \frac{E_{\mathrm{tot}}}{N_{\mathrm{LUP}}}, \tag{14}$$

where E_{tot} is the total energy consumed (in nanojoule) and $N_{\mathrm{LUP}} = L_x L_y L_z N_t$, with N_t the number of time steps.

The behavior of this metric in weak scaling is reported in Figure 4 for several velocity sets. The results show that the energy per lattice update remains approximately constant as the number of GPUs increases, indicating that the solver preserves its energy efficiency at scale. As for performance, the velocity set plays a key role: D3Q19 is the most energy-efficient configuration, while D3Q27h exhibits the highest energy cost per update, reflecting the increased computational workload associated with the larger number of degrees of freedom and reconstruction operations.

Overall, the results indicate that the present implementation achieves good scalability on modern GPU architectures, while maintaining a stable balance between performance and energy consumption across a wide range of problem sizes.

We finally position the present performance results with respect to state-of-the-art standard LB solvers for multiphase flows on GPU infrastructures.

In the class of distribution-function-based multiphase LBM solvers, accLB [23] provides a relevant multi-GPU reference. It couples a conservative Allen–Cahn phase-field formulation with a regularized LBM and relies on hybrid MPI–OpenACC parallelism. The accLB code demonstrates strong and weak scaling up to 64 GPUs, reaching sustained throughputs above 150 GLUPS for a 512^3 lattice domain; on a single NVIDIA A100 GPU, it achieves 2.58 GLUPS in single precision. Another state-of-the-art multiphase LBM reference is the conservative Allen–Cahn implementation of Holzer et al. [14], integrated into the waLBerla framework through automatic code generation. Their solver targets immiscible fluids at high density ratios, employs optimized generated CPU/GPU kernels, and exhibits excellent weak scaling on Piz Daint up to 2048 GPUs, with a reported parallel efficiency of about 98%. In particular, Holzer et al. [14] reported 2.66 GLUPS for the phase-field kernel and 1.36 GLUPS for the hydrodynamic kernel on a single NVIDIA Tesla V100 in double precision.

However, a quantitative one-to-one comparison among these solvers is not straightforward, since they differ not only in hardware and programming model, but also in the underlying LB formulation: accLB [23] uses a regularized population-based approach, whereas the waLBerla implementation of Holzer et al. [14] relies on distribution-function-based Allen–Cahn LB steps with BGK/MRT-type collisional dynamics. These differences affect memory traffic, arithmetic intensity, number of stored fields, stencil choice, and communication volume, and therefore make raw GLUPS values only partially comparable.

More broadly, the present work also belongs to the current effort of moving CFD solvers to GPU-accelerated architectures. Relevant examples include high-performance DNS/CFD codes such as AFiD [47], STREAmS [1], CaNS [4], FluTAS [5], CaNS-Fizzy [26], the pseudo-spectral DNS solver ported by A. Roccon in Ref. [35], and URANOS-2.0 [6], as well as GPU-oriented LBM codes such as TLBfind [34, 33] and LBcuda [2]. These codes define the broader GPU-CFD landscape, but they are not direct algorithmic counterparts of *LBFAST*, since they generally rely on different discretizations, physical models, or implementation strategies.

6. Conclusions

In this work, we have presented *LBFAST*, a lattice Boltzmann solver designed to efficiently implement the lightweight, moment-represented formulation on modern GPU-based architectures. By avoiding the storage of full population arrays and reconstructing the distributions on the fly from a reduced set of variables, the method significantly reduces the memory footprint while preserving the accuracy of the hydrodynamic description. This feature is

particularly relevant for large three-dimensional simulations, where memory bandwidth and device memory capacity are the primary limiting factors.

The numerical benchmarks considered in this study, including the viscous decay of the Taylor–Green vortex and the Laplace test for multicomponent systems, confirm the correctness and robustness of the implementation. In both cases, the solver reproduces the expected physical behavior across a wide range of parameters, demonstrating that the lightweight formulation can be effectively extended to complex flows without compromising stability.

From the performance standpoint, the code exhibits good strong- and weak-scaling properties on the Leonardo supercomputer, maintaining high throughput up to several hundreds of GPUs. The results highlight the importance of the MPI domain decomposition, with multi-dimensional layouts providing a clear advantage at large scales. The comparison among different velocity sets shows a consistent trade-off between computational cost and physical fidelity, with higher-order models requiring additional operations but remaining competitive in terms of scalability.

Energy efficiency has been assessed through the energy per lattice update, showing that the solver preserves a stable energy footprint in weak scaling. This indicates that the increase in parallelism does not introduce additional overheads in terms of energy consumption per unit of work, confirming the effectiveness of the communication–computation overlap and the overall balance of the implementation.

Overall, the present results demonstrate that the lightweight lattice Boltzmann approach provides a viable and efficient framework for large-scale simulations of fluid flows on GPU-accelerated systems. The combination of reduced memory usage, good scalability, and stable energy efficiency makes *LBFAST* a promising tool for next-generation high-performance applications, including multicomponent and thermally fluctuating systems.

Data availability

The *LBFAST* source code is publicly available at https://github.com/lauricella/LBFAST.git. The software is released under the Non-Commercial Research License – Version 1.0, allowing use, copy, and modification for research, educational, and other non-commercial purposes only.

Acknowledgements

A.M. and M.L. acknowledge funding from the Italian Government through the PRIN grant MOBIOS (ID: 2022N4ZNH3, CUP: F53C24001000006). M.L. and A.T. acknowledge support from GNFM-INdAM. M.L. and S.S. acknowledge support from the European Research Council (ERC Proof of Concept Grant No. 101187935, LBFAST). We acknowledge CINECA for the computing resources provided through the ISCRA-B project MIPLAST (HP10BZY7BK) on the Leonardo Booster system, which is part of the EuroHPC Joint Undertaking (EuroHPC JU) infrastructure.

References

[1] Bernardini, M., Modesti, D., Salvadore, F., Pirozzoli, S., 2021. Streams: A high-fidelity accelerated solver for direct numerical simulation of compressible turbulent flows. Computer Physics Communications 263, 107906.

[2] Bonaccorso, F., Lauricella, M., Montessori, A., Amati, G., Bernaschi, M., Spiga, F., Tiribocchi, A., Succi, S., 2022. Lbcuda: A high-performance cuda port of lbsoft for simulation of colloidal systems. Computer Physics Communications 277, 108380.

[3] Coreixas, C., Wissocq, G., Puigt, G., Boussuge, J.F., Sagaut, P., 2017. Recursive regularization step for high-order lattice Boltzmann methods. Physical Review E 96, 033306.

[4] Costa, P., Phillips, E., Brandt, L., Fatica, M., 2021. Gpu acceleration of cans for massively-parallel direct numerical simulations of canonical fluid flows. Computers and Mathematics with Applications 81, 502–511. Development and Application of Open-source Software for Problems with Numerical PDEs.

[5] Crialesi-Esposito, M., Scapin, N., Demou, A.D., Rosti, M.E., Costa, P., Spiga, F., Brandt, L., 2023. Flutas: A gpu-accelerated finite difference code for multiphase flows. Computer Physics Communications 284, 108602.

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

[7] Dinesh Kumar, E., Sannasiraj, S.A., Sundar, V., 2019. Phase field lattice Boltzmann model for air-water two phase flows. Physics of Fluids 31.

[8] Fakhari, A., Mitchell, T., Leonardi, C., Bolster, D., 2017. Improved locality of the phase-field lattice-Boltzmann model for immiscible fluids at high density ratios. Physical Review E 96, 053301.

[9] Feng, Y., Boivin, P., Jacob, J., Sagaut, P., 2019a. Hybrid recursive regularized lattice Boltzmann simulation of humid air with application to meteorological flows. Physical Review E 100, 023304.

[10] Feng, Y., Boivin, P., Jacob, J., Sagaut, P., 2019b. Hybrid recursive regularized thermal lattice Boltzmann model for high subsonic compressible flows. Journal of Computational Physics 394, 82–99.

[11] Ferrari, M.A., de Oliveira Jr, W.B., Lugarini, A., Franco, A.T., Hegele Jr, L.A., 2023. A graphic processing unit implementation for the moment representation of the lattice Boltzmann method. International Journal for Numerical Methods in Fluids 95, 1076–1089.

[12] Gounley, J., Vardhan, M., Draeger, E.W., Valero-Lara, P., Moore, S.V., Randles, A., 2021. Propagation pattern for moment representation of the lattice Boltzmann method. IEEE Transactions on Parallel and Distributed Systems 33, 642–653.

[13] Guo, Z., Zheng, C., Shi, B., 2002. Discrete lattice effects on the forcing term in the lattice Boltzmann method. Physical review E 65, 046308.

[14] Holzer, M., Bauer, M., Koestler, H., Ruede, U., 2021. Highly efficient lattice boltzmann multiphase simulations of immiscible fluids at high-density ratios on cpus and gpus through code generation. The International Journal of High Performance Computing Applications 35, 413–427.

[15] Jacob, J., Malaspinas, O., Sagaut, P., 2018. A new hybrid recursive regularised bhatnagar–gross–krook collision model for lattice boltzmann method-based large eddy simulation. Journal of Turbulence 19, 1051–1076.

[16] Jacqmin, D., 2000. Contact-line dynamics of a diffuse fluid interface. Journal of Fluid Mechanics 402, 57–88.

[17] Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G., Viggen, E.M., 2017. The lattice Boltzmann method. Springer International Publishing 10, 4–15.

[18] Krüger, T., Varnik, F., Raabe, D., 2009. Shear stress in lattice Boltzmann simulations. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics 79, 046704.

[19] Ladd, A.J., 1994. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. Journal of Fluid Mechanics 271, 285–309.

[20] Ladd, A.J., Verberg, R., 2001. Lattice-Boltzmann simulations of particle-fluid suspensions. Journal of Statistical Physics 104, 1191–1251.

[21] Latt, J., Chopard, B., 2006. Lattice Boltzmann method with regularized pre-collision distribution functions. Mathematics and Computers in Simulation 72, 165–168.

[22] Latt, J., et al., 2007. Hydrodynamic limit of lattice Boltzmann equations. Ph.D. thesis. University of Geneva, Geneva, Switzerland. doi:10.13097/archive-ouverte/unige:464.

[23] Lauricella, M., Mukherjee, A., Brandt, L., Succi, S., Izbassarov, D., Montessori, A., 2025a. acclb: A high-performance lattice Boltzmann code for multiphase turbulence on multi-gpu architectures. Procedia Computer Science 267, 40–51.

[24] Lauricella, M., Tiribocchi, A., Succi, S., Brandt, L., Mukherjee, A., La Rocca, M., Montessori, A., 2025b. Thread-safe multiphase lattice Boltzmann model for droplet and bubble dynamics at high density and viscosity contrasts. Physics of Fluids 37.

[25] Lehmann, M., 2022. Esoteric pull and esoteric push: Two simple in-place streaming schemes for the lattice Boltzmann method on GPUs. Computation 10, 92.

[26] Lupo, G., Wellens, P., Costa, P., 2025. Cans-fizzy: A gpu-accelerated finite difference solver for turbulent two-phase flows. arXiv preprint arXiv:2502.04189 .

[27] Malaspinas, O., 2015. Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization. arXiv preprint arXiv:1505.06900 .

[28] Malaspinas, O.P., 2009. Lattice Boltzmann method for the simulation of viscoelastic fluid flows. Ph.D. thesis. EPFL. Lausanne. URL: https://infoscience.epfl.ch/handle/20.500.14299/42243, doi:10.5075/epfl-thesis-4505.

[29] Mattila, K.K., Philippi, P.C., Hegele, L.A., 2017. High-order regularization in lattice-Boltzmann equations. Physics of Fluids 29, 046103.

[30] Montessori, A., Lauricella, M., Mukherjee, A., Brandt, L., 2026. Breakdown of Kolmogorov scaling and modified energy transfer in bubble-laden turbulence. Physical Review Fluids 11, 024605.

[31] Montessori, A., Prestininzi, P., La Rocca, M., Succi, S., 2015. Lattice boltzmann approach for complex nonequilibrium flows. Physical Review E 92, 043308.

[32] Nathen, P., Gaudlitz, D., Krause, M.J., Adams, N.A., 2018. On the stability and accuracy of the bgk, mrt and rlb boltzmann schemes for the simulation of turbulent flows. Commun Comput Phys 23, 1–31.

[33] Pelusi, F., Ascione, S., Sbragaglia, M., Bernaschi, M., 2023. Analysis of the heat transfer fluctuations in the rayleigh–bénard convection of concentrated emulsions with finite-size droplets. Soft Matter 19, 7192–7201.

[34] Pelusi, F., Lulli, M., Sbragaglia, M., Bernaschi, M., 2022. Tlbfind: a thermal lattice boltzmann code for concentrated emulsions with finite-size droplets. Computer Physics Communications 273, 108259.

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

[36] Shan, X., Yuan, X.F., Chen, H., 2006. Kinetic theory representation of hydrodynamics: a way beyond the Navier–Stokes equation. Journal of Fluid Mechanics 550, 413–441.

[37] Succi, S., 2018. The lattice Boltzmann equation: for complex states of flowing matter. Oxford University Press.

[38] Succi, S., Amati, G., Bernaschi, M., Falcucci, G., Lauricella, M., Montessori, A., 2019. Towards exascale lattice Boltzmann computing. Computers & Fluids 181, 107–115.

[39] Tiribocchi, A., Durve, M., Lauricella, M., Montessori, A., Tucny, J.M., Succi, S., 2025. Lattice Boltzmann simulations for soft flowing matter. Physics Reports 1105, 1–52.

[40] Tiribocchi, A., Montessori, A., Amati, G., Bernaschi, M., Bonaccorso, F., Orlandini, S., Succi, S., Lauricella, M., 2023. Lightweight lattice Boltzmann. The Journal of Chemical Physics 158.

[41] Tran, N.P., Lee, M., Hong, S., 2017. Performance optimization of 3d lattice Boltzmann flow solver on a GPU. Scientific Programming 2017, 1205892.

[42] Turisini, M., Cestari, M., Amati, G., 2024. Leonardo: A pan-european pre-exascale supercomputer for hpc and ai applications. Journal of

large-scale research facilities JLSRF 9.

[43] Vardhan, M., Gounley, J., Hegele, L., Draeger, E.W., Randles, A., 2019. Moment representation in the lattice Boltzmann method on massively parallel hardware, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–21.

[44] Wang, Z.J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H.T., et al., 2013. High-order cfd methods: current status and perspective. International Journal for Numerical Methods in Fluids 72, 811–845.

[45] Wissocq, G., Coreixas, C., Boussuge, J.F., 2020. Linear stability and isotropy properties of athermal regularized lattice boltzmann methods. Physical Review E 102, 053305.

[46] Wissocq, G., Sagaut, P., 2022. Hydrodynamic limits and numerical errors of isothermal lattice boltzmann schemes. Journal of Computational Physics 450, 110858.

[47] Zhu, X., Phillips, E., Spandan, V., Donners, J., Ruetsch, G., Romero, J., Ostilla-Mónico, R., Yang, Y., Lohse, D., Verzicco, R., Fatica, M., Stevens, R.J., 2018. Afid-gpu: A versatile navier–stokes solver for wall-bounded turbulent flows on gpu clusters. Computer Physics Communications 229, 199–210.

[48] Zu, Y., He, S., 2013. Phase-field-based lattice Boltzmann model for incompressible binary fluid systems with density and viscosity contrasts. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics 87, 043301.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 48–57

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

High-Performance Computing for subsurface flow and transport modelling of Olkiluoto

Albert Costa-Solé[a,*], Emilie Coene[a], Gisela Martí[a], Olga Riba[a], Orlando Silva[a], Tiina Lamminmäki[b], Lasse Koskinen[b], Antti Poteri[b], Antti Joutsen[b], Veli-Matti Pulkkanen[b], Tuomo Karvonen[c], Albin Nordström[d], Guido Deissmann[e]

[a]*Amphos 21 Consulting S.L., C. Pujades 340, 08019 Barcelona, Spain*
[b]*Posiva Oy, FI-27160 Olkiluoto, Eurajoki, Finland*
[c]*Waterhope Oy, FI-00101 Helsinki, Finland*
[d]*Swedish Nuclear Fuel and Waste Management Company (SKB), Solna, Sweden*
[e]*Institute for Energy and Climate Research: Nuclear Waste Management and Reactor Safety (IEK-6) and JARA-HPC, Jülich, Germany*

Abstract

The long-term safety assessment of ONKALO®, the world's first deep geological repository for nuclear waste at Olkiluoto (Finland), requires predicting the hydro-geochemical evolution of a fractured crystalline bedrock over thousands of years. This is done by using a comprehensive/holistic hydro-geochemical reactive transport modelling methodology to add new concepts (e.g., cation exchange, cement leachates) and improve other processes in existing models (e.g., anion exclusion, matrix diffusion, surface hydrology). A significant bottleneck is the Nernst-Planck (NP) formulation used to represent multi-component diffusion processes like anion exclusion. In particular, the NP approach is numerically stiff in large-scale models with approximately 50 million degrees of freedom. This paper presents a cation exchange (CE) formulation equivalent to the NP equations that enables anion exclusion description and reduces significantly the computational cost of large-scale 3D conservative and reactive transport simulations. It also includes an analysis of the scalability of PFLOTRAN on the supercomputer LUMI for production simulations, and the study of the palaeo-hydro-geochemical evolution of the Olkiluoto site using both NP and CE approaches. The results demonstrate that a CE model can be tuned to match the NP formulation, providing a practical tool for further modelling the hydro-geochemical evolution of Olkiluoto.

Keywords: High-Performance Computing; Flow and Transport; PFLOTRAN; Nuclear Waste Repository; Anion Exclusion; Nernst-Planck; Cation Exchange

1. Introduction

Predicting the long-term hydro-geochemical evolution of a deep geological repository for spent nuclear fuel requires solving coupled non-linear equations for flow, multicomponent transport, and geochemistry across a kilometric

* Corresponding author. e-mail: albert.costa@amphos21.com

10.1016/j.procs.2026.08.018

domain over timescales of thousands to a million years. At Olkiluoto on Finland's western coast, these challenges are addressed as Posiva Oy constructs ONKALO®, the world's first final disposal repository [1].

The bedrock is Precambrian crystalline rock (predominantly migmatitic gneisses), with groundwater flow through a complex fracture network characterised by the stochastic ODFN3 Discrete Fracture Network model [2, 3]. The rock matrix (porosity $\varphi_m \approx 0.3$–1%) provides significant solute storage through matrix diffusion, and negatively charged biotite surfaces induce anion exclusion in the matrix pore water, with observed chloride depletion factors of 0.13–0.40 [4, 5]. Groundwater compositions range from fresh meteoric (Cl $\approx$ 3 mg/L) to deep brine (Cl $\approx 42,000$ mg/L) over the time and depth profile [6]. The initial condition of the palaeomodel, 8,100 years ago, consists of a depth-dependent mix of subglacial, saline, and brine waters when the site was fully submerged. The palaeo-evolution includes Littorina seawater infiltration, island emergence (~2,500 years ago) due to post-glacial land uplift (~6 mm/ years), and the development of a meteoric freshwater lens. A sub-horizontal fracture zone (SHF) along the first ~100 m depth strongly controls vertical penetration of surface waters [7].

Numerical simulations are essential for analyzing and demonstrating repository safety. Modeling dissolved sulfide, produced by microbial sulfate reduction, helps predict copper canister corrosion at repository depth (430 m) [8]. Accurately representing fracture-matrix solute exchange and anion exclusion is key, as the coupled evolution of sulfate, iron, hydrogen, and the rock's geochemical buffer all depend on them. The resulting model comprises 3 million cells, 15 transported species, and 50 million degrees of freedom (DOFs), capturing 8,100 years of palaeo-hydro-geochemical evolution with non-linear interactions between density-driven flow and reactive chemistry. High-performance-computing is a key tool to overcome hurdles arising from model complexity coupled with the fabric of the site's hydrogeology.

The OL-STAR project (*OLkiluoto STate-of-Art Reactive hydrogeochemical model*) [9] is a collaboration between Posiva Oy and Amphos 21 Consulting. It was awarded 26.1 million core-hours on the LUMI-C partition under the EuroHPC programme to develop and run PFLOTRAN models [10, 11]. A major computational bottleneck is the representation of anion exclusion in the rock matrix. The Nernst-Planck (NP) equations [12, 4] are physically rigorous but cause severe numerical stiffness; for example, a single 1D case requires about 6 hours, compared to 5 minutes for Fickian transport. To overcome this, it was developed an equivalent cation exchange (CE) formulation that reduces computational cost by ~70× while preserving essential physics. To our knowledge, this work presents for the first time a 3D hydrogeochemical NP-based model addressing anion exclusion at large scale.

2. Modelling framework

The Olkiluoto palaeo-hydro-geochemical model, Fig. 1, couples density-driven groundwater flow with multicomponent reactive transport, solved using PFLOTRAN [10, 11]. The main symbols and notation used throughout the modelling framework are summarised in Table 2.

Saturated groundwater flow follows the mass conservation equation $\frac{\partial(\varphi\rho_w)}{\partial t} + \nabla\cdot(\rho_w\mathbf{q}) = Q_w$, where φ is the porosity, ρ_w the water density, and Q_w a source/sink term. The specific discharge $\mathbf{q}$ is given by Darcy's law with intrinsic tensor permeability $\mathbf{k}$, viscosity μ, and liquid pressure P. Density-dependent flow is driven by the strong salinity gradient and is described using a linear equation of state, $\rho_w = \rho_0 + \beta_s C_s$, where $\rho_0 = 1,000\ \mathrm{kg\,m^{-3}}$ and C_s is the NaCl-equivalent salinity. The coefficient $\beta_s = 40\ \mathrm{g\,mol^{-1}}$ was obtained from a seawater reference density of $\rho_{sw} = 1,024\ \mathrm{kg\,m^{-3}}$ at $C_s = 0.6\ \mathrm{mol\,L^{-1}}$. Transport properties are evaluated at 11 °C, which is the average temperature in the site. Also, pressure effects are neglected, thus focusing on the dominant salinity effect [13].

Transport is solved using PFLOTRAN's Global Implicit Reactive Transport (GIRT) mode. In the fracture continuum, the mass conservation equation for the total concentration Ψ_j^f of each primary species j reads [4, 14]:

$$\frac{\partial(\epsilon_f\varphi_f\Psi_j^f)}{\partial t} + \nabla\cdot(\epsilon_f\mathbf{\Omega}_j^f) = -\epsilon_f\sum_s s\nu_{js}I_s^f - \mathcal{A}_{fm}\mathcal{F}_j^{fm} \tag{1}$$

where ϵ_f is the fracture volume fraction, φ_f the fracture porosity, and $\mathbf{\Omega}_j^f$ is the total flux combining advection, dispersion, and electrochemical contributions. The terms ν_{js} and I_s^f are the stoichiometric coefficient and rate of mineral reaction s, and the last term couples the two continua through the specific interfacial area $\mathcal{A}_{fm}$ (fracture surface area per bulk volume) and the mass exchange flux $\mathcal{F}_j^{fm}$. In the matrix, the same conservation equation applies but without advection ($\mathbf{q} = 0$): transport is purely diffusive, and the modelling approach (Section 2.2 or Section 2.3)

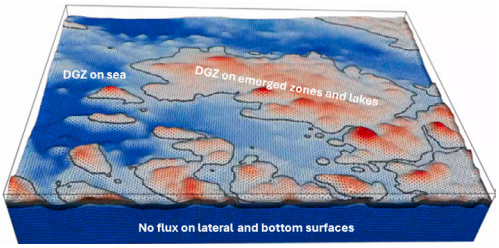


Fig. 1: Computational mesh (2.9 million primary cells) with topography colour-coded by elevation and its associated boundary conditions (DGZ: Dirichlet Zero Gradient). Cutting planes at $y = 2910$ m and $x = 3870$ m reveal bedrock heterogeneity, including fractures, with depth.

determines which diffusion equation is applied: Fickian or Nernst-Planck. Classically, diffusion is represented through Fick's law, which works well when electrochemical effects are not significant. However, in the Olkiluoto system chemical water compositions in the fracture and the rock matrix suggest anion exclusion occurs within the rock matrix. This process is inherently associated with charged mineral surfaces and can be better represented by multicomponent electrochemical diffusion. Nernst-Planck equations are a way of describing the above process [4, 12].

The following subsections describe how these equations are adapted to the dual-continuum representation of the Olkiluoto fractured rock, and how anion exclusion in the matrix is handled.

2.1. The dual-continuum conceptual model

At Olkiluoto, groundwater flows through a network of fractures within a low-permeability rock matrix. The fractures provide pathways for advective transport. The matrix, by contrast, acts as an immobile storage reservoir accessible only by diffusion. This creates a natural separation of timescales: solutes move rapidly through fractures but exchange slowly with the surrounding rock over distances of centimetres to meters. This exchange controls the long-term evolution of groundwater chemistry and sulfide concentrations at repository depth.

This physical picture is represented using a Dual Continuum Disconnected Matrix Model (DCDMM) and an Equivalent Continuum Porous Media (ECPM) approach [14, 15], in which the rock volume is partitioned into two overlapping continua. The primary (fracture) continuum carries advective and dispersive transport through a 3D mesh of 2.9 million cells (Fig. 1), with heterogeneous properties upscaled from the ODFN3 Discrete Fracture Network model [2] (Fig. 2). Fig. 2 illustrates two upscaled ECPM fields used by the dual-continuum model. The left panel shows the intrinsic permeability in the x direction, whereas the right panel shows the matrix half-length derived from the local DFN fracture statistics.

During the Olkiluoto island emergence, the top surface boundary conditions are split into three zones (Fig. 1): sea (Dirichlet with evolving Littorina-to-Baltic sea chemical composition and atmospheric pressure at the sea level), emerged land (Dirichlet with meteoric water composition and prescribed recharge), and lakes (Dirichlet with meteoric water composition and prescribed head). Lateral and bottom boundaries are set as no-flux for flow and transport. Time-dependent boundary conditions (shoreline displacement, evolving seawater composition, and heterogeneous effective recharge) are managed using an in-house Python script that updates these parameters at each time step.

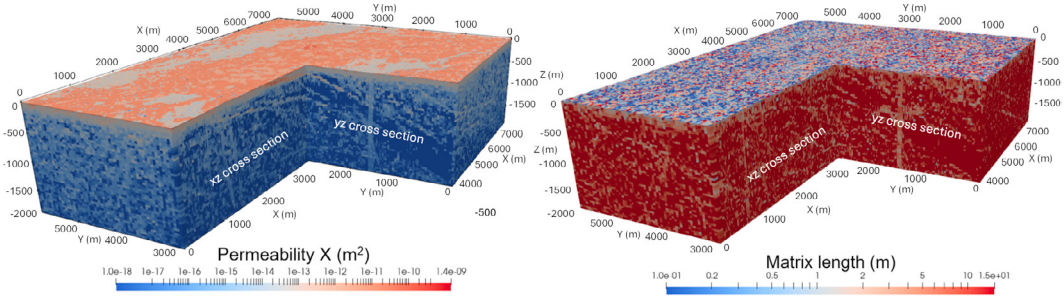


Fig. 2: ECPM properties upscaled from the ODFN3 Discrete Fracture Network model. Left: intrinsic permeability in the x-direction. Right: matrix half-length. Both fields are heterogeneous and reflect the local DFN fracture statistics.

Each fracture cell (primary continuum) is connected to a 1D line of 10 matrix cells (secondary continuum), through which solutes diffuse perpendicular to the fracture wall. In the matrix, advection is absent, and the mass balance equation for species j reduces to [14]:

$$\frac{\partial(\varphi_m \Psi_j^m)}{\partial t} + \nabla_\xi \cdot \mathbf{\Omega}_j^m = -\sum_s s\nu_{js} I_s^m \tag{2}$$

where ∇_ξ is the gradient operator in the secondary continuum coordinate ξ (distance from the fracture-matrix interface) and $\mathbf{\Omega}_j^m$ is the matrix diffusive flux, with species-dependent effective diffusion coefficient combining the matrix tortuosity and the species self-diffusion coefficient [4]. In the standard Fickian model, all species share a single diffusion coefficient; when electromigration is considered (Section 2.2), $\mathbf{\Omega}_j^m$ includes an additional electrochemical term. The fracture-matrix coupling in Eq. (1) is computed from the flux at the outer boundary of the matrix continuum (fracture-matrix interface) as $\mathcal{F}_j^{fm} = \mathbf{\Omega} j^m \cdot \mathbf{n}\big|\xi = \xi_{\text{out}}$ [4, 14]. This expression represents the continuity of total concentration at the fracture-matrix interface. In addition, zero-flux is assumed at the innermost cell in the matrix [14]. The matrix is characterised by its porosity φ_m and half-length L_m. The L_m is derived from the local DFN statistics and varying across the domain. Geochemical reactions (mineral dissolution, cation exchange, and microbial processes) occur independently in both continua. Salinity is tracked through an auxiliary tracer defined as a weighted sum of Cl^-, Na^+, Ca^{2+}, and Br^-.

2.2. Anion exclusion: the Nernst-Planck formulation

In the standard Fickian model, the matrix diffusive flux $\mathbf{\Omega}_j^m$ in Eq. (2) uses a single effective diffusion coefficient D_m for all species. However, the negatively charged surfaces of biotite and other clay minerals in the Olkiluoto matrix create a permanent electrical field that repels anions and attracts cations, reducing the effective porosity available for anion transport. In the NP formulation [4, 12], this effect is modelled by representing the mineral surface charge as a fixed concentration of immobile anions Q (eq/L). The presence of Q establishes a Donnan potential ψ_D that partitions ions between the charged and bulk porosity:

$$\xi_i = C_i^D / C_i^B = \Gamma_i \exp\left(-z_i F \psi_D / RT\right) \tag{3}$$

where z_i is the ion charge, F the Faraday constant, R the gas constant, T the absolute temperature, and the superscripts D and B denote the Donnan and bulk porosity, respectively. Because the exponential depends on z_i, divalent sulfate ($z = -2$) is excluded more strongly than monovalent chloride ($z = -1$) for the same ψ_D.

The Nernst-Planck equation replaces the Fickian flux for each individual species i with a coupled electrochemical flux [4]:

$$\mathbf{J} i^m = -\varphi_m D_i^m \left(\nabla \xi c_i - z_i c_i \frac{\sum_j D_j^m z_j \nabla_\xi c_j}{\sum_k z_k^2 D_k^m c_k} \right) \tag{4}$$

where D_i^m is the species-dependent pore diffusion coefficient in the matrix (product of matrix tortuosity and self-diffusion coefficient in unconstrained solution). The first term is Fickian diffusion; the second couples all charged species through an electromigration term that enforces electroneutrality. Physically, when a fast-diffusing ion such as Cl^- moves ahead of a slower counter-ion such as Ca^{2+}, an electrical potential builds up that retards the faster ion and accelerates the slower one. Numerical evaluation requires logarithmic averaging at cell interfaces [16] to avoid artefacts for asymmetric electrolytes. The resulting system is stiff, due to a 1D case of a fracture at 200 m depth requires approximately 6 hours, compared to 5 minutes for the equivalent Fickian model, making the direct application of NP in 3D reactive transport impractical without supercomputing.

2.3. The cation exchange formulation

To make anion exclusion computationally tractable at the site scale, it was developed a CE formulation that reproduces the macroscopic effect of the NP equations without replacing the Fickian flux. Instead, it modifies the storage terms in the matrix transport equation (Eq. (2)).

The central parameter is the anion-accessible porosity fraction, $f_a = \sum_i c_i^m |z_i| / \sum_i c_i^f |z_i|$ (5), defined as the ratio of charge-weighted anion concentrations in the matrix (c_i^m) to those in the fracture (c_i^f), with z_i the ion charge. This definition follows directly from the Donnan equilibrium (Eq. (3)): the partitioning of each anion depends on the Donnan potential, and the charge-weighted average yields an effective macroscopic f_a. Because the Donnan potential

Table 1: Scalability on Jureca [17] and LUMI. DOFs is DOFs per task. Bold: production.

Nodes	Tasks	LUMI	JURECA	DOFs	Efficiency
4 nodes	512	✓	✓	97 656	1.00
8 nodes	1 024	✓	✓	48 828	0.94
16 nodes	2 048	✓	✓	24 414	0.79
32 nodes	**4 096**	✓	×	**12 207**	**0.65**
64 nodes	8 192	✓	×	6 104	0.42
256 nodes	16 384	✓	×	3 052	0.22

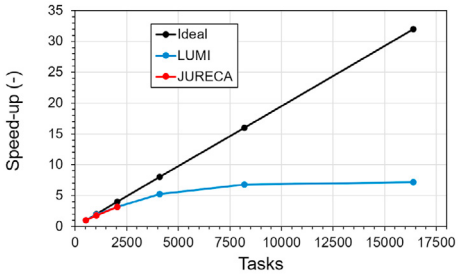


Fig. 3: Scalability for the palaeo-evolution (~50 M DOFs).

depends on the ionic strength relative to the fixed charge Q, f_a is not a material constant but varies with the local salinity: at high ionic strengths, anion exclusion is weak and f_a approaches unity; in dilute waters, f_a decreases substantially.

Using f_a, the matrix porosity is partitioned into an anion-accessible volume and a complementary volume where cations are retained on exchange sites that compensate the mineral surface charge. The overall matrix concentration of an anion is $c_i^m = f_a \bar{c}_i^m$, being $\bar{c}_i^m$ the concentration in the anion accessible porosity. For an exchangeable cation, it includes the exchanger contribution as $c_i^m = f_a \bar{c}_i^m + \mathrm{CEC} \cdot N_i$, where CEC is the cation exchange capacity and N_i is the equivalent fraction on the exchange sites.

The equivalence between the CE and NP approaches has been established through the f_a parameter, which has been estimated with the NP-predicted anion distribution ratio (Eq. (3)). A known limitation is that f_a is charge-independent: unlike NP, which predicts stronger exclusion for divalent anions, the CE approach applies the same accessible fraction to all anions. Despite this simplification, the CE approach captures the dominant effect: the overall reduction in anion mass transfer between fractures and the matrix, which is the primary control on the chloride and sulfate profiles used for model calibration. A 1D proxy model confirmed that CE and NP produce practically identical chloride profiles, while full NP predicts somewhat stronger divalent exclusion. The 1D CE model runs in approximately 5 minutes, compared with approximately 6 hours for NP, resulting in a roughly 70-fold cost reduction.

3. HPC deployment on LUMI

The Olkiluoto palaeomodel (2.9 million primary cells with 10 secondary continuum cells each, 15 chemical species plus flow, yielding ~50 million DOFs) was deployed on the LUMI-C partition, compiled with the Cray compiler environment, PETSc 3.21.5, and parallel HDF5 for I/O. Using fewer than 128 cores (1 node) is impractical for models of this size: each MPI task requires at least 500 MB of RAM for the secondary continuum arrays alone, and wall-clock times on a single node would extend to weeks.

The model was tested from 512 to 16,384 MPI tasks (Table 1, Fig. 3) by simulating the submerged period of the palaeo-evolution of Olkiluoto using a semi-structured mesh, which includes matrix diffusion and reaction. We observe that scalability is governed by the ratio of DOFs per MPI process: above ~10,000 DOFs/task, the computation is dominated by local linear algebra and secondary continuum solves, both of which scale well; below this threshold, inter-process communication begins to dominate. The practical optimum was found at 4,096 tasks across 32 nodes (128 tasks/node), where the measured speed-up relative to 512 tasks is approximately 5.2, corresponding to a strong-scaling efficiency of about 65% because the number of tasks increases by a factor of eight. This configuration was adopted for all production runs. Beyond 4,096 tasks, performance degrades as DOFs/task drop to ~6,000 (8,192 tasks) and ~3,000 (16,384 tasks), at which point the communication-to-computation ratio becomes unfavourable. For comparison, the same model was tested on JURECA-DC up to 2,048 tasks with consistent results; higher configurations could not be tested in JURECA due to PETSc segmentation faults traced to library version incompatibilities, which underscores the importance of cross-platform verification for production HPC codes. At the production configuration, a single 8,100-year palaeo-hydro-geochemical simulation completes in 24–48 wall-clock hours, consuming 100,000–200,000 core-hours depending on the complexity of the reactive system. Time-to-solution is problem-dependent: conservative transport with Fick's diffusion is the fastest, the CE formulation is intermediate, and nonlinear diffusion represented by NP equations is the most expensive due to the stiffness introduced by the electromigration terms. The numerical solution relies on PETSc-based solvers. Both the flow and transport systems use Newton's method, with FGMRES and Additive Schwarz preconditioning for the transport linear system and standard GMRES for the flow system. Sep-

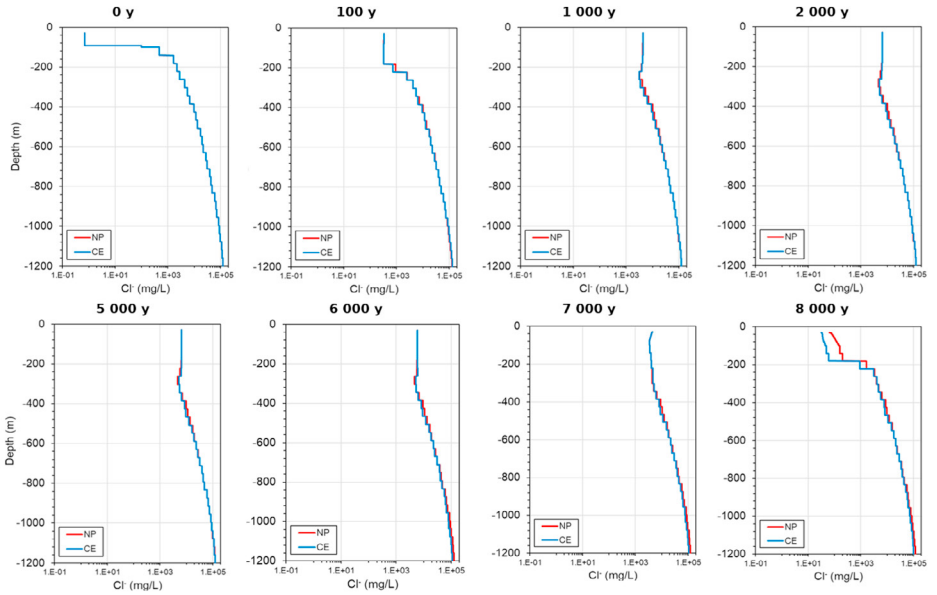


Fig. 4: Temporal evolution of chloride concentration vertical profiles. Blue: CE (f_a = 0.167); red: NP. CE parameterisation: NP with Q = 0.1 M. CE parametrization with CEC = 2.1 mol/m^3,$\phi_m = 6.3 \times 10^{-3}$.

arate convergence tolerances are enforced for the primary and secondary continuum unknowns to prevent the matrix diffusion equations from stalling global convergence. An adaptive time stepper with aggressive acceleration allows large steps during quasi-steady phases while retreating automatically during sharp transients. A complete modelling campaign (several ECPM realisations, multiple f_a sensitivity runs, and NP vs. CE comparisons) demands several million core-hours, well beyond institutional cluster capacity and justifying the EuroHPC allocation.

4. Results and discussion

This section presents the main results of the OL-STAR palaeo-hydro-geochemical conservative transport simulations in three parts. First, the equivalence between the NP and CE formulations is validated through temporal chloride profiles and a multispecies comparison at 8,100 y (Section 4.1). Second, the sensitivity of model predictions to the anion-accessible fraction f_a is evaluated against monitoring borehole data, and the robustness of the results with respect to matrix discretisation is demonstrated (Section 4.2). Third, the 3D chloride and sulfate distribution at 8,100 years is presented as vertical cross-sections through the model domain (Section 4.3).

4.1. NP–CE equivalence: temporal evolution and multispecies validation

A prerequisite for using the CE formulation in production 3D simulations is to confirm that it reproduces the NP solution across the full range of hydrogeochemical conditions encountered during the palaeo-evolution. Fig. 4 compares vertical chloride concentration profiles at the centre of the 3D domain shown in from the NP and CE (f_a = 0.167, CEC = 2.1 mol/m^3, Q = 0.1 M, $\phi_m = 6.3 \times 10^{-3}$) models at eight time snapshots. During the submerged phase, Littorina seawater progressively displaces subglacial water in the upper 400 m, while concentrations below 400 m reach quasi-steady state. After the island emergence (~7,100 years), meteoric infiltration creates a freshwater lens in the bedrock above 100–200 m depth. Throughout this evolution, the NP and CE profiles overlap almost perfectly. Divergence appears only in the latest stages (7,000–8,100 years) within the upper 200 m, where the freshwater lens produces low-salinity conditions that amplify the difference between charge-dependent NP partitioning and the CE model with constant f_a. Physically, dilution in the shallow zone reduces the ionic strength causing the NP formulation to predict a stronger Donnan exclusion than the constant f_a assumed by the CE model. As a result, the CE model allows slightly more freshwater penetration into the matrix. This behaviour is consistent with the theoretical expectation that the CE simplification becomes less accurate as salinity decreases relative to the fixed charge.

The multispecies comparison at 8 100 years (Fig. 5) extends the validation beyond chloride to all 10 transported species. The equivalent CE model was run for two values of f_a: 0.167 and 0.53. With f_a = 0.53, the CE model

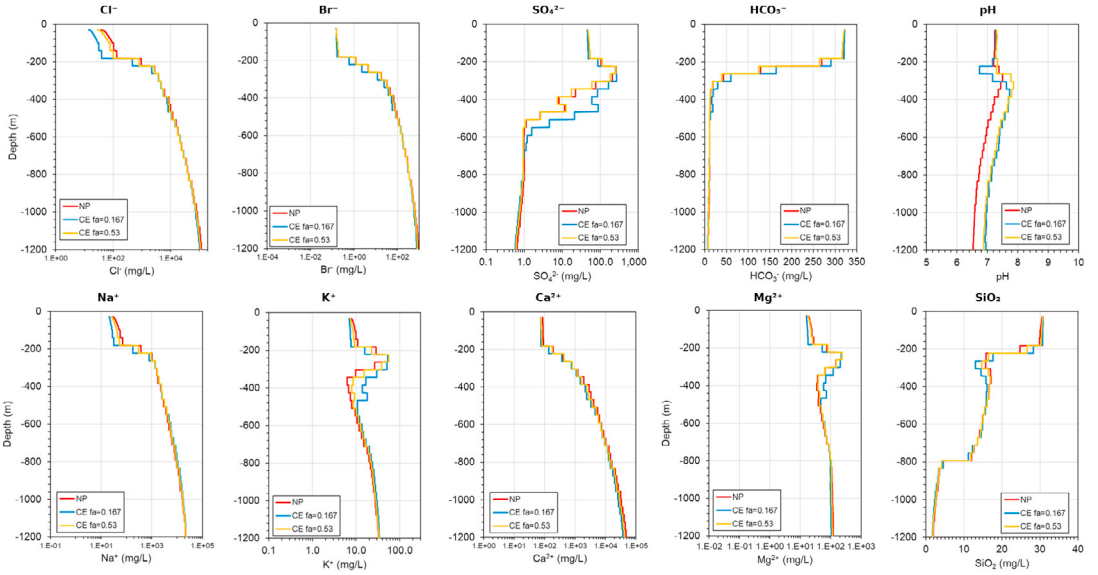


Fig. 5: 1D vertical concentration profiles of 10 species at 8,100 y: NP (red) vs. CE with $f_a = 0.167$ (blue) and $f_a = 0.53$ (yellow). CE parameterisation: CEC = 2.1 mol/m^3, $\phi_m = 6.3 \times 10^{-3}$.

closely reproduces the NP depth profiles for the monovalent anions Cl^- and Br^-, and provides a reasonable match for Na^+, K^+, Ca^{2+}, and SiO_2. The largest deviations appear for SO_4^{2-} in the Subglacial-Saline zone (200–600 m), where the charge-independent f_a of the CE formulation cannot reproduce the stronger divalent exclusion predicted by NP (Eq. (3)). Mg^{2+} and HCO_3^- also show notable differences in the upper 400 m, while pH is sensitive to the choice of f_a only locally, and both CE simulations produce different results than NP. Using $f_a = 0.167$ generally shifts all CE profiles further from the NP reference, most visibly for the anions in the shallow zone where anion exclusion is strongest. This illustrates that the choice of f_a has a quantifiable impact on predicted concentrations and must be treated as a calibration target if a comparison with NP is performed.

The combination of the CE formulation and High-Performance-Computing resources is the key enabling factor demonstrated here. Before this work, the NP approach was limited to 1D case models requiring hours per run; the CE approach reduces the per-simulation cost by ~70×, to the point where LUMI can deliver production 3D runs in 24–48 h and support systematic sensitivity analysis across the multi-dimensional parameter space (φ_m, f_a, Q).

4.2. Sensitivity to f_a, comparison with field data, and matrix discretisation

The anion accessible porosity fraction f_a (Eq. (5)) is the central calibration parameter of the CE formulation. As discussed in Section 2.3, f_a is not an intrinsic material parameter but depends on the local salinity through the Donnan potential. The NP-derived values in our simulations range from 0.41 near the surface (where meteoric dilution is strongest) to 0.90 at depth (where brine salinities dominate). This depth dependence is the fundamental reason why a single constant f_a cannot perfectly reproduce the NP solution at all depths, and why f_a must be calibrated against field data. The ability to quantify this effect through systematic 3D sensitivity analysis represents a practical advance for the safety case.

Fig. 6 shows the f_a sensitivity at 8,100 years for chloride, compared against monitoring borehole data. Chloride (top row), as a monovalent anion, is sensitive to f_a primarily in the upper 400 m, where the contrast between the dilute meteoric lens and the deeper saline waters amplifies the effect of anion exclusion on matrix–fracture exchange. It can be observed that f_a does not impact the concentration fields significantly. In general, the model shows high infiltration of meteoric water. This effect is mitigated for higher fractions of anion accessible porosity. The effect is mainly visible at low f_a values, beyond $f_a = 0.4$ results are comparable. The regions with higher concentrations correspond to submerged areas, where there is no recharge. Because the fracture–matrix solute exchange in the DCDMM depends on the size of the cell adjacent to the fracture-matrix interface, we tested the sensitivity of the results to the number of matrix cells (Fig. 7). Three configurations were compared: 5 cells (coarse mesh), 10 cells (base case), and 20 cells (fine mesh), all using $f_a = 0.9$ and the same CE parameterisation ($Q = 0.1$ M, $f_m = 6.3 \times 10^{-3}$). The depth profiles

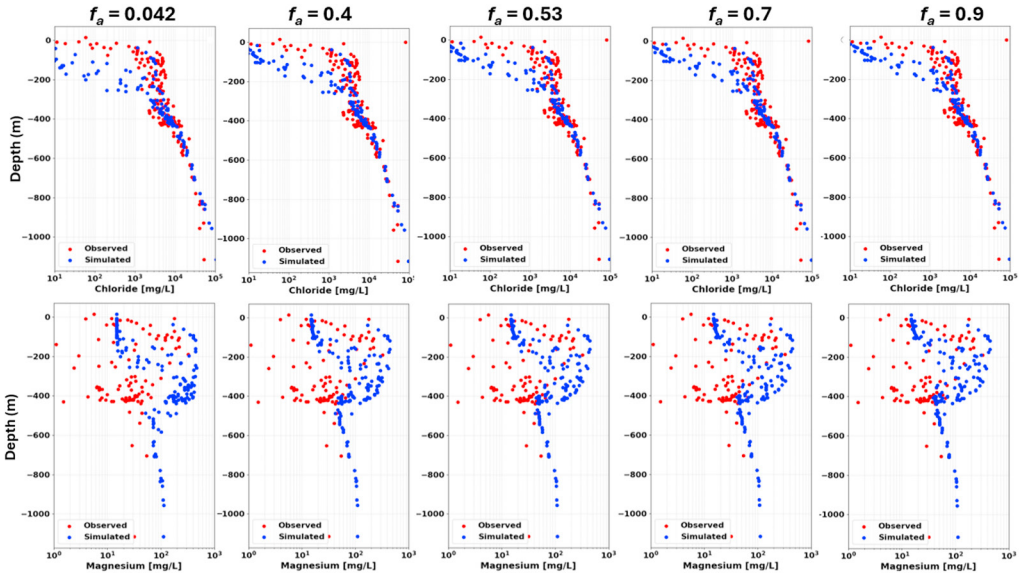


Fig. 6: Sensitivity of the model to anion accessible porosity fraction. Vertical concentration profiles of chloride (top) and magnesium (bottom) at 8, 100 years. Red: observed borehole data; blue: simulated.

of all 10 species are essentially insensitive to the matrix discretisation, with minor differences confined to the vicinity of sharp transport fronts. This confirms that the base-case discretisation of 10 matrix cells adequately resolves the diffusion gradients and that the model results are numerically robust.

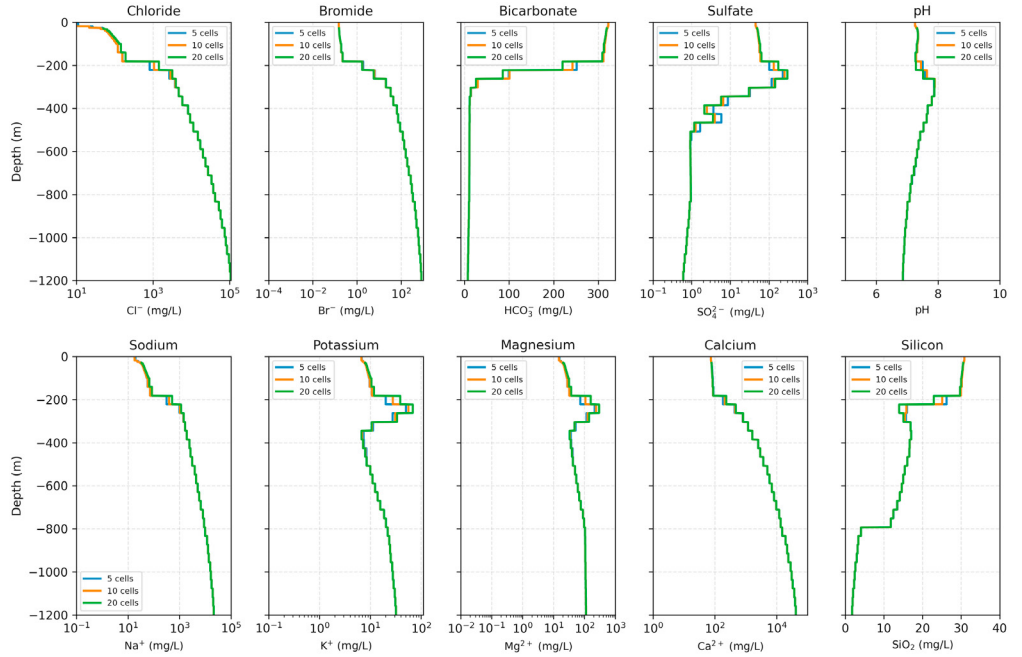


Fig. 7: Sensitivity of the model to the spatial matrix length discretisation (5, 10 and 20 cells). Concentration profiles at 8, 100 years.

4.3. 3D chloride distribution at 8,100 years

Chloride and sulfate are key species controlling density-driven flow and sulfide formation, respectively. The spatial distributions of these species at 8,100 years are shown in Fig. 8 as vertical cross-sections through the 3D CE model

domain. The xz section (at $y = 2,910$ m) and yz section (at $x = 3,870$ m) show the expected hydrogeochemical palaeo-evolution at Olkiluoto: a shallow freshwater lens (Cl $<$ 100 mg/L) resulting from post-emergence meteoric infiltration, a brackish Littorina-derived zone at intermediate depth (100–400 m), and deep saline-to-brine waters (Cl $>$ 10,000 mg/L) that have remained largely undisturbed since before 8,100 years. The freshwater penetration is deepest beneath topographic elevations and above the sub-horizontal fracture zone at ~140 m, consistent with the enhanced horizontal connectivity in this zone. These cross-sections demonstrate that the CE model, calibrated against depth profiles and monitoring data, reproduces the expected large-scale solute architecture at the full 3D site scale.

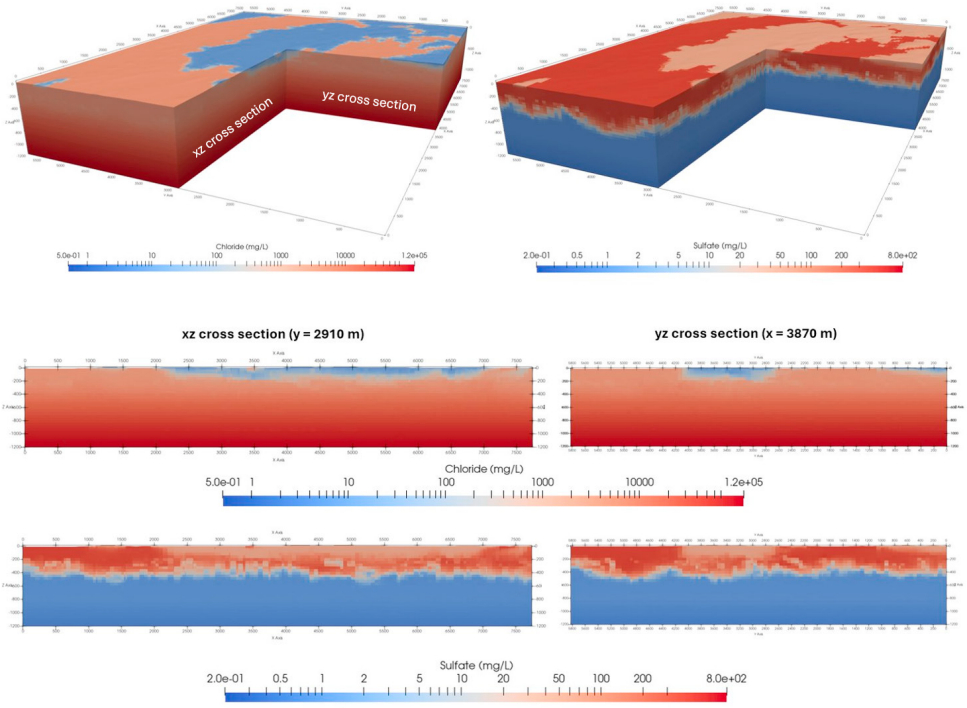


Fig. 8: Chloride and sulfate concentration distributions at 8,100 y through the CE model ($f_a = 0.9$).

5. Conclusions

This work demonstrates that the combination of a cation exchange modelling approach (CE) for anion exclusion and supercomputing resources enables 3D site-scale conservative and reactive transport modelling that was previously unfeasible for nuclear waste safety assessment. The approach was applied to simulate the hydrogeochemical palaeo-evolution of Olkiluoto site selected to build the first deep nuclear waste storage repository of the world.

The CE formulation, validated here against the full Nernst-Planck (NP) equations in both temporal and multispecies 3D configurations, reduces drastically the computational cost while preserving the essential physics. The NP-derived anion accessible porosity fraction f_a (Eq. (5)) provides a calibration target that links the CE parametrization to the underlying Donnan partitioning mechanism. All the presented results were mainly computed using the LUMI-C partition with 4,096 MPI tasks (approximately 65% strong-scaling efficiency relative to 512 tasks, and approximately 50 million degrees of freedom). Each 8,100-year palaeo-hydro-geochemical simulation completes in 24–48 wall-clock hours. This performance makes systematic sensitivity analysis and multi-realisation studies operationally viable. The simulations reproduce the observed groundwater stratification at Olkiluoto, including the freshwater lens, comparable to the spatial variability in the monitoring data.

These results establish a validated modelling foundation for the next phase of the OL-STAR project, which will focus on three objectives: (i) automated calibration across several ECPM stochastic realisations using the Optuna optimisation framework; (ii) integration of chemical reactions including microbially mediated sulfide production into the 3D reactive transport framework; and (iii) long-term post-closure simulations under future climate scenarios.

Code and data availability

The availability of datasets is subject to project confidentiality. PFLOTRAN is an open source code.

Nomenclature

Table 2: Nomenclature used in the governing equations.

Symbol	Meaning	Units	Symbol	Meaning	Units
$\varphi, \varphi_f, \varphi_m$	Total, fracture and matrix porosity	–	$\mathcal{F}_j^{fm}$	Fracture–matrix exchange flux	$\mathrm{mol\,m^{-2}\,s^{-1}}$
ρ_w, ρ_0	Water density and reference density	$\mathrm{kg\,m^{-3}}$	ξ	Matrix-continuum coordinate	m
$\mathbf{q}$	Darcy specific discharge	$\mathrm{m\,s^{-1}}$	L_m	Matrix half-length	m
Q_w	Water source/sink term	$\mathrm{kg\,m^{-3}\,s^{-1}}$	D_i^m	Matrix pore diffusion coefficient	$\mathrm{m^2\,s^{-1}}$
$\mathbf{k}$	Intrinsic permeability tensor	$\mathrm{m^2}$	z_i	Ionic charge of species i	–
μ	Dynamic viscosity	Pa s	F	Faraday constant	$\mathrm{C\,mol^{-1}}$
P	Liquid pressure	Pa	R	Gas constant	$\mathrm{J\,mol^{-1}\,K^{-1}}$
Ψ_j	Total concentration of species j	$\mathrm{mol\,m^{-3}}$	T	Absolute temperature	K
$\mathbf{\Omega}_j$	Total solute flux of species j	$\mathrm{mol\,m^{-2}\,s^{-1}}$	ψ_D	Donnan potential	V
ϵ_f	Fracture volume fraction	–	Q	Fixed immobile charge concentration	$\mathrm{eq\,L^{-1}}$
ν_{js}	Stoichiometric coefficient	–	f_a	Anion-accessible porosity fraction	–
I_s	Mineral reaction rate	$\mathrm{mol\,m^{-3}\,s^{-1}}$	CEC	Cation exchange capacity	$\mathrm{mol\,m^{-3}}$
$\mathcal{A}_{fm}$	Fracture–matrix interfacial area	$\mathrm{m^{-1}}$	N_i	Equivalent fraction on exchange sites	–

Acknowledgements

We acknowledge EuroHPC JU for awarding the project ID EHPC-REG-2025R02-204 and EHPC-REG-2022R03-225 access to the LUMI-C partition hosted by CSC – IT Center for Science Ltd. (Finland). The authors gratefully acknowledge computing time on the supercomputer JURECA at Forschungszentrum Jülich (Germany) under grant no. JIEK63. The OL-STAR project is funded by Posiva Oy.

References

[1] Posiva Oy, Olkiluoto Site Description 2018, Technical Report POSIVA 2021-10, Posiva Oy, Eurajoki, Finland, 2021.
[2] L. Hartley, P. Appleyard, S. Baxter, J. Hoek, D. Roberts, D. Swan, Hydrogeological structure model of Olkiluoto, Technical Report Posiva WR 2018-32, Posiva Oy, Eurajoki, Finland, 2018.
[3] P. Aalto, et al., Hydrogeology of Olkiluoto, Technical Report POSIVA 2021-15, Posiva Oy, Eurajoki, Finland, 2021.
[4] P. Trinchero, A. Nardi, O. Silva, P. Bruch, Simulating electrochemical migration and anion exclusion in porous and fractured media using pflotrannp, Computers & Geosciences 166 (2022) 105166.
[5] F. Eichinger, et al., Matrix pore water at Olkiluoto, Technical Report Posiva 2006-103, Posiva Oy, Eurajoki, Finland, 2006.
[6] Posiva Oy, Palaeohydrogeochemical Data for Olkiluoto, Technical Report POSIVA 2021-13, Posiva Oy, Eurajoki, Finland, 2021.
[7] T. Karvonen, Surface hydrology model of Olkiluoto, Working Report, Posiva Oy, Eurajoki, Finland, 2021.
[8] Posiva Oy, Sulfide Fluxes at Olkiluoto – 2021 Update, Technical Report POSIVA WR 2021-07, Posiva Oy, Eurajoki, Finland, 2021.
[9] Posiva Oy, Amphos 21 Consulting, OL-STAR: EuroHPC JU Final Report, Technical Report EHPC-REG-2022R03-225, EuroHPC Joint Undertaking, 2025.
[10] P. C. Lichtner, G. E. Hammond, C. Lu, S. Karra, G. Bisht, B. Andre, R. Mills, J. Kumar, PFLOTRAN user manual: A massively parallel reactive flow and transport model for describing surface and subsurface processes, Technical Report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States); Sandia . . . , 2015.
[11] G. E. Hammond, P. C. Lichtner, R. Mills, Evaluating the performance of parallel subsurface simulators: An illustrative example with pflotran, Water resources research 50 (2014) 208–228.
[12] T. Gimmi, P. Alt-Epping, Simulating donnan equilibria based on the nernst-planck equation, Geochimica et Cosmochimica Acta 232 (2018) 1–13.
[13] IOC, SCOR and IAPSO, The international thermodynamic equation of seawater – 2010: Calculation and use of thermodynamic properties, Intergovernmental Oceanographic Commission, UNESCO, 2010.
[14] A. Iraola, P. Trinchero, S. Karra, J. Molinero, Assessing dual continuum method for multicomponent reactive transport, Computers & Geosciences 130 (2019) 11–19.
[15] P. C. Lichtner, S. Karra, Modeling multiscale-multiphase-multicomponent reactive flows in porous media: Application to co2 sequestration and enhanced geothermal energy using pflotran, Computational Models for CO2 Geo-sequestration & Compressed Air Energy Storage (2014) 81–136.
[16] C. Tournassat, C. I. Steefel, T. Gimmi, Solving the nernst-planck equation in heterogeneous porous media with finite volume methods: Averaging approaches at interfaces, Water resources research 56 (2020) e2019WR026832.
[17] P. Thörnig, JURECA: Data-centric and booster modules implementing the modular supercomputing architecture at Jülich supercomputing centre, Journal of Large-Scale Research Facilities 7 (2021) A182–A182.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 58–65

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

Coupling LAMMPS with LPC3D for Mesoscopic Simulations of Electrolytes in Porous Carbons: A Python–MPI Workflow for CPU and GPU Supercomputers

El Hassane Lahrar[a,b,c,**], Mohamed Houssein Mohamed[a,c], Mathieu Salanne[a,c], Guillaume Jeanmairet[a,c], Caspar van Leeuwen[d], Céline Merlet[b,c,*]

[a] *Sorbonne Université, CNRS, Physicochimie des Électrolytes et Nanosystèmes Interfaciaux, F-75005 Paris, France*

[b] *CIRIMAT, Université de Toulouse, CNRS, 118 route de Narbonne, 31062 Toulouse, France*

[c] *Réseau sur le Stockage Électrochimique de l'Énergie (RS2E), Fédération de Recherche CNRS 3459, HUB de l'Énergie, Rue Baudelocque, 80039 Amiens, France*

[d] *SURF, Science Park 140, 1098 XG Amsterdam, The Netherlands*

Abstract

Mesoscopic simulations are useful for predicting properties of supercapacitors thanks to their ability to bridge the gap between atomistic detail and device-scale behavior, enabling the study of ion transport, charge storage, and pore-scale phenomena. In particular, mesoscopic lattice-gas simulations can access electrode length scales that remain outside the reach of fully atomistic molecular dynamics (MD) simulations due to the high computational costs of the latter. However, mesoscopic simulations require pore-resolved microscopic data that typically depend on pore size and electrolyte composition. This work presents a practical coupling between `LAMMPS`, a classical MD code with a focus on materials modeling, and the lattice-gas code `LPC3D`, in which `LPC3D` acts as the driver of an ensemble of independent `LAMMPS` simulations, one for each pore size. The coupling is implemented in Python by combining mpi4py based communicator splitting with the `LAMMPS` Python interface, enabling concurrent MD executions within a single MPI allocation. In order to parameterize pore-dependent fields for large-scale simulations of heterogeneous porous electrodes, post-processing procedures extract pore-resolved equilibrium number densities from MD trajectories and transfer these quantities to `LPC3D`. We describe deployment on high performance computing (HPC) systems in CPU-only mode (IntelMPI) and in GPU-accelerated mode (IntelMPI with Kokkos and NVIDIA backends), and we report the strong and weak scaling behaviors of the coupled workflow on multi node architectures.

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026

Keywords: porous carbons, supercapacitors, lattice-gas, LAMMPS, LPC3D, mpi4py, Kokkos, GPU acceleration.

* Corresponding author. Tel.: +33 (5) 61 55 62 84. celine.merlet@utoulouse.fr
** Corresponding author. Tel.: +33 (5) 61 55 78 70. el-hassane.lahrar@utoulouse.fr

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.019

1. Introduction

Electrochemical double-layer capacitors (EDLCs), also known as supercapacitors, are energy storage systems characterized by their high power densities, long cycle lifetimes, and wide operating temperature range [1, 2, 3, 4]. In these systems, charge storage is based on the voltage-driven electrosorption of ions from an electrolyte onto the electrode surfaces [5, 6, 7]. Most simulations of supercapacitors have been performed at the microscopic scale [8], using MD software such as `LAMMPS` [9], `ESPResSo` [10, 11] or `MetalWalls` [12, 13] providing valuable insights to the supercapacitors field. On the fundamental side, atomistic simulations on realistic structures, similar to experimental carbide-derived carbons, have revealed the microscopic mechanisms underlying charge storage and demonstrated that ions partially shed their solvation shell upon entry into sub-nanometer pores, providing a molecular level explanation for the anomalous capacitance increase reported experimentally [14]. Theoretical and Monte Carlo studies, conducted in parallel, established that the strong screening of ion-ion interactions in metallic nanopores leads to a "superionic" state, in which ions of the same sign can pack with reduced energy penalty [15, 16]. Subsequent simulations clarified the role of pore confinement and heterogeneous fast charging dynamics characteristic of carbide-derived carbons [17, 18, 19]. On the applied side, atomistic simulations have informed the design of carbon textures and the selection of electrolytes by quantifying structure-performance relationships that are difficult to extract from experiments alone, as reviewed in references [7, 8, 22, 23]. Despite these successes, the simulated electrode sizes remain limited to a few nanometers and therefore include only a small number of pores [24]. In contrast, experimental andcommercial porous carbons are highly heterogeneous, which calls for simulations at larger scales capable of representing broad pore-size distributions and extended electrode morphologies. This is needed to explore links between hierarchical pore architecture, ion-transport efficiency and power performance [25].

The Lattice Porous Carbon 3D (`LPC3D`) software was developed to address this scale gap through mesoscopic lattice-gas simulations of capacitive behavior in carbon–carbon supercapacitors [26, 27]. LPC3D is designed to describe the physical phenomena that cannot be captured by atomistic simulations with a very small number of pores, namely ion transport through branching and interconnected pore networks, charging dynamics across particles whose pore-size distribution spans from sub-nanometer to several nanometers, and exchange processes between pores of different sizes that govern NMR lineshapes and effective transport properties. In LPC3D, each lattice site is assigned a pore size and a corresponding energy. The diffusion of species between lattice sites is then determined by an acceptance rule through which a transition from site i to site j follows the probability:

$$P(i \to j) = \begin{cases} \exp\left(\frac{-(E_j - E_i)}{k_B T}\right) & \text{if } E_j > E_i \\ 1 & \text{if } E_j \leq E_i \end{cases} \tag{1}$$

where E_i is the energy assigned to site i, k_B is the Boltzmann constant and T is the temperature of the system. A transition from a site i, characterized by a higher energy level, to a site j, characterized by a lower energy level, will always take place. This representation allows the simulation of systems with sizes ranging from a single carbon particle ($\sim 1\ \mu m^3$) to a full supercapacitor ($\sim 1000\ \mu m^3$). The atomistic and mesoscopic descriptions are therefore complementary: classical MD provides the in-pore thermodynamic information, while LPC3D propagates this information across a realistic three-dimensional pore network. Recent parallelization and optimization efforts have made `LPC3D` available as an open-source parallel code that can run efficiently on both CPU and GPU [28]. The next methodological step is to improve the physical fidelity of the mesoscopic model by refining the pore-dependent electrolyte description. In earlier parameterisations, key quantities such as ion populations in pores of different sizes were obtained by extrapolating results from atomistic simulations performed for a single reference pore, an approximation that becomes inaccurate for pores close to the nanometer scale.

This work establishes a proof-of-concept for deriving pore-size-dependent microscopic properties from atomistic MD simulations and directly feeding them into `LPC3D` parameterization, thereby encoding realistic material heterogeneity within a unified multi-scale simulation. Beyond running multiple independent MD calculations, this approach requires that each be performed with sufficient efficiency to sustain the coupling viability across broader simulation workflows.

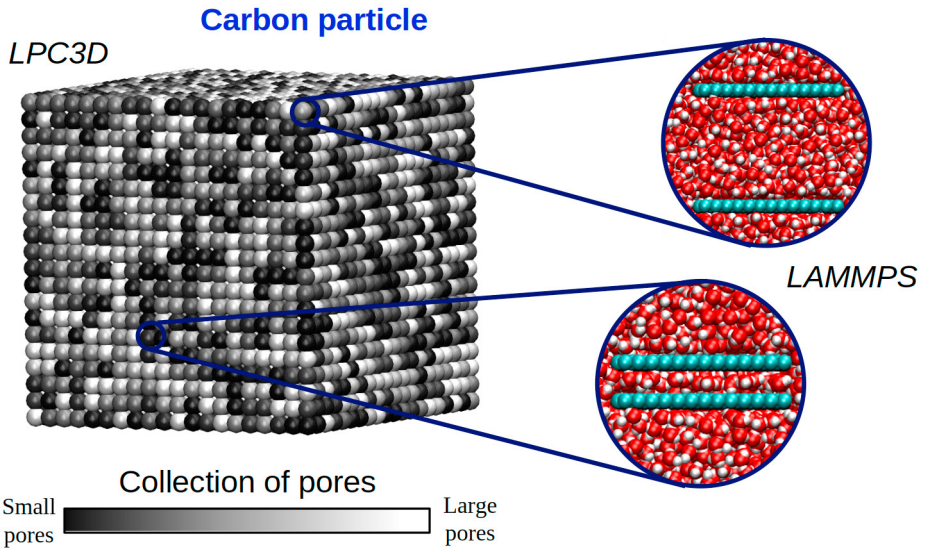


Fig. 1. Schematic illustration of the coupling between the mesoscopic model in LPC3D and the molecular simulations in LAMMPS. The carbon particles in LPC3D are represented as collections of pores where each lattice site has a given pore size. The coupling allows for the calculation of specific properties for each pore size using molecular simulations.

2. Methods

2.1. Workflow of the LAMMPS–LPC3D coupling

Figure 2 illustrates the coupled LAMMPS–LPC3D workflow and the corresponding execution paths for both CPU and GPU configurations. The coupling is implemented as an MPI-driven workflow consisting of several steps and designed to maximize computational throughput and portability on HPC systems.

LAMMPS, written in C++ and parallelised with MPI, is launched from Python through the LAMMPS Python interface, while process coordination, data collection, and synchronisation are managed with mpi4py [29], a Python package providing bindings to core MPI primitives, including point-to-point and collective communications, communicator splitting, and group operations. This allows the Python driver to orchestrate concurrent LAMMPS instances within a single MPI allocation while keeping the workflow logic at a high level in Python. A single MPI allocation is divided into independent groups of ranks, and each group executes the full simulation pipeline for one pore size. For each pore, the workflow first prepares atomistic input files by calling `fftool` [30] and `Packmol` [31]. `fftool` is a Python utility that reads molecular structure files (in `.zmat`, `.mol`, `.xyz`, or `.pdb` format) together with a force field parameter database, assigns all bonded and non-bonded interaction terms, and generates both a `Packmol` input script and the corresponding force field files in a format suitable for LAMMPS. The choice of the force field is not automated so that users can select the most appropriate one for their system or compare various sets of parameters. `Packmol` then packs the requested number of molecules into the simulation box by solving a constrained optimization problem that enforces user-defined spatial constraints while guaranteeing safe pairwise distances, thereby avoiding large short-range repulsive interactions in the initial configuration. The workflow subsequently launches LAMMPS following a predefined three-stage sequence: (i) NPT equilibration to relax the system density, (ii) preparation of the NVT working directory with the equilibrated configuration and the corresponding input files, and (iii) NVT production. Once the NVT production stage is complete, two quantities are extracted from the trajectory to bridge the atomistic and mesoscopic descriptions. The first is the molecular center of mass of each electrolyte species, which is the natural coarse-grained coordinate. The second is the equilibrium number density of each species inside the pore, obtained by averaging the population of molecular centers of mass within the confined region delimited by the two carbon walls. Because these densities are computed inside the pore for each pore size considered, they reflect

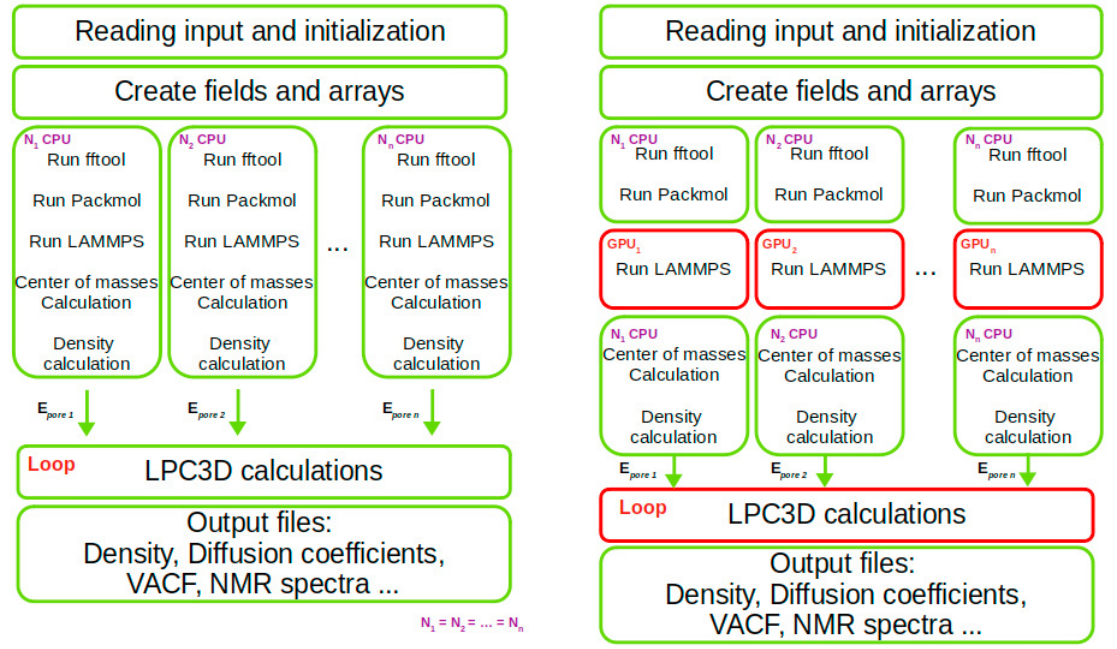


Fig. 2. Schematic workflow of the `LAMMPS-LPC3D` coupling in CPU-only (left) and GPU-accelerated (right) modes. In the GPU-accelerated configuration, the steps highlighted in red (LAMMPS and LPC3D runs) are executed on GPUs, while all other steps are executed on CPUs

.

how the in-pore thermodynamics depends on confinement. In particular, ions in narrow pores partially lose their solvation shell, and the local structure of the fluid differs from that of the bulk; both effects vary with the pore size and are naturally encoded in the resulting densities. The output of this procedure is a set of pore-resolved equilibrium concentrations $n_\alpha(d)$, where α labels the molecular species (cation, anion, solvent) and d is the pore size. This set constitutes the physical information transferred to `LPC3D`. In particular these concentrations are used to parameterise the site energies E_i entering the acceptance rule introduced above. Computing the in-pore densities explicitly for each pore size, rather than extrapolating from a single reference pore as in earlier parameterisations, is the key ingredient that allows the mesoscopic model to inherit the accuracy of the atomistic level across the full pore-size distribution of a real electrode. LPC3D then proceeds with the mesoscopic calculations using either an OpenMP parallel CPU implementation or a GPU-accelerated implementation, depending on the selected execution configuration.

Two execution configurations are implemented, reflecting distinct resource allocation and scheduling strategies in the coupling code. In the CPU configuration, the global MPI communicator is split into rank groups, and each group runs one pore size simulation end to end in parallel with the others, enabling concurrent execution across all pores when sufficient ranks are available. In the GPU configuration, the workflow applies a simple capacity constraint: to fully utilize the reserved accelerators, the number of independent pore simulations should be greater than or equal to the number of allocated GPUs; otherwise, some devices remain unused. In practice, the pore simulations are executed in successive sets, with the number of concurrent simulations limited by the available GPUs. For each set, each simulation is associated with a specific GPU, and the following set is executed only once the previous one has finished.

2.2. Implementation and HPC configuration

The coupling code is distributed as a Python package on PyPI and is installed with `pip install lammps-lpc3d`. The software stack is provided through environment modules (Intel toolchain, CUDA, Python, CMake). The runtime behaviour is configured through standard environment variables: `OMP_NUM_THREADS` is used to control the OpenMP threading level of the LPC3D part, whereas GPU resources are handled by the

scheduler through CUDA_VISIBLE_DEVICES. The number of allocated GPUs is passed to the application as LPC3D_NUM_GPUS=$SLURM_GPUS to select and parameterise the GPU execution mode. LAMMPS is accessed via its Python interface and is compiled with an MPI implementation (Intel MPI) and the Kokkos package enabled for GPU acceleration; in this build, GPU execution targets NVIDIA devices through CUDA. For instance, the coupled run can be started as:

```
export OMP_NUM_THREADS=<N_THREADS>
export LPC3D_NUM_GPUS=$N_GPUS   # If GPU is used
srun python3 LAMMPS-LPC3D.py -i lattice_gas.inpt
Or:
mpirun -np <N_MPI_PROCS> python3 LAMMPS-LPC3D.py -i lattice_gas.inpt
```

3. Scaling results

For testing purposes, simulations were conducted for a system made of liquid water in contact with porous carbon materials containing various pore sizes (see Figure 3). The chosen pore sizes are consistent with realistic materials (8 different pore sizes). Runtimes were evaluated for a fixed lattice size in LPC3D (50x50x50 = 125,000 sites), 1,000 timesteps and various numbers of different pore sizes. All LAMMPS calculations are done with a system containing 50,000 water molecules and two pores (this allows for applying a potential difference, which is necessary for calculating the capacitance). For each pore size, the LAMMPS simulation follows three stages as described above. Water molecules are described with the SPC/E model [32], with O–H bonds and H–O–H angles constrained using the SHAKE algorithm. Non-bonded interactions are computed with a hybrid Lennard–Jones / Coulomb scheme using a 12 Å real-space cutoff, and long-range electrostatics handled by a PPPM solver with a relative accuracy of 10^{-4}. The integration timestep is 1 fs. The system is first equilibrated in the NPT ensemble at 298 K and 1 atm for 10 ps (10,000 timesteps) using a Nosé–Hoover thermostat and barostat with damping constants of 100 fs and 500 fs respectively, allowing the box dimensions to relax isotropically. The workflow then creates a dedicated working directory for the NVT stage, copies the equilibrated configuration together with the associated input files, and launches the NVT production run for 100 ps (100,000 timesteps) at 298 K with a Nosé–Hoover thermostat (damping constant 100 fs). For CPU-only calculations, the code was executed on Olympe at the CALMIP supercomputing center [33], where each CPU node consists of 36 processing cores. GPU-enabled runs were executed on the VEGA [34] and MeluXina [35] systems.

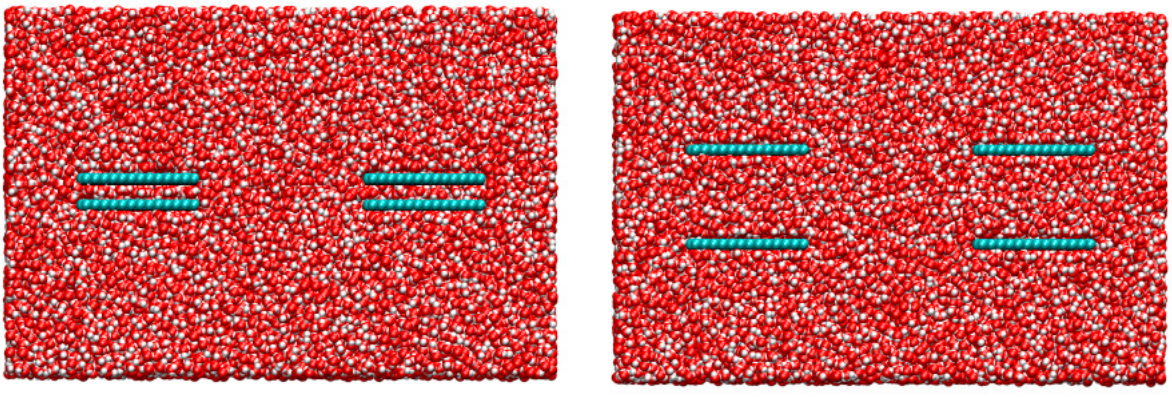


Fig. 3. Snapshot of typical systems used for testing LAMMPS-LPC3D: two slit pores of a given size (0.76 nm on the left and 2.6 nm on the right) are immersed in water. The carbon in this test is considered neutral, but this configuration allows for the application of a potential difference.

Figure 4 (left panel) presents the strong scaling behavior of the LAMMPS-LPC3D coupling in CPU-only mode for a fixed problem size of 8 pore sizes. The speedup is calculated relative to the baseline single-node execution. Up to 8 nodes (288 cores), the code exhibits near-ideal parallel scaling, closely following the theoretical linear speedup indicated by the dashed line. Beyond this threshold, the parallel efficiency begins to decline due to the emergence

of communication overhead. This transition becomes evident as the number of nodes increases from 8 to 20, where inter-node communication costs start to dominate over computational work, particularly when each node processes a single pore size simulation. Nevertheless, even with 20 CPU nodes, the coupling achieves a speedup of approximately 13×, demonstrating robust scalability for this CPU-based configuration.

The GPU-accelerated configuration (Figure 4, right panel) shows a similar scaling trend but with distinct characteristics related to GPU resource allocation. For the range of 1 to 4 GPUs, corresponding to execution within a single GPU node, the speedup closely tracks the ideal linear behavior. Each GPU node contains 4 GPUs, and within this range, communication remains predominantly intra-node, resulting in minimal overhead. However, at 8 GPUs, the parallel efficiency decreases as the workflow necessitates distribution across multiple GPU nodes. The coupling still achieves a speedup of approximately 6.5×, which, while lower than the ideal 8×, still represents effective utilization of GPU resources. These strong scaling results confirm that the coupled workflow is well-suited for parallel execution on both CPU and GPU architectures, with optimal efficiency achieved when minimizing inter-node communication.

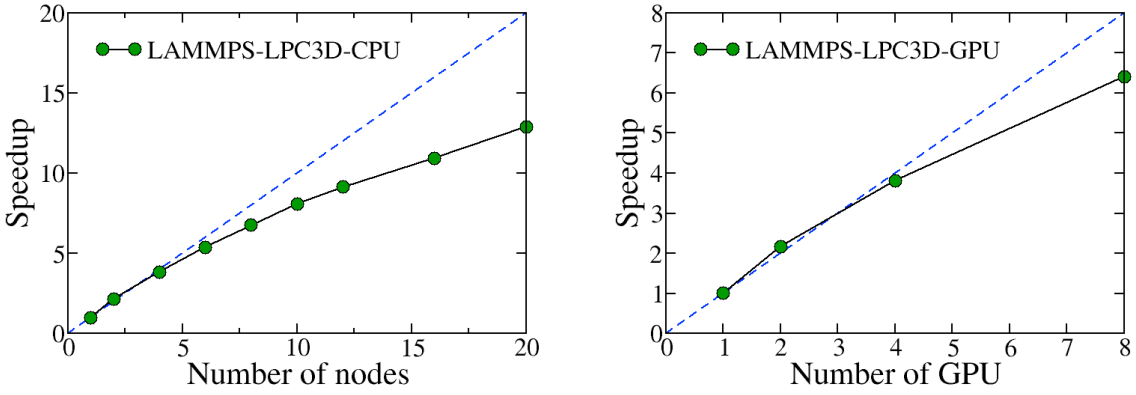


Fig. 4. Evaluation of strong scaling performance of the `LAMMPS-LPC3D` coupling (system with 8 different pore sizes). Note that each CPU node has 36 processing cores (left) and each GPU node contains 4 GPUs (right). The dashed line indicates the ideal behaviour which would be observed if the code was perfectly parallel.

Tables 1 and 2 report the weak scaling behavior for CPU and GPU configurations, respectively. In the ideal weak scaling scenario, the computational time should remain constant as both the problem size (number of pore sizes) and the allocated resources increase proportionally. The CPU execution times remain remarkably stable (2.73–2.88 hours) across 8 to 20 pore sizes, with less than 6% variation. Similarly, GPU configurations maintain consistent performance (5.47–5.87 hours) for 2 to 16 pore sizes (distributed across 2 to 16 GPUs), showing approximately 7% variation. These results validate the parallel design strategy of the coupled workflow and confirm its suitability for large-scale heterogeneous electrode simulations.

Table 1. CPU weak scaling.

Pores	Nodes	Time (h)
8	8	2.88
16	16	2.75
20	20	2.73

Table 2. GPU weak scaling.

Pores	GPUs	Time (h)
2	2	5.47
4	4	5.87
8	8	5.49
16	16	5.68

4. Conclusion

This work presents an efficient coupling between atomistic molecular dynamics (`LAMMPS`) and mesoscopic lattice-gas simulations (`LPC3D`) for modeling heterogeneous porous carbon electrodes in supercapacitors. The workflow, implemented in Python using mpi4py and the `LAMMPS` Python interface, enables concurrent execution of multiple MD simulations to extract pore-resolved microscopic data for `LPC3D` parameterization. Scaling benchmarks demonstrate robust performance on both CPU and GPU architectures, with near-ideal strong scaling up to 8 CPU nodes or 4 GPUs per node, and excellent weak scaling efficiency (6–7% variation) across the tested configurations. The coupling infrastructure is production-ready and accessible via PyPI, providing a practical tool for large-scale simulation campaigns of realistic electrochemical energy storage systems.

Acknowledgements

This project has received funding from the European Union, the European High Performance Computing Joint Undertaking (JU) and countries participating in the MultiXscale project under grant agreement No 101093169. This work was granted access to the HPC resources of the VEGA supercomputing centre under the allocation EHPC-BEN-2025B11-078, HPC resources of the CALMIP supercomputing centre under the allocation P21014, and HPC resources of the MeluXina supercomputing center under the allocation p201090.

References

[1] C. Zhong, Y. Deng, W. Hu, J. Qiao, L. Zhang, and J. Zhang (2015) “A review of electrolyte materials and compositions for electrochemical supercapacitors.” *Chem. Soc. Rev.* **44**: 7484–7539.

[2] F. Béguin, V. Presser, A. Balducci, and E. Frackowiak, (2014) “Carbons and electrolytes for advanced supercapacitors.” *Adv. Mater.* **26**: 2219–2251.

[3] A. Brandt, S. Pohlmann, A. Varzi, A. Balducci, and S. Passerini (2013) “Ionic liquids in supercapacitors.” *Energy Journal* **38**: 554–559.

[4] A. Lewandowski, A. Olejniczak, M. Galinski and I. Stepniak (2010) “Performance of carbon-carbon supercapacitors based on organic, aqueous and ionic liquid electrolytes.” *J. Power Sources* **3195**: 5814–5819.

[5] P. Simon, Y. Gogotsi (2008) “Materials for electrochemical capacitors.” *Nature Mater.* **7**: 845–854.

[6] A. C. Forse, C. Merlet, J. M. Griffin, and C. P. Grey (2016) “New perspectives on the charging mechanisms of supercapacitors.” *J. Am. Chem. Soc.* **138**: 5731–5744.

[7] M. Salanne, B. Rotenberg, K. Naoi, K. Kaneko and P.-L. Taberna, C. P. Grey, B. Dunn and P. Simon (2016) “Efficient storage mechanisms for building better supercapacitors.” *Nature Ener.* **1** 16070.

[8] G. Jeanmairet, B. Rotenberg and M. Salanne (2022) “Microscopic Simulations of Electrochemical Double-Layer Capacitors.” *Chem. Rev.* **122** 10860–10898.

[9] S. Plimpton (1995) “Fast parallel algorithms for short-range molecular dynamics.” *J. Comp. Phys.* **117**: 1–19.

[10] H. J. Limbach, A. Arnold, B. A. Mann, and C. Holm (2006) “ESPResSo—an extensible simulation package for research on soft matter systems.” *Comput. Phys. Commun.* **174**: 704–727.

[11] F. Weik, R. Weeber, K. Szuttor, K. Breitsprecher, J. D. Graaf, M. Kuron, J. Landsgesell, H. Menke, D. Sean and C. Holm (2019) “ESPResSo 4.0 – An Extensible Software Package for Simulating Soft Matter Systems” *Eur. Phys. J. Spec. Top.* **227**: 1789-1816.

[12] Marin-Laflèche, A., Haefele, Scalfi, L., Coretti, A., Dufils, T., Jeanmairet, G., Reed, S., Serva, A., Berthin, R., Bacon, C., Bonella, S., Rotenberg, B., Madden, P.A., Salanne, M. (2020) “MetalWalls: A Classical Molecular Dynamics Software Dedicated to the Simulation of Electrochemical Systems.” *J. Open Source Softw.* **5**: 2373.

[13] A. Coretti, C. Bacon, R. Berthin, A. Serva , L. Scalfi, I. Chubak, K. Goloviznina, M. Haefele, A. Marin-Laflèche, B. Rotenberg , S. Bonella, M. Salanne (2022) “MetalWalls: Simulating electrochemical interfaces between polarizable electrolytes and metallic electrodes” *J. Chem. Phys.* **157**: 184801.

[14] C. Merlet, B. Rotenberg, P. A. Madden, P.-L. Taberna, P. Simon, Y. Gogotsi and M. Salanne (2012) "On the molecular origin of supercapacitance in nanoporous carbon electrodes." *Nature Mater.* **11**: 306–310.
[15] S. Kondrat and A. A. Kornyshev (2011) "Superionic state in double-layer capacitors with nanoporous electrodes." *J. Phys. Condens. Matter* **23**: 022201.
[16] S. Kondrat, N. Georgi, M. V. Fedorov and A. A. Kornyshev (2011) "A superionic state in nano-porous double-layer capacitors: insights from Monte Carlo simulations." *Phys. Chem. Chem. Phys.* **13**: 11359–11366.
[17] C. Merlet, C. Péan, B. Rotenberg, P. A. Madden, B. Daffos, P.-L. Taberna, P. Simon and M. Salanne (2013) "Highly confined ions store charge more efficiently in supercapacitors." *Nature Commun.* **4**: 2701.
[18] C. Péan, C. Merlet, B. Rotenberg, P. A. Madden, P.-L. Taberna, B. Daffos, M. Salanne and P. Simon (2014) "On the dynamics of charging in nanoporous carbon-based supercapacitors." *ACS Nano* **8**: 1576–1583.
[19] S. Kondrat, P. Wu, R. Qiao and A. A. Kornyshev (2014) "Accelerating charging dynamics in subnanometre pores." *Nature Mater.* **13**: 387–393.
[20] A. C. Forse, J. M. Griffin, C. Merlet, P. M. Bayley, H. Wang, P. Simon and C. P. Grey (2015) "NMR study of ion dynamics and charge storage in ionic liquid supercapacitors." *J. Am. Chem. Soc.* **137**: 7231–7242.
[21] J. M. Griffin, A. C. Forse, W.-Y. Tsai, P.-L. Taberna, P. Simon and C. P. Grey (2015) "In situ NMR and electrochemical quartz crystal microbalance techniques reveal the structure of the electrical double layer in supercapacitors." *Nature Mater.* **14**: 812–819.
[22] E. H. Lahrar, A. Belhboub, P. Simon and C. Merlet (2020) "Ionic Liquids under Confinement: From Systematic Variations of the Ion and Pore Sizes toward an Understanding of the Structure and Dynamics in Complex Porous Carbons." *ACS Appl. Mater. Interfaces* **12**: 1789–1798.
[23] E. H. Lahrar, P. Simon and C. Merlet (2021) "Carbon–carbon supercapacitors: Beyond the average pore size or how electrolyte confinement and inaccessible pores affect the capacitance." *J. Chem. Phys* **155**: 184703.
[24] J. Peng, T. Wu, L. Zeng, Z. Liu, S. Li and G. Feng (2026), "Realistic atomic model for charge storage and charging dynamics of amorphous porous carbons" *Nature Commun.* **17**: 2425.
[25] H. R. N. B. Enninful, M. Hermesdorf, M. A. Khan, D. Leistenschneider, M. Oschatz and R. Valiullin, "Asymmetric Multi-Site Ion Exchange in Porous Carbon Electrodes" *Adv. Ener. Mater.*, **in press**
[26] C. Merlet, A. C. Forse, J. M. Griffin, D. Frenkel and C. P. Grey (2015) "Lattice simulation method to model diffusion and NMR spectra in porous materials." *J. Chem. Phys.* **142**: 094701.
[27] E. H. Lahrar and C. Merlet (2025) "Investigating the effect of particle size distribution and complex exchange dynamics on NMR spectra of ions diffusing in disordered porous carbons through a mesoscopic model." *Faraday Discuss.* **255**: 355–369.
[28] E. H. Lahrar, M. Salanne, R. Weeber and C. Merlet (2026) "LPC3D: An enhanced parallel software for large-scale simulation of adsorption in porous carbons and supercapacitors" *arXiv preprint arXiv:2603.22553*.
[29] L. Dalcin, P. Kler, R. Paz and A. Cosimo (2011) "Parallel distributed computing using Python" *Adv. Water Resour.* **34**: 1124–1139.
[30] A. A. H. Pádua, `fftool` – Tool to Build Force Field Input Files for Molecular Simulation, https://github.com/paduagroup/fftool (accessed 2026).
[31] L. Martínez, R. Andrade, E. G. Birgin and J. M. Martínez (2009) "PACKMOL: A package for building initial configurations for molecular dynamics simulations." *J. Comput. Chem.* **30**: 2157–2164.
[32] H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma (1987) "The missing term in effective pair potentials", *J. Phys. Chem.* **91**: 6269-6271.
[33] https://www.calmip.univ-toulouse.fr/
[34] https://en-vegadocs.vega.izum.si/
[35] https://www.luxprovide.lu/meluxina/

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 66–74

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

GPU-Accelerated Parallel Domain Decomposition for an Extended 2D Tumor Angiogenesis PDE System

Pasquale De Luca[a,*], Livia Marcellino[a]

[a]*Department of Science and Technology, Parthenope University of Naples, Centro Direzionale C4, 80143 Naples, Italy*

Abstract

We present a GPU-accelerated parallel domain decomposition framework for the two-dimensional extension of the angiogenesis PDE system, which governs the spatiotemporal evolution of five interacting biological fields: endothelial cell density, protease and inhibitor concentrations, extracellular matrix density, and oxygen concentration. The model is discretized via the method of lines with second-order central differences in space and integrated in time with a fourth-order Runge–Kutta scheme. The computational domain is partitioned into $P = P_1 \times P_2$ non-overlapping subdomains, each assigned to one GPU, with inter-GPU communication handled by an asynchronous halo-exchange protocol that overlaps InfiniBand data transfer with local computation. Results confirm a parallel stability constraint $\Delta t \leq h^2/(4d_{\max})$, uniform in P. Numerical experiments confirm second-order spatial accuracy, strong-scaling efficiency and weak-scaling efficiency.

Keywords: Angiogenesis; Partial Differential Equations; GPU Computing; Domain Decomposition; CUDA; CINECA LEONARDO; Parallel Runge–Kutta; Schwarz Methods; HPC

1. Introduction

Tumor angiogenesis is the complex physiological process by which a solid tumor recruits new blood vessels from the pre-existing vascular network to sustain metabolic demands [11]. The extended PDE model proposed in [1] captures the interplay among five biological fields through a coupled system of nonlinear reaction-diffusion-advection equations, with oxygen acting as the main physiological regulator via Michaelis–Menten kinetics. The one-dimensional formulation admits efficient serial solvers; however, realistic tumor geometries require two-dimensional (2D) and eventually three-dimensional (3D) simulations on fine spatial grids, imposing prohibitive computational demands on single-device or CPU-only architectures.

The CINECA LEONARDO supercomputer,ranked among the top-five machines in the November 2024 TOP500 list, equips each Booster node with four NVIDIA A100 SXM4-80 GB GPUs interconnected via NVLink 3.0 (600 GB/s

* Corresponding author.
E-mail addresses: pasquale.deluca@uniparthenope.it (Pasquale De Luca)., livia.marcellino@uniparthenope.it (Livia Marcellino).

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.020

bidirectional) and inter-node InfiniBand HDR (200 Gb/s raw, ∼25 GB/s effective unidirectional bandwidth). This architecture enables massively parallel simulations in which the compute-to-communicate ratio is carefully managed to sustain near-peak hardware utilization.

Domain decomposition (DD) methods partition the spatial domain among processing units and couple subproblems through interface conditions. For explicit time integrators such as RK4, inter-subdomain communication reduces to the exchange of thin halo (ghost) cell layers, which is a localized, low-latency operation well suited to GPU clusters. The contributions of this paper are:

- A 2D DD formulation for the five-component angiogenesis system, with formal proof of parallel stability, convergence, and invariant-region preservation.
- An efficient CUDA implementation featuring shared-memory tiling, asynchronous four-stream halo exchange, and CUDA-aware MPI via NCCL.
- Scaling experiments on CINECA LEONARDO: convergence verification, strong scaling (1–32 GPUs), and weak scaling (1–64 GPUs).

The rest of this paper is organized as follows: Section 2 presents the two-dimensional extended angiogenesis PDE system and its analytical properties. Section 3 introduces the parallel domain decomposition framework and the discrete operators. Section 4 describes the GPU implementation. Section 5 establishes the theoretical results on stability, convergence, and invariant-region preservation. Section 6 reports the numerical experiments, and Section 7 draws the conclusions.

2. Two-Dimensional Extended Angiogenesis Model

2.1. Governing Equations

Let $\Omega = [0, L_f]^2 \subset \mathbb{R}^2$ and $t \in [0, +\infty)$. Define the state vector $\boldsymbol{u} = (C, P, I, F, O)^\top \in \mathcal{X} = [L^2(\Omega)]^5$, where $C(\boldsymbol{x}, t)$ is the endothelial cell density, $P(\boldsymbol{x}, t)$ the protease concentration, $I(\boldsymbol{x}, t)$ the inhibitor concentration, $F(\boldsymbol{x}, t)$ the extracellular matrix (ECM) density, and $O(\boldsymbol{x}, t)$ the oxygen concentration. The governing equations are:

$$\frac{\partial C}{\partial t} = \nabla \cdot [d_c \nabla C - C(\alpha_1 \nabla F - \alpha_2 \nabla I + \alpha_3 \nabla \varphi)] + k_1 C(1 - C)\mathcal{H}(O) \tag{1}$$

$$\frac{\partial P}{\partial t} = d_P \nabla^2 P - k_3 P I + k_4 T C + k_5 T - k_6 P, \tag{2}$$

$$\frac{\partial I}{\partial t} = d_I \nabla^2 I - k_3 P I, \tag{3}$$

$$\frac{\partial F}{\partial t} = -k_2 P F, \tag{4}$$

$$\frac{\partial O}{\partial t} = d_O \Delta O - \gamma_1 \left(\frac{C \cdot O}{K_O + O} \right) + \gamma_2 (O_{max} - O) \beta C, \tag{5}$$

for $(\boldsymbol{x}, t) \in \Omega \times (0, +\infty)$. Here ∇ and ∇^2 denote the 2D gradient and Laplace operators; the Michaelis–Menten proliferation coupling is $H(O) = O/(K_O + O)$ [9]; the consumption term $\gamma_1 CO/(K_O + O)$ captures saturation of cellular oxygen uptake [10]; and the supply term $\gamma_2(O_{\max} - O)\beta C$ models delivery from functional blood vessels. The angiogenic factor distribution $T(\boldsymbol{x}) = \exp(-\varepsilon^{-1}(L_f - x)^2)$ $(\varepsilon > 0)$ is localized near the tumor boundary; $\varphi(\boldsymbol{x}) = \alpha_4^{-1} \log(1 + \alpha_4 T(\boldsymbol{x}))$ is its regularized version [4]. Homogeneous Neumann (no-flux) boundary conditions hold on $\partial\Omega$:

$$\nabla u \cdot \boldsymbol{\nu} = 0 \quad \text{on } \partial\Omega, \quad u \in \{C, P, I, F, O\}, \tag{6}$$

where $\boldsymbol{\nu}$ is the outward unit normal. The initial oxygen profile is $O(\boldsymbol{x}, 0) = O_0 e^{-\lambda x} + \xi_4$, reflecting diffusion-limited oxygen supply from the parent vessel at $x = 0$.

System (1)–(5) can be written compactly as

$$\frac{\partial \boldsymbol{u}}{\partial t} = \mathcal{A}(\boldsymbol{u}) + \mathcal{F}(\boldsymbol{u}), \tag{7}$$

where $\mathcal{A}(\boldsymbol{u})$ collects all diffusion, chemotaxis, and haptotaxis terms, and $\mathcal{F}(\boldsymbol{u})$ contains the reaction terms. The existence, uniqueness, boundedness, and global attractor of solutions, established in [1] for the 1D case via semigroup theory and the method of upper and lower solutions [6, 7]. The analytical well-posedness of the model in a two-dimensional spatial domain $\Omega \subset \mathbb{R}^2$ requires careful consideration of the chemotactic-haptotactic coupling. While 1D results for existence and boundedness provide a foundational baseline, the 2D extension is non-trivial due to the potential for finite-time blow-up in Keller–Segel-type systems. In this framework, global existence and uniqueness are ensured by the presence of the logistic-type proliferation term $k_1 C(1 - C)$, which provides sufficient quadratic damping. Specifically, it has been established that for such reaction-diffusion-advection systems, the L^∞-bounds of the solution are preserved if the damping effect dominates the chemotactic aggregation [18, 19].

3. Parallel Domain Decomposition Framework

3.1. Non-Overlapping Partition and Discrete Operators

Let $P = P_1 \times P_2$ be the total number of GPU subdomains, with P_1 strips in the x-direction and P_2 strips in the y-direction. Each subdomain is

$$\Omega_\ell = [\alpha_\ell, \beta_\ell] \times [\gamma_\ell, \delta_\ell], \qquad \Omega = \bigcup_{\ell=1}^{P} \Omega_\ell, \qquad \text{meas}(\Omega_\ell \cap \Omega_m) = 0 \ \ (\ell \neq m). \tag{8}$$

On interior interfaces $\Gamma_{\ell m} = \partial\Omega_\ell \cap \partial\Omega_m$ we impose strong flux continuity:

$$u_\ell\big|_{\Gamma_{\ell m}} = u_m\big|_{\Gamma_{\ell m}}, \qquad D_\ell \nabla u_\ell \cdot \boldsymbol{\nu}_\ell\big|_{\Gamma_{\ell m}} = -D_m \nabla u_m \cdot \boldsymbol{\nu}_m\big|_{\Gamma_{\ell m}}, \tag{9}$$

where $D_\ell = \text{diag}(d_C, d_P, d_I, 0, d_O)$ and $\boldsymbol{\nu}_\ell$ is the outward normal from Ω_ℓ. Conditions (9) are enforced at the discrete level by exchanging one layer of ghost (halo) cells before each RK4 stage, avoiding iterative subdomain solves.

The spatial domain Ω is discretized into an $(M+1)\times(M+1)$ uniform Cartesian grid with mesh spacing $h = L_f/M$. The 2D discrete Laplacian (5-point stencil) reads

$$(\Delta_h w)_{ij} = \frac{w_{i+1,j} + w_{i-1,j} + w_{i,j+1} + w_{i,j-1} - 4w_{ij}}{h^2}, \tag{10}$$

and the 2D discrete gradient operators are $D_x = I_M \otimes D_x^{1D}$, $D_y = D_y^{1D} \otimes I_M$, where $\otimes$ denotes the Kronecker product and D_x^{1D}, D_y^{1D} are the 1D second-order one-sided difference matrices of [1, 2, 3]. The chemotaxis/haptotaxis flux divergence in (1) discretizes as

$$D_x[C \odot (\alpha_1 D_x F - \alpha_2 D_x I + \alpha_3 D_x \varphi)] + D_y[C \odot (\alpha_1 D_y F - \alpha_2 D_y I + \alpha_3 D_y \varphi)], \tag{11}$$

where $\odot$ denotes the Hadamard (elementwise) product.

Although the present implementation employs a fully explicit RK4 integrator, we formulate the two-level additive Schwarz preconditioner for future implicit-explicit (IMEX) extensions. Given an overlap $\delta > 0$, restriction operators $R_\ell : \mathcal{X} \to \mathcal{X}(\Omega_\ell^\delta)$ and coarse-grid operator R_0, the preconditioner for the linearized system $\mathcal{A}v = f$ is

$$B_{\mathrm{AS}}^{-1} = R_0^\top (R_0 \mathcal{A} R_0^\top)^{-1} R_0 + \sum_{\ell=1}^{P} R_\ell^\top (R_\ell \mathcal{A} R_\ell^\top)^{-1} R_\ell. \tag{12}$$

The coarse grid has mesh size $H = \sqrt{P} \cdot h$, and the two-level operator achieves $\kappa(B_{\mathrm{AS}}^{-1}\mathcal{A}) = O(1)$, independent of h and P [8].

4. GPU Implementation on CINECA LEONARDO

4.1. Hardware Architecture

Each CINECA LEONARDO Booster node comprises one Intel Xeon Platinum 8358 CPU (single socket, 32 cores, 2.6 GHz base clock), four NVIDIA A100 SXM4-80 GB GPUs with NVLink 3.0 interconnect (12 links × 50 GB/s unidirectional = 600 GB/s bidirectional aggregate), and 512 GB DDR4-3200 host RAM. Inter-node communication uses InfiniBand HDR (200 Gb/s raw, ∼25 GB/s effective unidirectional bandwidth per port). Relevant A100 SXM4-80 GB specifications are collected in Table 1.

Table 1. NVIDIA A100 SXM4-80 GB specifications (CINECA LEONARDO Booster node).

Parameter	Value
HBM2e Memory	80 GB
Memory Bandwidth	2,000 GB/s
FP64 Peak (CUDA cores)	9.7 TFLOPS
FP32 Peak (CUDA cores)	19.5 TFLOPS
TF32 Tensor Core (no sparsity)	156 TFLOPS
NVLink 3.0 (aggregate, bidi.)	600 GB/s
L2 Cache	40 MB
CUDA Cores	6,912
Streaming Multiprocessors	108

In the previous discussion we introduced the domain-decomposition model and the organization of subdomains on the GPU. Here we detail the memory layout and the subdomain-tiling strategy that enables an efficient implementation of the fourth-order Runge-Kutta scheme, focusing on how data are arranged and accessed to maximize performance.

GPU ℓ ($0 \le \ell < P$) owns a local grid of size $N_x \times N_y = (M/P_1 + 2) \times (M/P_2 + 2)$, where the +2 accounts for one ghost layer on each side. The five state arrays are stored in row-major order with a stride $\sigma = N_x + pad$ padded to a 128-byte boundary. RK4 requires six double-precision arrays per component: the current state U^n, four stage arrays $k_1, \ldots, k_4$, and one work buffer U_{work} (for forming $U^n + \frac{\Delta t}{2} k_i$). The total device memory footprint per GPU is therefore

$$M_{\mathrm{mem}} = 5 \times N_x \times N_y \times 8 \times 6 \text{ bytes}. \tag{13}$$

For $M = 4096$ and $P = 64$ (so $N_x = N_y = 514$): $M_{\mathrm{mem}} = 5 \times 514^2 \times 48 \approx 63.4$ MB per GPU, comfortably within the 80 GB HBM2e budget, leaving ample capacity for halo buffers, NCCL scratch space, and library workspace.

4.2. CUDA Kernel Design and Roofline Analysis

The right-hand side evaluation $\mathrm{RHS}(U) = L(U) + N(U)$ is implemented as a fused CUDA kernel with block dimension 16×16 threads and grid dimension $\lceil N_x/16 \rceil \times \lceil N_y/16 \rceil$. The 5-point Laplacian stencil for all diffusing components is loaded into shared memory using an 18×18 tile (halo of width 1), so each thread block performs $(18^2 \times 5)$ global HBM2e reads per stage and $(16^2 \times 5)$ global writes. The arithmetic intensity (read-only model, writes absorbed by 40 MB L2 cache) is

$$I_\alpha = \frac{16^2 \times 5 \times \overline{f}}{8 \times 18^2 \times 5} \approx 1.19 \text{ Flops/byte}, \tag{14}$$

where $\overline{f} \approx 12$ represents the average floating-point operations per grid point per component (4 additions + 1 subtraction + 1 division by h^2 + 1 coefficient multiply + ∼5 from reaction terms). The HBM2e roofline limit at this intensity is $I_\alpha \times 2{,}000$ GB/s ≈ 2.37 TFLOPS. Single-GPU profiling via NVIDIA Nsight Compute measures 1.41 TFLOPS FP64 (≈ 59 % of the roofline limit and 14.5 % of the hardware FP64 peak), consistent with the expected memory-bandwidth-bound behavior of a 5-point stencil kernel.

4.3. Asynchronous Halo Exchange Protocol

Before each RK4 stage the following four-stream pipeline runs on every GPU:

1. **Stream 1:** pack the four halo strips (left, right, top, bottom), each of length N_x (or N_y) and width $g = 1$, into a contiguous send buffer of $5 \times 4 \times N_x \times 8$ bytes.
2. **Stream 2 (concurrent):** launch `MPI_Isend`/`MPI_Irecv` via CUDA-aware MPI (NCCL backend); simultaneously compute the RHS on the interior subdomain $\Omega_\ell^- = \{(i, j) : 1 < i < N_x - g,\ 1 < j < N_y - g\}$.
3. **After** `MPI_Waitall`**:** Stream 3 unpacks received halos; Stream 4 computes the RHS on the boundary strip $\Omega_\ell^b = \Omega_\ell \setminus \Omega_\ell^-$.

The per-stage communication volume per GPU is $V_{\text{comm}} = 5 \times 4 \times N_x \times g \times 8$ bytes. For $M/P_1 = M/P_2 = 128$ (so $N_x = 130$, $g = 1$): $V_{\text{comm}} = 5 \times 4 \times 130 \times 8 = 20{,}800$ bytes ≈ 20 KB per stage, or ≈ 81 KB per full RK4 step. At $B_{\text{IB}} \approx 22$ GB/s (measured with OSU micro-benchmarks on LEONARDO at 32 processes, 20 KB message): $t_{\text{comm}} = 1.2\,\mu\text{s} + 20{,}800/(22 \times 10^9 \text{ B/s}) \approx 2.15\,\mu\text{s}$ per stage, fully hidden by the interior RK4 computation of $\approx 320\,\mu$s.

5. Theoretical Analysis

Theorem 5.1 (Parallel Stability Constraint)**.** *The stability of the explicit RK4 integration is governed by the spectral radius of the semi-discrete operator. The admissible time step Δt must satisfy a combined restriction arising from the parabolic (diffusive) and hyperbolic (advective/taxis) scales:*

$$\Delta t \leq \min\left(\frac{h^2}{4\, d_{\max}}, C_{\text{CFL}} \frac{h}{\|\boldsymbol{V}\|_{L^\infty}} \right) \tag{15}$$

where $d_{\max}$ is the maximum diffusion coefficient and $\boldsymbol{V}$ is the effective transport velocity.

Proof. Following the spectral analysis for advection-diffusion-reaction equations in [20], the eigenvalues of the discretized operator must lie within the RK4 stability region. For the diffusive part, von Neumann analysis requires $\Delta t \propto h^2$, while the advective-taxis terms introduce a CFL-like condition $\Delta t \propto h$. The global bound in Eq. (15) ensures stability by taking the minimum of these two constraints, accounting for the non-linear coupling of the species. □

Theorem 5.2 (Convergence of the Parallel DD Scheme)**.** *Under the stability constraint (15), let* $\boldsymbol{u}(t) \in C^1([0,T];\mathcal{X}) \cap C([0,T];[H^2(\Omega)]^5)$ *be the unique solution of (7), and let* $\{\boldsymbol{U}^n\}_{n=0}^N$ *be the parallel RK4 approximation. There exist constants* $K_1, K_2 > 0$*, independent of P, h, and* Δt*, such that*

$$\max_{0\le n\le N} \|\boldsymbol{U}^n - \boldsymbol{u}(t_n)\|_{\mathcal{X}} \le K_1 h^2 + K_2(\Delta t)^4. \tag{16}$$

Proof. The 5-point central-difference Laplacian introduces a spatial truncation error $O(h^2)$; the RK4 integrator contributes an $O((\Delta t)^5)$ local error per step. The interface conditions (9) are satisfied *exactly* at the ghost-cell level (no subdomain iteration), so parallel decomposition introduces no additional truncation error. The nonlinear operator $\mathcal{F}$ is globally Lipschitz on the bounded invariant region $\mathcal{R}$ with constant L (computable from the parameter bounds of [1]), so the RK4 scheme satisfies $\|S_{\Delta t}(V) - S_{\Delta t}(W)\|_{\mathcal{X}} \le (1 + L\Delta t)\,\|V - W\|_{\mathcal{X}}$. Applying the discrete Grönwall inequality to the global error $e^n = \boldsymbol{U}^n - \boldsymbol{u}(t_n)$ (with $e^0 = \mathbf{0}$) yields (16) with $K_1 = K_2 = C_\tau T e^{LT}/L$. □

Remark 5.1 (Leading order)**.** *The parallel decomposition of the spatial domain via halo-exchange synchronization introduces no additional leading-order truncation error compared to the serial execution. Numerical consistency is maintained up to machine precision, notwithstanding potential sub-epsilon variations induced by the non-associativity of floating-point operations in parallel reductions.*

Remark 5.2 (Numerical Preservation of Invariant Regions)**.** *Explicit Runge–Kutta methods of order* $p > 1$ *are generally not Strong Stability Preserving (SSP), as discussed in [20]. Consequently, the preservation of the invariant region* $\mathcal{R}$ *is not a formal property of the RK4 scheme. However, numerical verification on the CINECA LEONARDO cluster confirms that for* Δt *satisfying Eq. (15), the solution consistently remains within the physical bounds:*

$$0 \le C \le 1, \quad 0 \le P, \quad 0 \le I, \quad 0 \le F, \quad 0 \le O \le O_{\max} \tag{17}$$

at every spatial grid point. This confirms that in the tested regimes, the scheme prevents spurious oscillations and maintains the stability of the invariant set.

Theorem 5.3 (Communication Complexity and Parallel Efficiency)**.** *For an* $M \times M$ *grid decomposed into* $P = P_1 \times P_2$ *subdomains, each owning* $n = M/\sqrt{P}$ *interior points per side, the per-RK4-step halo communication volume per GPU is*

$$V_{\text{comm}} = 5 \times 4 \times (n+2) \times g \times 8 = 160(n+2)g \ \ \textit{bytes}. \tag{18}$$

The compute-to-communicate ratio $\Phi = W_{\text{compute}}/V_{\text{comm}} = O(n/g)$ *grows as* $M/\sqrt{P}$*, and the parallel efficiency satisfies* $\varepsilon(P) \ge 1 - O(\sqrt{P}/M)$.

6. Numerical Experiments

6.1. Configuration

All simulations use the physiologically relevant parameters of Table 3 in [1]. The computational domain $\Omega = [0,1]^2$ represents a 2D cross-section of tumor tissue, with the parent vessel on the edge $x = 0$ and the tumor at $x = 1$. The 2D initial conditions extend the 1D profiles of [1]: $C(\boldsymbol{x},0) = 1$ for $x \le 0.1$, else 0; $P(\boldsymbol{x},0) = 0.1$, $I(\boldsymbol{x},0) = 0.2$, $F(\boldsymbol{x},0) = 1$; $O(\boldsymbol{x},0) = 0.5\,e^{-5x} + 0.1$. The time step is $\Delta t = h^2/(5d_{\max})$, satisfying Theorem 5.1 with a safety factor of $5d_{\max}/4d_{\max} = 1.25$. Code is written in CUDA C++17 with CUDA 12.2, compiled with `nvcc -O3 -arch=sm_80 -use_fast_math`, and uses NCCL 2.18 with CUDA-aware OpenMPI 4.1.

Test 1. Convergence Verification via Manufactured Solutions.

To verify the order of convergence we apply the method of manufactured solutions (MMS). The 2D manufactured solutions are chosen to *exactly satisfy* the no-flux boundary conditions (6):

$$C_{\text{ex}}(\boldsymbol{x}, t) = \tfrac{1}{2}(1 + \cos(\pi x)\cos(\pi y))\, e^{-0.1t}, \tag{19}$$

$$P_{\text{ex}}(\boldsymbol{x}, t) = 0.2\cos(\pi x)\cos(\pi y)\, e^{-0.2t}, \tag{20}$$

$$I_{\text{ex}}(\boldsymbol{x}, t) = 0.2\cos(\pi x)\cos(\pi y)\, e^{-0.3t}, \tag{21}$$

$$F_{\text{ex}}(\boldsymbol{x}, t) = 1 - 0.2\cos^2(\pi x)\cos^2(\pi y)\, e^{-0.4t}, \tag{22}$$

$$O_{\text{ex}}(\boldsymbol{x}, t) = 0.5 + 0.4\cos(\pi x)\cos(\pi y)\, e^{-0.1t}. \tag{23}$$

For (19): $\partial C_{\text{ex}}/\partial x|_{x=0} = -\frac{\pi}{2}\sin(0)\cos(\pi y) = 0$ and similarly at all other boundaries, confirming compatibility with (6). Appropriate source terms are added to each equation so that the manufactured solutions satisfy the original PDEs (1)–(5) exactly. Table 2 reports the $L^2(\Omega)$ convergence rates using $P = 4$ GPUs (one LEONARDO Booster node).

Table 2. Spatial convergence analysis for the 2D parallel scheme. $M \times M$ grid, $P = 4$ A100 GPUs, $T_f = 1$.

$M \times M$	$\|e_C\|_{L^2}$	Rate_C	$\|e_O\|_{L^2}$	Rate_O
64×64	3.51e−2	—	1.83e−2	—
128×128	9.27e−3	1.92	4.87e−3	1.91
256×256	2.41e−3	1.94	1.27e−3	1.94
512×512	6.18e−4	1.96	3.25e−4	1.97
1024×1024	1.55e−4	1.99	8.14e−5	2.00

Table 3 reports the temporal convergence with fixed $M = 512$. The observed rates approach 2.0 rather than the formal order 4 of RK4, consistent with the semi-stiff behavior reported in the 1D case [1]: the diffusion terms impose severe stability constraints that effectively limit the achievable temporal accuracy of explicit integrators.

Table 3. Temporal convergence analysis. $M = 512$, $P = 4$ GPUs, $T_f = 1$.

Δt	$\|e_C\|_{L^2}$	Rate_C	$\|e_O\|_{L^2}$	Rate_O
2.0e−4	4.32e−5	—	2.19e−5	—
1.0e−4	1.14e−5	1.92	5.81e−6	1.91
5.0e−5	2.98e−6	1.94	1.52e−6	1.93
2.5e−5	7.51e−7	1.99	3.82e−7	1.99
1.0e−5	1.21e−7	1.99	6.15e−8	1.99

The empirical observation of second-order temporal convergence, despite the use of an RK4 integrator, is a formal consequence of stage order reduction in the presence of stiffness. In parabolic problems where the stability constraint enforces $\Delta t \propto h^2$, the internal stages of explicit Runge–Kutta methods fail to maintain full accuracy, leading to an effective convergence rate $O(\Delta t^{q/p})$ or, as observed, a degradation towards the second order consistent with the treatment of stiff diffusive terms.

Test 2. Strong Scaling.

Strong scaling is assessed by fixing the total grid at $M = 2048 \times 2048$ and varying P from 1 to 32 GPUs (1–8 LEONARDO Booster nodes). Table 4 reports wall-clock time for $T_f = 10$ time units.

The efficiency drop from 97.0 % at $P = 2$ to 65.9 % at $P = 32$ is analyzed via the Amdahl bound $\varepsilon(P) = (Pf_s + (1 - f_s))^{-1}$, with serial fraction $f_s \approx 1.2\,\%$ estimated from single-GPU Nsight Systems profiling. Amdahl predicts $\varepsilon(32) = (32 \times 0.012 + 0.988)^{-1} = 1/1.372 \approx 72.9\,\%$; the observed 65.9 % lies 7 % below this bound, corresponding

Table 4. Strong scaling on CINECA LEONARDO. $M = 2048 \times 2048$, $T_f = 10$.

GPUs	Nodes	Wall-clock (s)	Speedup	Efficiency (%)
1	1	487.3	1.00	100.0
2	1	251.2	1.94	97.0
4	1	131.1	3.72	92.9
8	2	71.4	6.82	85.3
16	4	39.8	12.24	76.5
32	8	23.1	21.09	65.9

to an effective serial fraction $f_s^{\text{eff}} \approx 1.7\,\%$. The additional loss is attributable to inter-node InfiniBand latency: $t_{\text{IB}} \approx 1.2\,\mu\text{s} + 81\,\text{KB}/22\,\text{GB/s} \approx 4.9\,\mu\text{s}$ per halo exchange per RK4 step becomes non-negligible at $P \geq 8$, where the per-GPU subdomain shrinks and the compute-to-communicate ratio falls below the full-overlap threshold.

Test 3. Weak Scaling.

Weak scaling fixes the per-GPU subgrid at 512×512 and scales the total problem size with P (Table 5).

Table 5. Weak scaling on CINECA LEONARDO. 512×512 per GPU, $T_f = 10$.

GPUs	Total Grid	Wall-clock (s)	Weak Efficiency (%)
1	512×512	31.2	100.0
4	1024×1024	32.5	96.0
16	2048×2048	34.1	91.5
64	4096×4096	36.3	86.0

The 86.0 % weak-scaling efficiency at 64 GPUs (4096×4096, 16 nodes) is consistent with Theorem 5.3: the theoretical bound gives $\varepsilon_{\text{weak}} \geq 1 - O(\sqrt{P}/M) = 1 - O(8/4096) \approx 99.8\,\%$, showing that the dominant efficiency loss ($\sim$14 %) is InfiniBand latency, not communication volume. GPU utilization (measured via Nsight Systems) exceeds 88 % for all subgrid sizes $n \geq 256 \times 256$, confirming that the asynchronous halo-overlap protocol effectively hides communication latency.

Remark 6.1 (Roofline Performance)**.** *The fused RHS kernel achieves 1.41 TFLOPS FP64 ($\approx$ 14.5 % of the 9.7 TFLOPS FP64 hardware peak), consistent with the memory-bandwidth roofline limit $I_\alpha \times B_{\text{HBM}} = 1.19 \times 2{,}000 = 2{,}370$ GFLOPS (i.e., 59 % of the roofline is achieved). Temporal blocking or s-step methods [15] could raise this toward the roofline ceiling, at the cost of increased register pressure.*

Remark 6.2 (Invariant Region Verification)**.** *Theorem 5.2 is confirmed numerically: all five component fields satisfy the bounds (17) at every one of the $4096^2 = 1.68 \times 10^7$ spatial grid points (distributed across $P = 64$ GPUs) and at every time step. In particular, $C(\boldsymbol{x}, t) \in [0, 1]$ and $O(\boldsymbol{x}, t) \in [0, O_{\max}]$ are maintained throughout, confirming that parallel decomposition introduces no spurious violations of physical constraints.*

7. Conclusions

We have presented a mathematically rigorous and computationally efficient GPU-parallel domain decomposition framework for the extended 2D tumor angiogenesis PDE system of De Luca and Marcellino [1]. Four theorems establish: (1) the parallel RK4 stability constraint $\Delta t \leq h^2/(4d_{\max})$, independent of the number of subdomains P; (2) convergence of order $O(h^2 + \Delta t^4)$ uniformly across all GPUs (Theorem 5.2); (3) preservation of the bounded invariant region $\mathcal{R}$ under ghost-cell exchange (Theorem 5.2); (4) an efficiency bound $\varepsilon(P) \geq 1 - O(\sqrt{P}/M)$ (Theorem 5.3). Numerical experiments on CINECA LEONARDO confirm second-order spatial accuracy, 92.9 % single-node strong-scaling efficiency (4 A100 GPUs), and 86.0 % weak-scaling efficiency at 64 A100 GPUs (4096×4096 grid). The framework provides a robust, open-source computational platform for physiologically realistic parameter studies, drug efficacy modeling, and multi-scale tumor vascular network simulations.

Acknowledgements

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. Computational resources on CINECA LEONARDO were provided under ISCRA-C project HP10CI4Q69. This project is supported by INdAM - GNCS, Italy project 2026 "Metodi Numerici e Machine Learning per Sistemi Dinamici Complessi: Identificazione, Approssimazione", ref. no. CUP E53C25002010001. De Luca P. and Marcellino L. are member of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

References

[1] De Luca P, Marcellino L. Analytical and numerical properties of an extended angiogenesis PDEs model. *Journal of Mathematical Biology* 2025;91:62. https://doi.org/10.1007/s00285-025-02293-y
[2] 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.
[3] 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.
[4] De Luca P, Galletti A, Giunta G, Marcellino L. A numerical approach for a 1D tumor-angiogenesis simulations model. *Applied Numerical Mathematics* 2025;210:83–94. https://doi.org/10.1016/j.apnum.2024.11.017
[5] De Luca P, Marcellino L. Conservation law analysis in numerical schema for a tumor angiogenesis PDE system. *Mathematics* 2025;13(1):28. https://doi.org/10.3390/math13010028
[6] Henry D. *Geometric Theory of Semilinear Parabolic Equations*. Lecture Notes in Mathematics, vol. 840. Springer, Berlin; 1981.
[7] Temam R. *Infinite-Dimensional Dynamical Systems in Mechanics and Physics*, 2nd ed. Applied Mathematical Sciences, vol. 68. Springer, New York; 1997.
[8] Toselli A, Widlund O. *Domain Decomposition Methods—Algorithms and Theory*. Springer Series in Computational Mathematics, vol. 34. Springer, Berlin; 2005.
[9] Srinivasan B. A guide to the Michaelis–Menten equation: steady state and beyond. *FEBS Journal* 2022;289(20):6086–6098.
[10] Cornish-Bowden A. One hundred years of Michaelis–Menten kinetics. *Perspectives in Science* 2015;4:3–9.
[11] Anderson ARA, Chaplain MAJ. Continuous and discrete mathematical models of tumor-induced angiogenesis. *Bulletin of Mathematical Biology* 1998;60(5):857–899.
[12] Mohammad Mirzaei N, Kevrekidis PG, Shahriyari L. Oxygen, angiogenesis, cancer and immune interplay in breast tumour microenvironment: a computational investigation. *Royal Society Open Science* 2024;11(12):240718.
[13] Botti L, Piccinelli M, Ene-Iordache B, Remuzzi A, Antiga L. An adaptive mesh refinement solver for large-scale simulation of biological flows. *Int. J. Numer. Meth. Biomed. Eng.* 2010;26(1):86–100.
[14] Lee YC, Yu JC, Ni K, et al. Improved prediction of anti-angiogenic peptides based on machine learning models and comprehensive features from peptide sequences. *Scientific Reports* 2024;14:14387.
[15] Datta K, Murphy M, Volkov V, Williams S, Carter J, Oliker L, et al. Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In: *SC'08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. IEEE; 2008. p. 1–12.
[16] Goldstein JA. A perturbation theorem for evolution equations and some applications. *Illinois Journal of Mathematics* 1974;18(2):196–207.
[17] CINECA. LEONARDO Booster: Technical Specifications and User Guide. 2024. https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.1
[18] Tao Y, Winkler M. Global existence and boundedness in a chemotaxis-haptotaxis model with self-constriction. *Journal of Differential Equations* 2012;252(1):152–181.
[19] Xiang T. Global existence and boundedness in a 2D chemotaxis-haptotaxis system with logistic source. *Discrete & Continuous Dynamical Systems - B* 2018;23(4):1453–1475.
[20] Hundsdorfer W, Verwer JG. *Numerical solution of time-dependent advection-diffusion-reaction equations*. Springer Science & Business Media; 2013.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 75–84

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

Distributed Large-Scale Traffic Simulation for Urban and Regional Mobility Analysis

Paulo Silva[a,*], Pavlína Smolková[a], Radek Furmánek[a], João Barbosa[a], Matej Špeťko[a], Emanuele Vitali[b], Kateřina Slaninová[a], Jan Martinovič[a]

[a]*IT4Innovations, VSB - Technical University of Ostrava, Ostrava, Czech Republic*
[b]*CSC - IT Center for Science, Espoo, Finland*

Abstract

Large-scale traffic simulation is increasingly relevant for urban and regional mobility analysis, particularly in scenarios that require extensive routing computations, repeated executions, and substantial data generation. In such settings, distributed and high-performance computing resources are important not only for reducing execution time, but also for enabling simulation campaigns of practical scale. This work builds on our previous efforts in distributed route computation and focuses on broader large-scale simulation workloads with Ruth, a mesoscopic traffic simulator designed for large-scale scenario execution and mobility data generation. The methodology relies on distributed simulations involving over 900,000 vehicles on regional road networks exceeding 190,000 nodes, together with runtime observations from different communication backends and execution modes. We present execution results obtained on Karolina and discuss simulator-side and runtime-level optimisations supported through EuroHPC access to Karolina and LUMI, which enabled larger workloads and more efficient simulations. The work also considers a dedicated front-end connected to the LEXIS Platform to simplify access for non-expert users, as well as exploratory integration with external decision-support environments such as FLOREON+, including support for scenario visualisation and analysis. This work extends the use of Ruth from kernel-level distributed execution towards broader urban and regional simulation. The findings highlight practical aspects of execution, access, and data generation in distributed environments, while supporting ongoing optimisation efforts and larger future scenarios, including nation-wide traffic simulation studies.

Keywords: Traffic simulation; distributed computing; high-performance computing; mobility analysis; route computation

1. Introduction

Urban and regional mobility studies increasingly rely on simulation campaigns that go beyond isolated test cases. Evaluating large routing workloads, repeating simulations under multiple parameter settings, and generating sizeable mobility datasets place pressure not only on the modelling side, but also on execution time, workflow management, and

* Corresponding author. Tel.: +420 597 329 500.
E-mail address: paulo.silva@vsb.cz (Paulo Silva).

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.021

downstream data handling. These issues become more pronounced at regional scale, where campaigns may involve hundreds of thousands of vehicles, millions of route computations, and substantial post-processing requirements[1, 2].

A number of established traffic simulators provide strong support for detailed traffic modelling and traffic-management studies. Widely used examples include SUMO, VISSIM, MATSim, and CityFlow, each with different modelling assumptions and execution targets [3–6]. At the same time, many practical studies require a different balance between behavioural detail and execution throughput, especially when the objective is to run repeated simulations, evaluate large routing workloads, or generate sizeable synthetic mobility datasets for downstream analysis. In such settings, the challenge is not only to simulate traffic realistically, but also to execute, manage, and reuse large campaigns efficiently across heterogeneous computing environments.

Ruth addresses this space as a deterministic mesoscopic traffic simulator designed for large-scale route computation, traffic flow analysis, and mobility data generation [7]. Rather than focusing on microscopic vehicle behaviour, Ruth is aimed at workloads in which distributed execution, scenario repetition, and high-throughput processing are central. Our previous work reported on distributed route computation with Ruth and analysed initial scalability behaviour on Prague and Boston scenarios, using the EVKIT-based execution layer and identifying where routing ceased to dominate overall runtime [2]. Since then, the simulator and its workflow have evolved in several directions. These include larger regional simulations, multiple execution modes, broader integration paths for downstream use, and continued development of the execution layer through the Asynchronous Communication and Execution (ACE) library [8, 9].

In this paper we shift the emphasis from kernel-level distribution alone to a broader view of large-scale simulations and their operational workflow. Ruth is considered here as a distributed traffic-simulation system for urban and regional mobility analysis, with emphasis on simulation execution, workflow flexibility, and mobility data generation. In addition to direct execution on High-Performance Computing (HPC) systems, in this paper we present a dedicated Ruth front-end connected to the LEXIS platform as a higher-level access path for users who prefer not to interact directly with batch scripts and cluster environments [10, 11]. We also discuss exploratory integration with external decision-support environments such as FLOREON+, which combines layers related to hydrological situation, air quality, traffic, and landslides or land depressions, while also supporting the ingestion and further interpretation of simulation outputs.

The main contributions of this paper are fourfold. First, we report representative urban and regional workloads that document the progression from the earlier Prague and Boston studies to a substantially larger Central Bohemian Region (CBR) scenario. Second, we describe the operational workflow of Ruth beyond the internal simulator architecture, including multiple execution modes, portal-based access, generated outputs, and downstream analysis and visualisation paths. Third, we report runtime-evolution results for the current execution layer, comparing the earlier EVKIT implementation with ACE configured both with MPI and ZeroMQ backends, and local optimizations on the simulator. While the SC24 and SC25 poster versions reported ACE results only for the MPI-based execution path, the present paper also includes ACE (ZeroMQ) results as part of the current backend-validation efforts [8, 9]. Fourth, we discuss practical implications of large-scale simulation campaigns in terms of data generation, post-processing, execution portability across HPC systems, and continued use of generated outputs in downstream analytical workflows. Overall, we present Ruth as a distributed simulation workflow, with the CBR scenario as the main case study, ACE as the execution-layer evolution, and LEXIS and FLOREON+ as operational integration paths.

The remainder of the paper is organised as follows. Section 2 introduces Ruth, its execution modes, and its workflow integration paths. Section 3 presents the experimental setup, including scenario classes, compute environments, simulation settings, and the higher-level operational workflow. Section 4 reports the main results, covering regional simulations, runtime evolution and backend validation, and workflow implications for large-scale data production and downstream use. Section 5 concludes the paper and outlines ongoing work.

2. Ruth, Execution Modes, and Workflow Integration

2.1. Simulator overview

Ruth combines vehicle movement, routing, queue management, and a global network view into a single mesoscopic simulation workflow. In terms of modelling scope, it sits between microscopic tools such as SUMO, MATSim,

and CityFlow and larger traffic-assignment environments such as Dynameq and TransModeler [3, 5, 6, 12, 13]. Its main value proposition is support for routing-intensive mesoscopic campaigns with repeated execution, distributed route computation, large outputs, and downstream analysis workflows. This distinguishes Ruth from large-scale uses of MATSim or SUMO, where scalability primarily serves agent-based or microscopic studies rather than routing-intensive mesoscopic campaigns. Ruth is released under the GPL-3.0 license, which supports research-oriented use, reproducibility, and community-driven development. The global view tracks vehicles on each segment over time and updates segment speeds according to traffic density. Vehicles move along precomputed routes, but route changes can be triggered when updated segment conditions justify recomputation, and each simulation step produces FCD records describing time, vehicle identity, and segment position. Determinism is ensured through a user-defined seed and sorting of distributed route-computation results by vehicle ID before merging.

The simulator architecture is shown in Figure 1. The core simulator contains vehicle advancement, routing, queue handling, and output generation. The most computationally expensive components remain the distributed workloads associated with alternative-route computation and Probabilistic Time-Dependent Routing (PTDR) [14]. This separation is useful in practice because not all parts of the simulator scale in the same way. Route computation benefits from distribution earlier than other functions such as route selection and vehicle advancement, which become more visible as routing time is reduced.

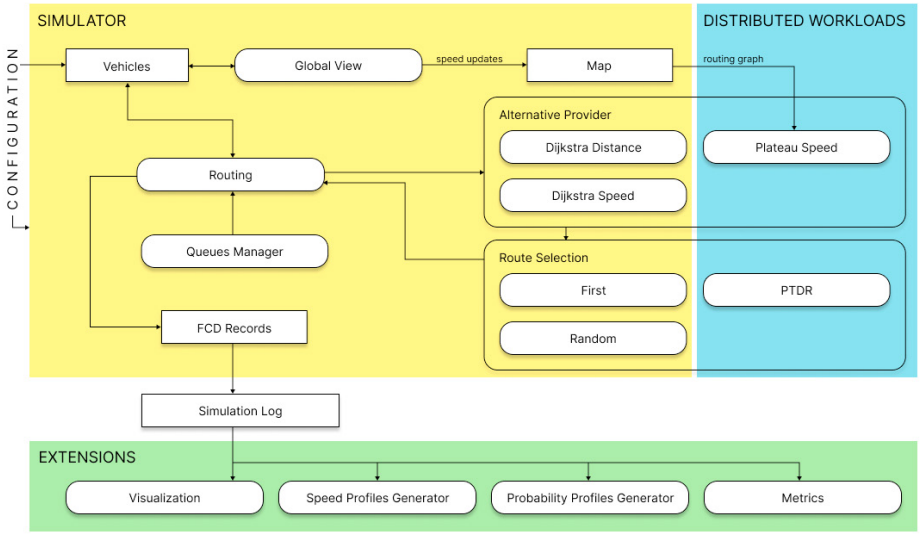


Fig. 1: High-level architecture of Ruth. Alternative-route computation and PTDR remain the dominant distributed workloads, while route selection and subsequent vehicle advancement become increasingly visible as routing time is reduced.

2.2. *Execution modes and the dedicated Ruth front-end*

Ruth is not tied to a single execution mode. It can be executed locally for development, debugging, and smaller-scale simulation scenarios, submitted directly to an HPC system through the usual batch workflow, or accessed through a dedicated web front-end built on the LEXIS platform [10, 11]. The simulator source code is available through the public Ruth repository [7], and the dedicated front-end is available at `https://ruth.lexis.tech`. The front-end does not replace the native execution path. Its role is to provide an ergonomic abstraction layer for users who prefer a higher-level interface to the simulator instead of direct interaction with the cluster environment.

The dedicated Ruth entry point is separate from the main LEXIS portal and is tailored to Ruth-specific scenario selection, parameter configuration, and job submission. In practical terms, it provides a curated interface to the simulator rather than a general-purpose cluster interface. The interface can be used to revisit previously executed simulations (as shown in Figure 2), load predefined scenarios and simulation settings, or repeat simulation templates, offering a higher-level path to HPC-backed execution without requiring direct interaction with batch scripts, software environ-

ments, or file-staging details. The LEXIS-based front-end is therefore relevant not because it is the only way to run Ruth, but because it makes large simulations more accessible to a wider user group while preserving the option of direct cluster submission for more complex workflows.

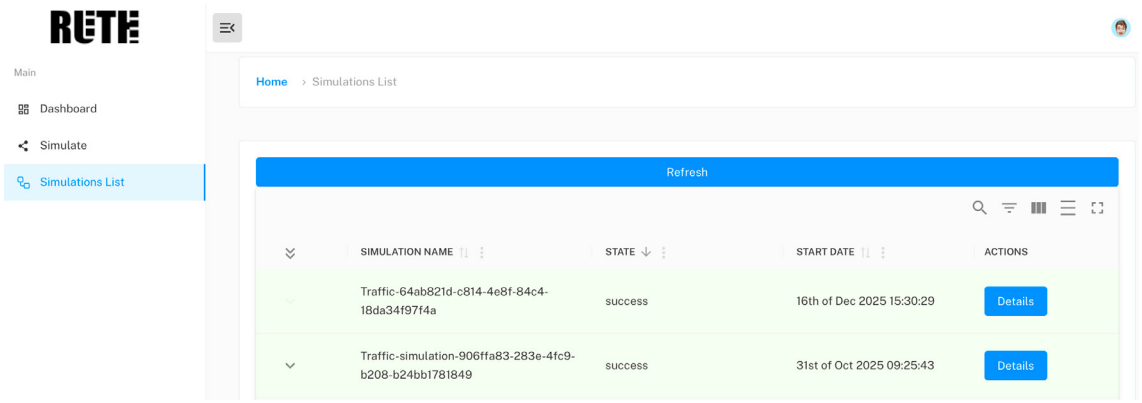


Fig. 2: Dedicated Ruth front-end built on the LEXIS platform. The interface offers an ergonomic entry point for non-expert users, while direct local and HPC-native execution remain available for development and production workflows.

2.3. *Integration with external visualisation and decision support*

We have performed integration work that makes Ruth outputs compatible with external environments that support scenario inspection and communication of results. Figure 3 shows a Central Bohemian scenario visualised through FLOREON+, where segment colours represent simulated traffic conditions over time. This integration is downstream from the simulator. Ruth remains responsible for execution and data generation, while FLOREON+ provides a map-based environment for viewing the outputs, navigating time, and inspecting regional congestion patterns through a spatial interface with timeline controls [15].

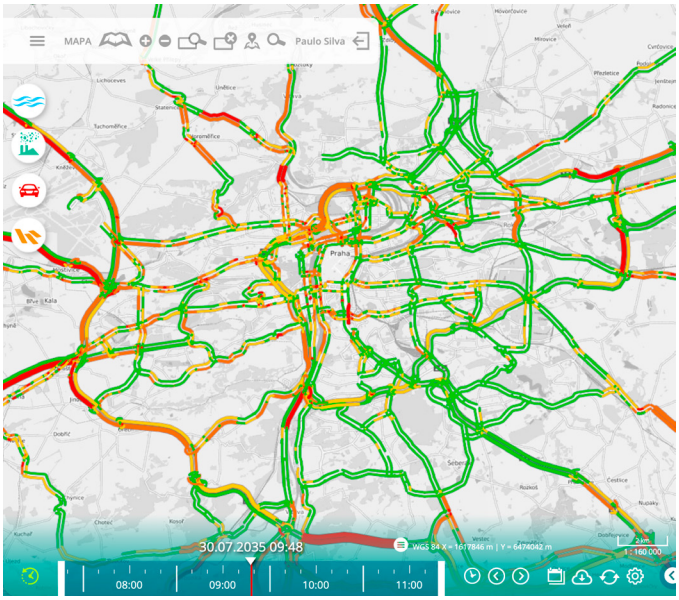


Fig. 3: Ruth output visualised in FLOREON+ for a 2035 traffic scenario in the Central Bohemian Region, Czech Republic. The external interface is used for map-based inspection of segment-level traffic states over time.

As large simulation campaigns generate outputs that are computationally demanding to visualise with standard approaches and difficult to interpret directly from raw files or aggregate statistics alone, this integration supports downstream analysis, interpretation, and further system-level coupling. The external interface provides a practical way to inspect how congestion propagates spatially and temporally across the network and to communicate the results to users who are not working inside the simulation environment. It also provides a natural basis for future studies involving disruption, infrastructure, and emergency-related scenarios, although those use cases are not evaluated quantitatively in this paper.

3. Experimental Setup

3.1. Simulation and routing workloads

Table 1 summarises the representative scenarios discussed in this paper. These scenarios cover two distinct workload classes: end-to-end traffic simulations, in which vehicles are advanced through the network and FCD outputs are generated, and route-computation workloads, in which only alternative routes are computed on the graph without executing the full vehicle simulation. Prague and Boston remain useful reference cases because they were used in the earlier distributed-routing study [2]. The new regional scenario, referred to here as CBR, is the main scale extension. It is substantially larger than the previously reported Boston graph and supports campaign configurations with up to 936,000 vehicles. Representative runs generate approximately 50 GB of FCD output per simulation, with total output volume increasing further when repeated executions are performed for statistical analysis or alternative scenario settings.

The purpose of this overview is not to define a common benchmark for direct comparison across all scenarios. Instead, it documents the progression in graph size, workload class, and simulation scale across successive stages of development. Detailed results are reported only for the scenarios directly relevant to the analyses presented in Section 4.

# - Scenario	Nodes	Segments	Workload Class	Sample Size	Relevance
1 - Boston	51,181	125,888	Full Simulation	300,000 vehicles	Previous study
2 - CBR (Central Bohemian Region)	194,585	478,517	Full Simulation	936,000 vehicles	Large regional campaign
3 - Prague	21,817	49,540	Route Computation	2,627,640 routes	Routing backend evolution (Urban area)
4 - CBR (Central Bohemian Region)	194,585	478,517	Route Computation	25,000,000 routes	Routing backend evolution (Regional area)

Table 1: Overview of representative scenarios discussed in this work. The table documents the evolution in workload scale and scenario scope and it is not intended as a uniform benchmark set for direct performance comparison across all cases.

3.2. Compute environments and project access

The experimental work was executed on the CPU partition of Karolina supercomputer at IT4Innovations and on LUMI-C partition of Lumi supercomputer at IT Center for Science (CSC) [16, 17]. These clusters were used both for direct batch execution and for workflows exposed through the dedicated Ruth front-end. Smaller routing studies used one to eight nodes for backend comparison, while the earlier Boston study used up to fifty Karolina nodes and remains a useful baseline for understanding where routing stops dominating the total runtime [2].

All execution results reported in this paper were performed on Karolina CPU. LUMI-C was nevertheless used extensively for additional runs, validation, backend testing, workflow verification, and repeated campaign execution during the development process. The purpose of using both systems was therefore not to perform a direct cluster-

to-cluster comparison, but to support validation of the simulator and its execution workflow across distinct HPC environments. A systematic comparison between Karolina and LUMI will be part of future work.

The EuroHPC access relevant to this paper was awarded through project EU2024D11-072, *Scalable and Intelligent Traffic Modeling for Improved Urban Planning*. The allocated compute time on these systems (4000 node hours on Karolina CPU and 7000 node hours on LUMI-C) supported not only the reported execution results, but also repeated runs for calibration, parameter tuning, validation, and statistical analysis under alternative scenario settings. It also enabled post-processing and aggregation of large simulation outputs, and verification of execution behaviour across distinct HPC environments while the simulator runtime and backend continued to evolve.

3.3. Simulation settings

The simulator ingests (as inputs) a network description, an Origin–Destination matrix, and a configuration file that controls route computation, map updates and various other simulation settings. Across the reported simulations, the main parameters were kept within a stable range in order to preserve continuity with the earlier Prague/Boston study while the execution layer evolved. In particular, the round frequency was set to 5 s or 60 s, the number of computed alternatives was varied between 1 and 3, map updates were performed every 15 or 60 s, the LoS vehicle tolerance was set to 5 s, the travel-time threshold for route changes varied between 0, 5%, and 10%, and the saving interval ranged from 0 to 100 s.

3.4. Operational workflow and execution modes

Figure 4 provides a higher-level view of Ruth that complements the internal architecture shown earlier. It summarises how input data, execution modes, compute environments, outputs, and visualisation options are connected in practice. The workflow starts from the Origin–Destination matrix and simulation settings. Ruth can be executed locally for development, debugging, and smaller scenarios, submitted directly to HPC systems through command-line or batch workflows, or accessed through a dedicated front-end connected to the LEXIS platform. This front-end provides a higher-level interface for configuring and launching simulations without requiring direct interaction with batch scripts or file-staging details. In the experiments discussed in this paper, Ruth was executed on Karolina and LUMI, although the workflow is not restricted to these systems. The execution produces FCD data, metrics, and logs, which can be used for downstream processing and analysis. These outputs can also be visualised through Ruth's built-in 2D viewer or integrated with external environments such as FLOREON+.

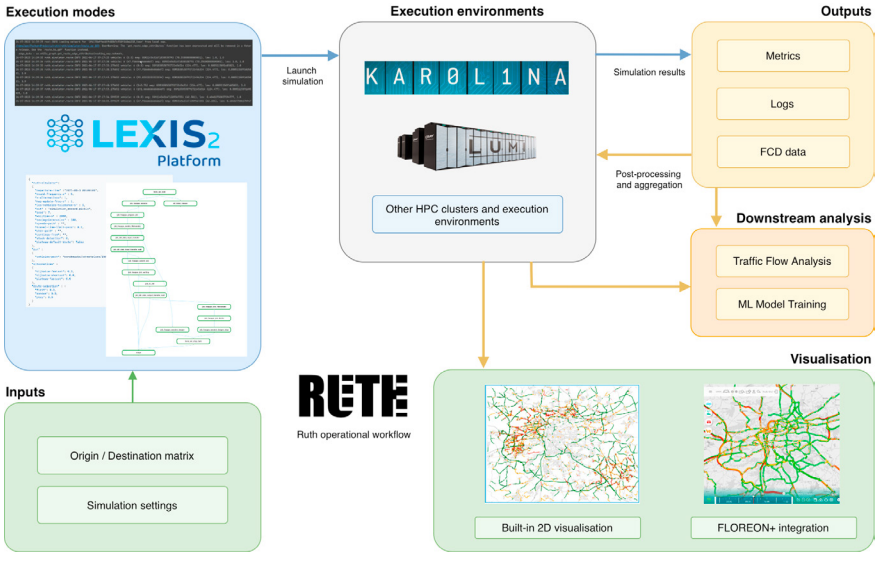


Fig. 4: Higher-level operational view of Ruth, showing the relation between input data, execution modes, compute environments, generated outputs, downstream analysis, and visualisation/integration options.

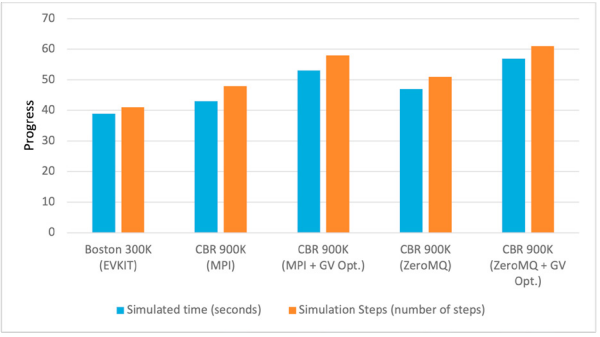
4. Results and Discussion

4.1. Regional-scale campaign evolution

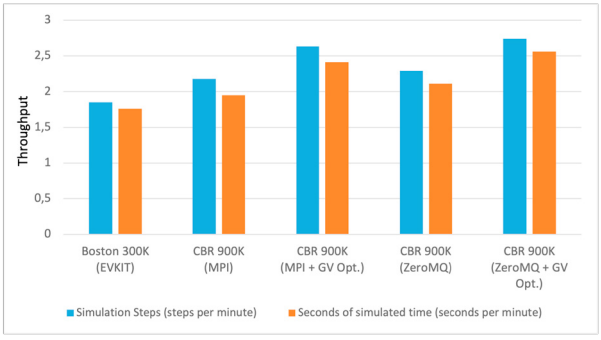
The main step beyond our previous work is the shift from city-scale workloads toward larger regional campaigns. Our earlier study reported Prague experiments with 10,000 and 60,000 vehicles and a Boston study with up to 300,000 vehicles [2]. The CBR regional scenario extends this substantially. The graph contains 194,585 nodes and 478,517 segments, and simulation configurations reach over 900,000 vehicles. Representative single runs produce on the order of 50 GB of FCD data.

At this scale, simulation management becomes part of the experimental workload. The elapsed time of the routing kernel remains important, but it is no longer sufficient to characterise the cost of the simulation. Storage of FCD outputs, movement of files between systems, repeated executions, and post-processing of multiple runs all contribute to the effective cost of the simulations. This is particularly relevant when statistical analyses require multiple repetitions or when scenario exploration is performed across several parameter settings, since the cumulative output volume grows quickly beyond the size of a single run.

Figure 5 complements these campaign characteristics by showing how far different implementations advance the simulation within approximately the same wall-time limit of 22 minutes. The comparison should be read as an end-to-end progress indicator rather than as a controlled microbenchmark of a single implementation change. The Boston EVKIT run is included as a historical baseline, while the CBR ACE runs reflect the current execution layer and, where applicable, additional simulator-side optimisation denoted as GlobalView Optimization (GV Opt.). Under a comparable execution-time budget, the ACE-based CBR runs reach a later simulated time horizon and complete more simulation steps than the earlier Boston EVKIT case, despite operating on a larger graph and a larger traffic workload.



(a) Simulation progress under a fixed wall-time limit of 22 minutes.



(b) Normalized execution throughput per minute of wall-clock computation.

Fig. 5: Comparison of simulation progress and execution throughput under a fixed wall-time limit of approximately 22 minutes on Karolina using 2 nodes in all cases. Figura (a) shows simulated time reached and completed simulation steps. Figure (b) shows normalized throughput, expressed as completed steps per minute of wall-clock computation and simulated seconds per minute of wall-clock computation. The Boston EVKIT run is shown as a historical baseline, while the CBR ACE and CBR ACE + GV Opt. runs reflect the current execution layer and simulator-side optimisation. The figure is intended as an end-to-end progress comparison rather than as a controlled microbenchmark of individual implementation changes.

In addition, these outputs are subsequently reused in later processing and modelling workflows, including dataset preparation for the InnovAIte project [1]. In practice, raw FCD outputs from multiple simulation configurations are transformed through a high-throughput Parsl-based pipeline into structured mobility datasets [18]. The downstream models are outside the scope of this paper, but they motivate the need for reproducible, repeated simulation campaigns. The same execution model is currently being extended beyond regional studies toward nation-wide scenarios, which further increases the importance of output management and stable distributed execution.

[1] https://innovaite.sk

4.2. Runtime evolution and backend validation

The execution layer has evolved from the EVKIT-based implementation described in the earlier work toward an ACE-based implementation that provides more direct control over communication and task execution. In the current version, ACE supports both MPI and ZeroMQ backends selected at compile time. This progression was first outlined in poster form at SC24 and SC25, where the reported ACE results were limited to the MPI-based execution path [8, 9]. In contrast, in this paper we also include ACE (ZeroMQ) results, reported here as part of the current runtime evolution and backend-validation efforts.

Table 2 summarises the Prague-scale route-computation workload with 2,627,640 route requests. The EVKIT column reports total elapsed time (i.e., computation plus worker-spawn overhead). Both ACE backends reduce elapsed time substantially relative to EVKIT. Across the reported configurations, ACE (MPI) and ACE (ZeroMQ) achieve comparable performance, with small differences depending on node count. At this scale, the main observation is that the ACE-based execution layer removes the large startup overhead that characterised EVKIT and provides a more direct basis for backend comparison.

Nodes	EVKIT total (s)	ACE (MPI) (s)	ACE (ZeroMQ) (s)	MPI speedup	ZeroMQ speedup
1	631.91	139	126	4.55×	5.02×
2	854.30	100	72	8.54×	11.87×
4	1,004.14	153	60	6.56×	16.74×
8	979.54	64	70	15.31×	13.99×

Table 2: Runtime evolution for 2,627,640 route requests on the Prague graph across the currently validated node counts. EVKIT total time includes both route computation and worker-spawn overhead.

The larger CBR request batch in Table 3 is more demanding, with 25,000,000 route requests on the regional graph. At this scale, ACE remains competitive with the EVKIT baseline across the reported node counts for both backends. The current results do not point to a single preferred backend. Instead, they show that both ACE (MPI) and ACE (ZeroMQ) are viable execution options, while also exposing configurations in which additional tuning and validation are still required. In the presented CBR runs, the ZeroMQ backend is slightly faster than MPI at the reported node counts.

Nodes	EVKIT total (s)	ACE (MPI) (s)	ACE (ZeroMQ) (s)	MPI speedup	ZeroMQ speedup
1	82,138.03	69,059	68,176	1.19×	1.20×
2	50,199.45	34,604	33,705	1.45×	1.49×
4	26,284.46	18,626	16,924	1.41×	1.55×
8	11,641.54	9,116	8,646	1.28×	1.35×

Table 3: Runtime evolution for 25,000,000 route requests on the CBR graph across the currently validated node counts.

Validation beyond 8 nodes is still ongoing, so those runs are not included in the reported tables. At this stage, the main issue is backend robustness at larger scale, especially for ACE (ZeroMQ), where port-binding conflicts during synchronisation and process startup can make some runs unstable. A revised implementation aimed at these spawning and connection-management issues is already under test, but the larger-node re-executions are left for future work.

Beyond the elapsed times reported in Tables 2 and 3, backend validation was also examined through a weak-scaling-style experiment on the CBR workload (shown in Table 4). These measurements are included as work in progress and should be interpreted as validation data for the current ACE backends rather than as final comparative performance claims. In this experiment, the number of requests increased proportionally with node count, and the objective was to observe whether ACE (MPI) and ACE (ZeroMQ) remained stable while maintaining comparable elapsed time over the reported range.

Nodes	Requests	ACE (MPI) (s)	ACE (ZeroMQ) (s)
1	1,000,000	2,977.45	2,767.88
2	2,000,000	2,955.24	2,775.52
4	4,000,000	3,021.64	2,816.61
8	8,000,000	3,208.79	2,875.38

Table 4: Weak-scaling-style backend validation on the CBR workload across the currently validated node counts. These measurements are included as work in progress and should be interpreted as validation data rather than as final performance claims.

Taken together, these results show that the execution layer is no longer limited to the EVKIT implementation and that ACE already supports both MPI and ZeroMQ backends across substantial route-computation workloads. At the same time, the current validation results identify backend-stability issues that still need to be resolved before extending the analysis to larger node counts. For this reason, the results are presented here as runtime evolution and backend-validation evidence rather than as a definitive comparison between communication strategies.

4.3. Workflow implications at campaign scale

Three main observations follow from the experiments. First, the larger regional simulations confirm that distributed traffic simulation is now constrained by more than alternative-route computation alone. Data movement and post-processing become significant once FCD outputs reach tens of gigabytes per run. Second, the access layer affects who can use the simulator and how repeatable the workflow can be. The dedicated Ruth front-end built on LEXIS lowers the barrier to scenario submission without replacing the native local or HPC execution paths. Third, the FLOREON+ integration provides a practical mechanism for inspecting outputs in space and time once the simulations are completed.

Taken together, these observations indicate that the current stage of Ruth is best characterised as a distributed simulation workflow rather than as an isolated routing benchmark. The relevant performance questions now involve execution, access, data production, and subsequent inspection within the same simulation setting.

5. Conclusion

We presented Ruth as a distributed simulation workflow for urban and regional mobility analysis, centred on the CBR scenario, the ACE execution-layer evolution, and operational integration through LEXIS and FLOREON+. Beyond our earlier work centred on distributed route computation, we reported broader simulation workloads, including regional campaigns with up to 936,000 vehicles and substantial FCD output per run. The reported results show that Ruth is no longer limited to smaller urban reference cases and can support more demanding regional studies with practical relevance for repeated scenario execution, downstream analysis, and mobility data generation.

We also described the evolution of the execution layer from the earlier EVKIT-based implementation toward ACE configured with MPI and ZeroMQ backends. This is also a point of progression with respect to the earlier SC24 and SC25 poster results, which reported ACE only for the MPI-based execution path. The current measurements show that ACE already provides a more flexible execution basis and competitive route-computation performance across the reported workloads, while also indicating that backend validation and optimisation at larger node counts are still ongoing.

In addition to execution performance, we highlighted workflow aspects that are important for large simulation campaigns. These include multiple execution modes, the dedicated LEXIS front-end as a higher-level access path, post-processing and aggregation of large simulation outputs, portability across Karolina CPU and LUMI-C, and downstream integration with environments such as FLOREON+. These aspects are relevant not only for obtaining individual execution results, but also for enabling repeated simulations, parameter variation, calibration, validation, and broader use of generated mobility datasets.

Ongoing work focuses on further optimisation of the execution layer, continued backend validation at larger scale, and extension toward even larger scenarios, including nation-wide traffic simulation studies. Assessing the realism of generated traffic outputs against observed counts, travel times, congestion patterns, or reference simulations is

an important next step, but is outside the scope of this paper. Additional work also concerns the continued use of generated datasets in downstream modelling and analysis workflows.

Acknowledgements

We acknowledge EuroHPC JU for awarding the project ID EU2024D11-072 access to 4000 node hours on Karolina CPU hosted by IT4Innovations and 7000 node hours on LUMI-C hosted by CSC. This work was also supported by the Ministry of Education, Youth and Sports of the Czech Republic through e-INFRA CZ (ID: 90254), by the InnovAIte Slovakia project under grant agreement No. 09I02-03-V01-00029 funded by the Slovak Recovery and Resilience Plan, and by the EPICURE project under grant agreement No. 101139786.

References

[1] Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. Largest: A benchmark dataset for large-scale traffic forecasting. *Advances in Neural Information Processing Systems*, 36:75354–75371, 2023.

[2] Paulo Silva, Pavlína Smolková, Sofia Michailidu, Jakub Beránek, Roman Macháček, Kateřina Slaninová, Jan Martinovič, and Radim Cmar. High-performance computing for distributed route computation in traffic flow models. *Procedia Computer Science*, 255:83–92, 2025. DOI: https://doi.org/10.1016/j.procs.2025.02.263.

[3] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Fl"otter"od, Robert Hilbrich, Leonhard L"ucken, Johannes Rummel, Peter Wagner, and Evamarie Wie"sner. Microscopic traffic simulation using sumo. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2575–2582, 2019.

[4] Martin Fellendorf. Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority. *64th Institute of Transportation Engineers Annual Meeting*, pages 1–9, 1994.

[5] Andreas Horni, Kai Nagel, and Kay W. Axhausen. *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, 2016. DOI: https://doi.org/10.5334/baw.

[6] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8797–8806, 2019.

[7] IT4Innovations. Ruth - traffic simulator. GitHub repository, 2026. https://github.com/It4innovations/ruth.

[8] Paulo Silva, Pavlína Smolková, et al. Assessing the impact of real-time traffic updates on traffic flow: A high-performance computing perspective on scalability and demand. Poster presented at SC24: The International Conference for High Performance Computing, Networking, Storage, and Analysis, 2024. https://sc24.supercomputing.org/proceedings/poster/poster_pages/post107.html.

[9] Paulo Silva, Pavlína Smolková, Kateřina Slaninová, Jan Martinovič, João Barbosa, Matej Špeťko, and Emanuele Vitali. Scalable alternative route computation with ace: A c++17 library for hpc traffic simulations. Poster presented at SC25: The International Conference for High Performance Computing, Networking, Storage, and Analysis, 2025. https://sc25.supercomputing.org/proceedings/posters/poster_pages/post115.html.

[10] LEXIS Platform. Lexis platform documentation. Online documentation, 2026. https://docs.lexis.tech/index.html.

[11] IT4Innovations. Ruth front-end on lexis. Web application, 2026. https://ruth.lexis.tech.

[12] Michael Mahut and Michael Florian. Traffic simulation with dynameq. *Fundamentals of Traffic Simulation*, pages 323–361, 2010.

[13] Ramachandran Balakrishna, Daniel Morgan, Howard Slavin, and Qi Yang. Large-scale traffic simulation tools for planning and operations management. *IFAC Proceedings Volumes*, 42(15):117–122, 2009.

[14] Emanuele Vitali, Davide Gadioli, Gianluca Palermo, Martin Golasowski, João Bispo, Pedro Pinto, Jan Martinovič, Kateřina Slaninová, João M. P. Cardoso, and Cristina Silvano. An efficient monte carlo-based probabilistic time-dependent routing calculation targeting a server-side car navigation system. *IEEE Transactions on Emerging Topics in Computing*, 9(2):1006–1019, 2021. DOI: https://doi.org/10.1109/TETC.2019.2919801.

[15] FLOREON+. Floreon+ platform. Web application, 2026. https://floreon.eu/.

[16] IT4Innovations. Karolina - hardware overview. System documentation, 2026. https://docs.it4i.cz/en/docs/clusters/karolina/hardware-overview.

[17] LUMI Documentation. Cpu nodes - lumi-c. System documentation, 2026. https://docs.lumi-supercomputer.eu/hardware/lumic/.

[18] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S. Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin M. Wozniak, Ian Foster, et al. Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 25–36, 2019.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 85–94

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

Coupling ESPResSo and waLBerla for scalable soft matter simulations with hydrodynamic interactions

Jean-Noël Grad[a], Ingo Tischler[a], Alexander Reinauer[a], Hideki Kobayashi[a], Michael Kuron[a], Frederik Hennig[b], Markus Holzer[b], Pedro Santos Neves[c], Satish Kamath[d], Christian Holm[a], Rudolf Weeber[a,*]

[a] *Institute for Computational Physics, University of Stuttgart, Allmandring 3, 70569 Stuttgart, Germany*
[b] *Chair for System Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstrasse 11, 91058 Erlangen, Germany*
[c] *Center for Information Technology, University of Groningen, Smitsborg, Nettelbosje 1, 9747 AJ Groningen, The Netherlands*
[d] *Compute Services, SURF, Science Park 140, 1098 XG Amsterdam, The Netherlands*

Abstract

Many problems in soft matter research involve particles suspended in a solvent, where hydrodynamic interactions play a crucial role. A well-established simulation approach for such systems combines molecular dynamics for the particles with the lattice-Boltzmann method for the solvent. We present the coupling of the coarse-grained molecular dynamics code ESPResSo to the waLBerla library, a high-performance framework for lattice-Boltzmann and other stencil-based methods. This coupling code, developed in the context of the EuroHPC Centre of Excellence MultiXscale, enables large-scale, multi-GPU simulations of coupled particle–fluid systems within ESPResSo. Beyond hydrodynamics, the integration also provides a diffusion–advection–reaction solver coupled to electrostatics and lattice-Boltzmann for electrokinetic simulations. A key advantage of using waLBerla is its code generation infrastructure, which facilitates both the adaptation of algorithms and their optimisation for different hardware architectures. We demonstrate the scalability of coupled simulations with multi-GPU benchmarks on MareNostrum 5.

Keywords: HPC; performance portability; molecular dynamics; lattice-Boltzmann method; electrokinetics; waLBerla; ESPResSo; EESSI

1. Introduction

Soft matter science is concerned with systems at the mesoscale, i.e., on the order of nanometres to micrometres, in which the strength of interactions is on a similar magnitude to the thermal energy. Many materials of technological and biological importance fall into this category. Examples include colloidal suspensions [1], polymer solutions [2], hydrogels [3], and biological cells [4]. In these systems, a solvent such as water is nearly always present. When studying the dynamics of soft matter, it is often necessary to include the long-range hydrodynamic coupling introduced

* Corresponding author.
E-mail address: rudolf.weeber@icp.uni-stuttgart.de

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.022

by the solvent into a simulation model. Doing this on an atomistic level, i.e., by tracking all solvent molecules, is typically not feasible due to the large number of molecules in even a cubic micrometer of material. It is also not necessary to know the trajectory of the solvent molecules, only the resulting flow field is needed.

In soft-matter physics, the dispersion of particles within a solvent represents a central model, suitable for systems such as colloidal dispersions, emulsions, polymer solutions, and biological fluids such as blood. In these systems, the mesoscopic structure and its internal degrees of freedom determine the macroscopic response to thermal fluctuations ($\sim k_BT$ at room temperature) and weak external forces. Solvent dynamics is essential to elucidate the rheological and dynamical properties of these dispersions. Solvent molecules not only drive Brownian motion through molecular collisions but also mediate long-range hydrodynamic interactions between dispersed particles. In non-equilibrium states, such as those induced by shear flow, the coupling between the fluid response and the particles' spatial configuration leads to complex, non-linear rheological phenomena. Moreover, there often is a separation of scales: even a 10 nm colloidal particle is approximately 30 times larger than a water molecule. Therefore, a common approach in soft matter is to model the system as interacting particles coupled to a solvent described at the continuum level using the Navier-Stokes equations. Various simulation techniques have been developed for such models [5, 6, 7, 8, 9]. A common choice is to couple coarse-grained molecular dynamics (CGMD) for the trajectories of the discrete particles with the lattice-Boltzmann (LB) method for the continuous solvent. While CGMD captures short-ranged and electrostatic forces between the particles, the LB method handles solvent dynamics by means of the Navier-Stokes equations. Thermal fluctuations, necessary to obtain Brownian motion, are included in both the LB solver and the coupling between CGMD and LB solver.

Although the coupling of CGMD and LB is a well-established approach, keeping up with changing scientific and HPC requirements even for the individual parts, CGMD and LB, requires considerable effort. Achieving high accuracy, performance, and scalability across diverse computing architectures while keeping the code maintenance effort at bay for coupled solvers remains a significant challenge [10]. One approach to address it is, rather than maintaining a combined code base, to couple well-established individual solvers for CGMD and LB by a lean bridge component. Here, we report on coupling ESPResSo, a CGMD package with a rich set of features for particle-based simulations, to waLBerla, an HPC-ready framework for stencil methods such as the LB method. The coupling lets ESPResSo make use of the multi-GPU capabilities and the flexibility provided by waLBerla's automated code generation in soft matter models including particle and solvent dynamics. Achieving these capabilities by adding a custom implementation of the LB method would not have been possible with the available resources.

2. The software packages

We briefly discuss the two codes, ESPResSo for the coarse-grained molecular dynamics and waLBerla for the LB method. ESPResSo is a versatile simulation package designed for research in soft matter systems, biological physics, and process engineering. Its core functionality is centred on CGMD to model physical and chemical processes at the mesoscale. A wide range of algorithms are provided: electrostatic interactions can be handled through several algorithms, including P^3M [11, 12], MMM1D [13], the induced charge computation (ICC*) method [14, 15], or the electrostatic layer correction (ELC) method [16, 17]. Hydrodynamic interactions are addressed through Stokesian Dynamics [18] or the LB method to solve the compressible Navier-Stokes equation. Chemical reactions within the system can be modelled at different levels of detail: through explicit bond breaking and formation [19], Monte Carlo methods [20], or on a continuum scale via a finite-volume diffusion–advection–reaction method [21]. ESPResSo combines a high-performance C++ backend with a Python interface, balancing computational efficiency with accessibility for the scientific community. Parallelisation of many-particle simulations is achieved through a hybrid MPI and shared memory approach. Additionally, many features offer GPU acceleration through CUDA, with support for execution on multiple graphics processing units.

WaLBerla is a multiphysics simulation software framework with a primary focus on stencil methods such as LB hydrodynamics. Its design emphasises scalability for high-performance computing (HPC) applications. Parallelisation strategies and data distribution, both on CPU and GPU, are specifically engineered to deliver excellent performance on peta- and exascale supercomputers. A key feature of waLBerla is its reliance on automated code generation, i.e., hardware specific kernels are automatically generated from the mathematical definition of the respective simulation algorithm by the pystencils package [22]. The generated kernels can leverage vectorisation, CUDA, HIP, or OpenMP,

ensuring optimised execution across diverse hardware platforms. For the LB method, the precise mathematical formulation is automatically constructed by the lbmpy library [23, 24]. This allows the LB method to be flexibly adapted to the requirements of a given system, for instance, by selecting or customising the LB collision model. In the context of soft matter, this framework is used, e.g., to add thermal fluctuations to the fluid dynamics.

ESPResSo is available on EESSI [25], the European service for streaming scientific software. EESSI releases are built for multiple CPU microarchitectures with hardware-specific optimisations and for multiple CUDA compute capabilities, so that the best performance is achieved on relevant HPC hardware. Quality is ensured through the ReFrame [26] framework, with a benchmark testsuite that measures both the weak scaling efficiency of ESPResSo and the correctness of the simulations on EuroHPC JU supercomputers [27, 28]. These benchmarks are run periodically on HPC clusters[1]. Pre-releases of ESPResSo that included the waLBerla bridge were made available a year before the official 5.0.0 release on the EESSI development repository[2] and were used in the continuous integration workflows of client software pyMBE [29, 30] and SwarmRL [31] via the EESSI GitHub Action[3].

3. Coupling waLBerla to ESPResSo

Both ESPResSo's LB implementation and the finite volume diffusion–advection–reaction solver (used mainly for electrokinetics) make use of waLBerla. Rather than directly using waLBerla as a library from within ESPResSo, we implemented a waLBerla bridge component. This is, in essence, a separate library which handles the details of setting up the solvers and their lattices using waLBerla in an MPI-parallel environment. The library then provides a lean interface for obtaining and changing the flow field, applying forces, changing boundary conditions, and running the solver. In this way, the implementation details of waLBerla are hidden from the CGMD code.

The LB and electrokinetics (EK) solvers in ESPResSo heavily rely on waLBerla's code generation capabilities [32]. At the core of stencil methods such as the LB method and finite volume solvers are fields stored on a lattice and update rules which describe how the value at one lattice site is updated based on the values of its neighbours. The pystencils package [22] provides the framework necessary to define these fields and update the rules. The latter are defined in terms of symbolic math using SymPy [33]. Based on those definitions, pystencils generates the C++ or CUDA code of kernels to apply the update rule efficiently on different hardware targets. The kernels can then be executed either directly from Python or via the MPI-parallel waLBerla framework. The latter is used by ESPResSo's waLBerla bridge. The LB building blocks, such as the stream-collide operation, are provided by lbmpy [23, 24] as kernels expressed in the algebraic language of pystencils. ESPResSo adds further definitions for other aspects, such as Lees–Edwards boundary conditions. The complete architecture of the code generation workflow is depicted in Fig. 1.

The use of automatic code generation offers several advantages. First, the code can be generated for different hardware targets, making use of target-specific optimisations such as vectorisation and parallelism. Rather than requiring domain-specific expertise to manually tune code for each target platform, performance portability is handled by pystencils. Second, based on knowledge of the mathematical properties of the numerical method, front-end libraries such as lbmpy may optimise kernels using domain-specific algebraic transformations that go beyond the capabilities of general-purpose compilers [24]. Finally, using automated code generation makes it much easier for researchers to modify or enhance the algorithms: rather than dealing with hardware-specific, optimised code, they can focus on the mathematical definitions, while optimisation and portability are handled by the code generation framework.

In practical terms, the waLBerla bridge is distributed as part of ESPResSo. It uses waLBerla as a library, which is obtained via CMake's FetchContent mechanism and built alongside ESPResSo. As the code generation has additional dependencies, we chose to ship pre-generated C++ and CUDA source files for the kernels with ESPResSo. However, should users wish to customise the implementation, the generation pipeline is provided as part of the ESPResSo source tree. Users of ESPResSo do not have to be aware of these architectural details. The LB and EK solvers as well as the coupling of particle and solvent dynamics are controlled via ESPResSo's Python interface along with the CGMD aspects of the simulation.

[1] ReFrame testsuite dashboard: https://dashboard.eessi.io/
[2] EESSI development repository: https://www.eessi.io/docs/repositories/dev.eessi.io/
[3] EESSI GitHub Action: https://github.com/marketplace/actions/eessi

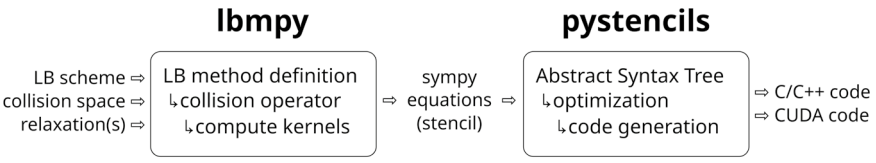


Fig. 1. Code generation workflow.

4. Coupling particle and solvent dynamics

ESPResSo implements two schemes to couple the dynamics of the particles with that of the solvent: momentum-exchange coupling [34] and inertialess tracers [35]. The former is typically used to couple ions, polymers, and colloidal particles, while the latter is used to model particles that follow streamlines or that make up the surface of a deformable object, such as a red blood cell.

For momentum-exchange coupling, the fluid velocity at the particle position is obtained by interpolation from the nearest LB grid points. A coupling force is then calculated based on the velocity difference between fluid and particle. This force is applied to the particle, the opposing force to the fluid by back-interpolation onto the nearest LB grid points. This is illustrated in Fig. 2. In contrast, Newton's equation of motion is not integrated for inertialess tracers. Instead, these particles are propagated solely based on the fluid velocity at their position. Forces on the tracers due to their interaction with other particles are directly applied to the fluid at the tracers' position. For example, multiple tracers can be connected with bonds to create soft deformable objects, such as red blood cells. This is often referred to as the immersed boundary method [36].

Aside from the two main coupling schemes, it is also possible to use particle positions as force sources on the fluid. This approach enables the modelling of self-propelled particles, for instance, by introducing a force dipole on a raspberry particle [37]. In such a swimmer model, the particle is made up of multiple beads, and two of these beads exert opposing forces on the surrounding fluid. The forces on these two beads can be directed toward or away from one another, thereby generating a dipole field within the fluid. If the centre of mass of the raspberry particle does not coincide with the centre of the dipole field, the resulting net force propels the swimmer. This is illustrated in Fig. 3.

All these coupling methods rely on two basic calls to ESPResSo's waLBerla bridge: getting fluid velocities at given positions and applying forces to the fluid at given positions. Both involve interpolation between the continuous particle positions and the LB lattice points surrounding the particle. The interpolation is handled by the waLBerla bridge. When running LB on a GPU, this interpolation is done on the device. There are two kernel calls for the coupling: the first to obtain interpolated fluid velocities and the second to apply forces to the fluid. The corresponding data transfer consists of 3 floating point values per particle for the former and 6 for the latter. This results in typical transfer volumes of 10 MiB per device for 300 000 particles in single-precision. Hence, the CPU-to-GPU data transfer is not a major performance concern. The actual state of the LB solver, approximately 50 floating point values per LB site, is never communicated to the CPU unless requested by the user for input/output (I/O).

In an MPI-parallel simulation, the same domain decomposition is applied in the CGMD simulation and the LB solver. In both cases, ghost layers are employed to handle the interaction with neighbouring MPI ranks. For the LB method, the ghost layers at each side of the MPI rank's volume consist of a single layer of LB cells. For the EK method, two layers of EK cells are needed to handle flux boundary conditions. On the CGMD side, the size of the ghost layer is determined by the interaction range of short-range forces such as the Lennard-Jones interaction. This has to be taken into account for the particle coupling: a particle close to the boundary, edge or corner of an MPI rank's local volume may, due to the interpolation to surrounding lattice sites, also contribute coupling forces to the volume controlled by a neighbouring MPI rank. This happens by also coupling the ghost of the particle on that neighbouring MPI rank. Care has to be taken, not to miss or double-count coupling forces. This is shown schematically in Fig. 4.

Thermal fluctuations are a key factor in the dynamics of soft matter. When included in the modelling, they have to be applied both to the fluid and to the particles. The former is done by adding random stresses to the fluid in the collision operator, the latter by adding a random term to the particle-fluid coupling force [40, 41].

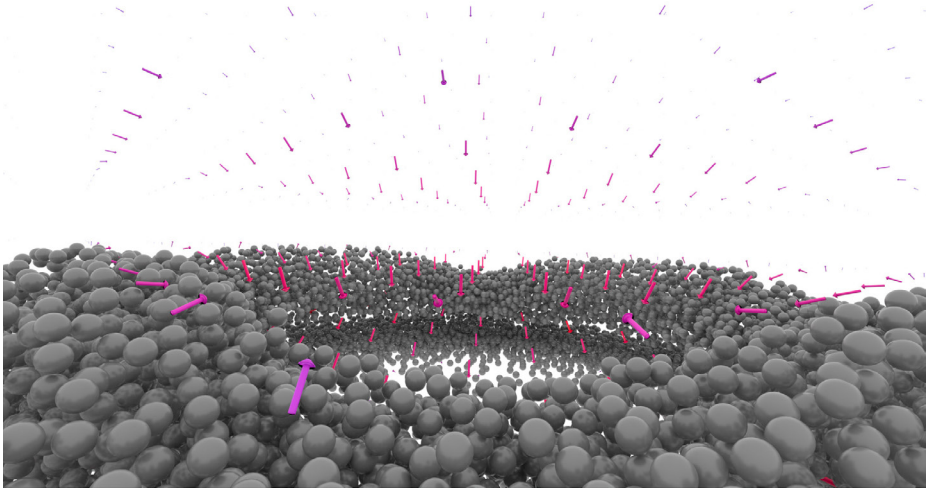


Fig. 2. Example of settling dynamics. Particles (grey spheres) are suspended in a fluid and fall due to gravity, eventually bouncing off the bottom wall of the container. Friction with the fluid causes the formation of a fluid flow (magenta arrows) which eventually disperses the particles into a circular pattern. Rendered with ZnVis [38] and republished with permission from [39]. Image credits: Paul Hohenberger.

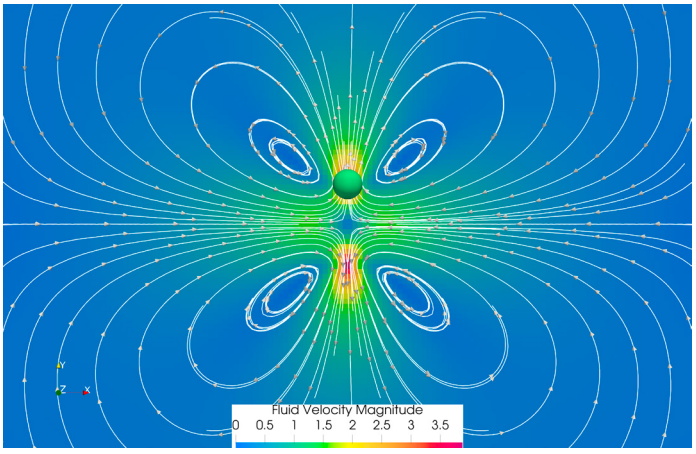


Fig. 3. Example of an active particle inducing a dipole field into the surrounding fluid. The particle induces a force on its position in the upward direction and simultaneously a force downward a small distance below its position. The resulting flow field propels the particle upward.

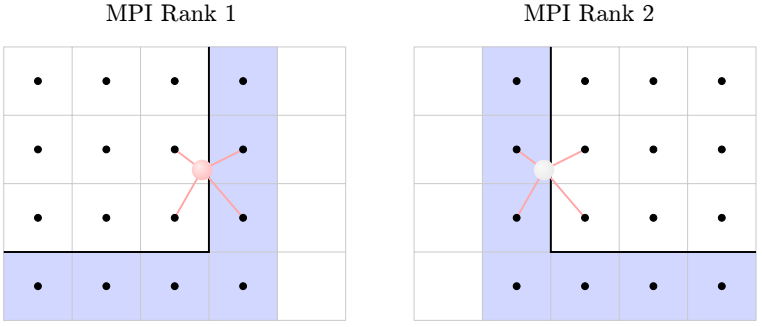


Fig. 4. Schematics of the ghost particle coupling. While the particle (red) exists on MPI rank 1, its ghost counterpart (gray) exists in the ghost layer (blue) of MPI rank 2. During the force interpolation of the ghost particles, it exerts a force on the real cells of MPI rank 2.

5. Diffusion–advection–reaction solver for electrokinetics

Aside from LB hydrodynamics coupled to coarse-grained molecular dynamics, ESPResSo also includes a finite volume based diffusion–advection–reaction solver coupled to LB hydrodynamics. In conjunction with an electrostatics solver, this is mainly used for electrokinetics. Here, charged and reactive solutes are modelled at the continuum level. The approach relies on a continuity equation, which describes the change of solute concentration in a region at a given point in time due to the solute fluxes in and out of that region [42, 21]. The fluxes occur due to Fickian diffusion of the solutes, their advection by an underlying flow of the solvent, and electrostatic forces due to the electrostatic potential created by charged solutes, or by an external electric field. When solutes move due to diffusion and electrostatic forces, they will also drag along the solvent. This is reflected by an additional friction force on the fluid, calculated from the solute fluxes. In addition, reactions can be included in the model, which can locally consume or create species concentrations. An example system with the EK algorithm is illustrated in Fig. 5.

All EK kernels are based on waLBerla's automated code generation capabilities. The electrostatics solver leverages the fast Fourier transform (FFT), which is a communication-bound algorithm [43, 44]; its scaling behaviour depends on the lattice size and favours highly composite numbers. HeFFTe [45] is used to interface ESPResSo with FFT libraries, such as FFTW for CPU and cuFFT for GPU. At present, cuFFTMp [46] is not yet supported, limiting the parallel performance of multi-GPU multi-node simulations. The finite volume solver runs concurrently with LB, sharing the infrastructure for the lattice and the fields: it can directly access the flow velocity and add forces to the fluid without data being copied. When run on a GPU, this is done entirely on the device. The simulation consists of the following steps, starting from an initial solute concentration and fluid flow field:

1. calculate charge distribution based on the concentration of solute species,
2. calculate electrostatic potential due to the solutes using an FFT-based solver for the Poisson equation,
3. calculate solute fluxes due to solute diffusion and electrostatics,
4. apply forces to LB based on solute fluxes dragging along the fluid,
5. calculate advective solute fluxes based on the flow velocity from LB,
6. integrate the diffusion–advection–reaction solver and LB by one time step.

To model thermal fluctuations in the solute dynamics, both the LB solver and EK solver need to be thermalised. These fluctuations are included as stochastic solute fluxes in the continuity equation. For EK, thermal fluctuations are particularly relevant when modelling chemical reactions with a reaction order not equal to one, e.g. $2A \rightarrow B$ [21].

6. Boundary conditions

In a coupled simulation, consistent boundary conditions have to be applied across all solvers. While the most common case in soft matter simulations are periodic boundaries, ESPResSo offers additional boundary conditions for the coarse-grained molecular dynamics, LB and EK solvers. Boundary conditions for particles are realised by means of repulsive interaction potentials, either by adding "obstacles" defined by geometric shapes, or by imposing an "energy landscape" for the simulation volume on a lattice.

The LB method employs velocity bounce-back boundary conditions, for the diffusion–advection–reaction method, boundary conditions can be applied to fix the solute density to a prescribed value or to fix fluxes to a prescribed value, acting as a source or sink for species and to control the flow of species between cells, respectively. The latter reduces to the common case of a no-flux boundary condition when set to zero. Additionally, localised catalytic reactions can be introduced, converting solute densities of different species into one another according to a defined reaction scheme. This allows for the modelling of reactive boundaries, as they are commonly found in catalytic systems.

Each of the mentioned boundary conditions can be controlled on a per-lattice-node basis, allowing for the creation of highly complex geometries with an arbitrary variety of boundary conditions [47]. Furthermore, the CGMD and LB solver supports Lees–Edwards boundary conditions, a special type of periodic boundary conditions that allows for the shearing of the system by imposing a velocity difference across periodic boundaries. Such a system is particularly useful for studying the rheological properties of soft matter systems, such as colloidal dispersions or polymer networks under shear flow.

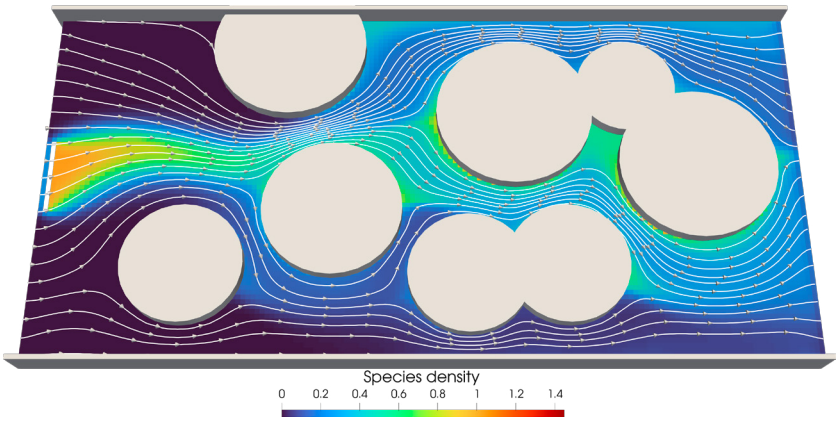


Fig. 5. Example of an advection–diffusion system inside a porous channel. On the left boundary the fluid streams into the channel with a constant velocity and on the left side in centre there is an inlet for chemical species which then diffuse and move along with the fluid. The white lines are the streamlines of the fluid and the colour shows the ion density.

7. Performance considerations and benchmarks

The shift towards accelerators as main drivers of HPC systems' performance makes it necessary to adapt scientific codes. ESPResSo 5.0 [48] introduced initial support for shared-memory parallelism and multi-GPU execution via waLBerla, Kokkos and Cabana [49, 50, 51], such that one MPI task is launched per accelerator, while the other CPUs are handled by OpenMP. At this point ESPResSo supports this programming model for lattice-based kernels, short-ranged inter-particle forces, electrostatic interactions, and the integration of the equations of motion.

As an illustration of performance, we report benchmarks on MareNostrum 5 of a pure thermalised LB fluid with 96% parallel efficiency, which is consistent with other LB software such as accLB [52] or PALABOS [53], and a dilute suspension of particles coupled to an LB fluid with 70% parallel efficiency (Fig. 6, blue curves). This performance drop is attributed to the poor scalability of the Lennard-Jones code. Offloading of the short-range force evaluation to the GPU with Cabana [51] is still an ongoing project. The thermalised LB fluid was then coupled to a thermalised diffusion–advection–reaction solver to model reactive chemical species at the continuum level, with 84% parallel efficiency due to the extra ghost communication required by EK, and then electrical charges were added to the solutes, with 38% parallel efficiency due to the cuFFT communication bottleneck in multi-node simulations (Fig. 6, orange curves). In each scenario, the largest possible lattice size that could fit in the accelerator memory was chosen.

Macroscopic properties such as LB fluid velocities and EK solute concentrations and fluxes can be written to disk using the interoperable VTK [54] file format. This data format is hardware-independent and can be read natively by ParaView [55] and the vtk Python package. The parallel write rate on the General Parallel File System (GPFS) available on MareNostrum 5 scales well with the number of accelerators (Fig. 7). The build scripts, benchmark scripts, raw data, and analysis scripts are available in a data set [56].

8. Summary and Outlook

We report on the coupling of coarse-grained molecular dynamics provided by ESPResSo to LB hydrodynamics and EK provided by waLBerla. The coupling is realised through a lean bridge library that hides the technical details of waLBerla from the molecular dynamics code. This makes the bridge potentially reusable for coupling other simulation codes to waLBerla. The coupling brings together the rich feature set of ESPResSo for coarse-grained molecular dynamics, including diverse interaction potentials, electrostatics, and chemical reactions, with state-of-the-art lattice-based methods from waLBerla and its code generation framework. Particle-fluid coupling schemes include momentum exchange, inertialess tracers for the immersed boundary method, and dipole swimmers for active particles. The automated code generation based on pystencils and lbmpy provides performance portability across CPU and GPU architectures and allows scientists to modify and enhance algorithms based on mathematical definitions rather than dealing with hardware-specific optimised code. This proved instrumental for incorporating thermal fluctuations and Lees–Edwards boundary conditions to the LB solver.

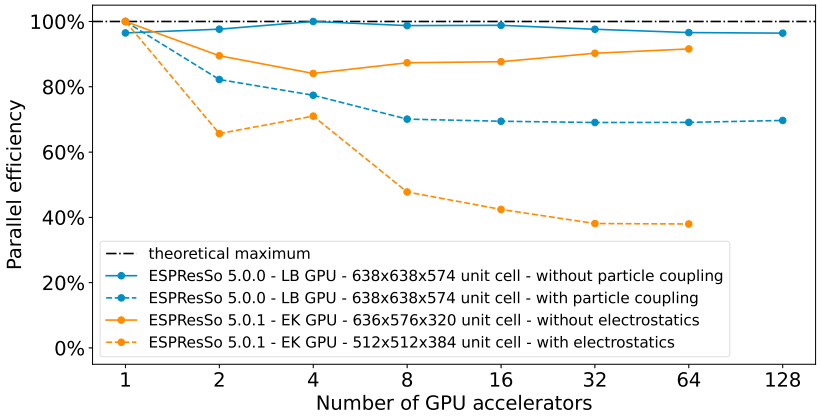


Fig. 6. Weak scaling of thermalised LB fluid simulations on MareNostrum 5 (2x 40-core Intel Xeon Platinum 8460Y+ CPUs and 4x NVIDIA H100 64 GiB HBM2 accelerators per node). The LB system with particle coupling contains 22 500 particles per CPU and ~234 million cells per GPU for a volume fraction of 0.1%, and consumes 62 GiB of VRAM and 1.15 GiB of RAM per task. EK configurations need a smaller unit cell (GPU-local lattice) since the LB fluid and EK species are stored in multiple lattices; reactions take place on catalytic surfaces modelled as solid boundaries. The unit cell was chosen to saturate accelerator memory, except for EK with electrostatics, where composite numbers are more favourable.

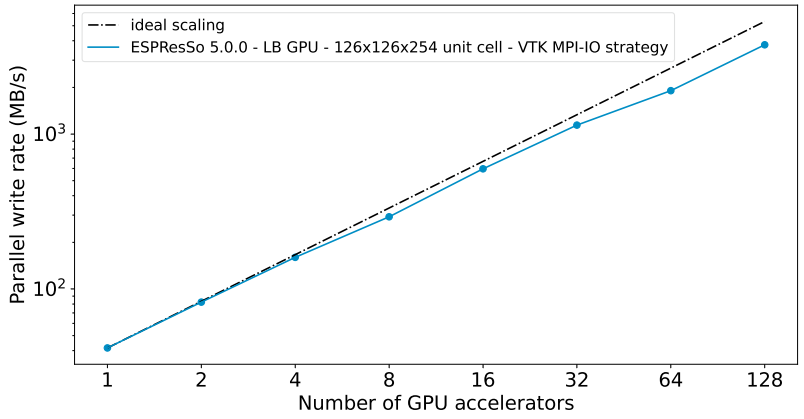


Fig. 7. Parallel write performance of a thermalised LB fluid simulation without particle coupling on MareNostrum 5. The simulated system contains ~4 million cells per GPU on a 126x126x254 unit cell. This configuration consumes 1.8 GiB of VRAM and writes 300 MiB per GPU per frame. Each benchmark run writes in total 100 GiB per node. Larger system sizes were not tried, since the amount of data to write to disk is close to the maximal MPI buffer size. The write rate is 41 MBps for 1 GPU. The parallel performance is 86% at 32 GPUs and 71% at 128 GPUs.

Acknowledgements

Funded by the European Union. This work has received funding from the EuroHPC JU and countries participating in the project under grant agreement No 101093169 (“CoE MultiXscale”) and No 956000 (“SCALABLE”), with co-funding from the Federal Ministry of Education and Research (Bundesministerium für Forschung, Technologie und Raumfahrt, BMFTR) under the funding code 16HPC095 and the Ministry of Economic Affairs and Climate Policy (Ministerie van Economische Zaken en Klimaat); the responsibility for the content of this publication lies with the authors. We acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grants No 528726435 and No 254456455. We thank Martin Bauer for his initial work on pystencils and lbmpy and help at the start of the project. We acknowledge EuroHPC JU for awarding the project ID EHPC-DEV-2025D11-078 access to MareNostrum 5 on the accelerated partition hosted at the Barcelona Supercomputing Center, Spain.

References

[1] J. T. Padding, A. A. Louis, Hydrodynamic interactions and Brownian forces in colloidal suspensions: Coarse-graining over time and length scales, Physical Review E: Statistical, Nonlinear, and Soft Matter Physics 74 (031402) (2006) 031402. doi:10.1103/PhysRevE.74.031402.

[2] S. Bharadwaj, B.-J. Niebuur, K. Nothdurft, W. Richtering, N. F. A. Van Der Vegt, C. M. Papadakis, Cononsolvency of thermoresponsive polymers: Where we are now and where we are going, Soft Matter 18 (15) (2022) 2884–2909. doi:10.1039/D2SM00146B.

[3] X. Li, J. P. Gong, Design principles for strong and tough hydrogels, Nature Reviews Materials 9 (6) (2024) 380–398. doi:10.1038/s41578-024-00672-3.

[4] C. Guindani, L. C. Da Silva, S. Cao, T. Ivanov, K. Landfester, Synthetic cells: From simple bio-inspired modules to sophisticated integrated systems, Angewandte Chemie 134 (16) (2022) e202110855. doi:10.1002/ange.202110855.

[5] E. R. Smith, P. E. Theodorakis, Multiscale simulation of fluids: Coupling molecular and continuum, Physical Chemistry Chemical Physics 26 (2) (2024) 724–744. doi:10.1039/D3CP03579D.

[6] E. G. Flekkøy, G. Wagner, J. Feder, Hybrid model for combined particle and continuum dynamics, Europhysics Letters (EPL) 52 (3) (2000) 271–276. doi:10.1209/epl/i2000-00434-8.

[7] W. Ren, Analytical and numerical study of coupled atomistic-continuum methods for fluids, Journal of Computational Physics 227 (2) (2007) 1353–1371. doi:10.1016/j.jcp.2007.09.007.

[8] E. R. Smith, P. E. Theodorakis, R. V. Craster, O. K. Matar, Moving Contact Lines: Linking Molecular Dynamics and Continuum-Scale Modeling, Langmuir 34 (42) (2018) 12501–12518. doi:10.1021/acs.langmuir.8b00466.

[9] Y. Wang, W. Zhou, A dynamic coupling scheme for fluid system by combining lattice Boltzmann method and molecular dynamics, International Journal of Modern Physics C 36 (02) (2025) 2450178. doi:10.1142/S012918312450178X.

[10] S. Kemmler, C. Rettinger, U. Rüde, P. Cuéllar, H. Köstler, Efficiency and scalability of fully-resolved fluid-particle simulations on heterogeneous CPU-GPU architectures, The International Journal of High Performance Computing Applications 39 (3) (2025) 345–363. doi:10.1177/10943420241313385.

[11] M. Deserno, C. Holm, How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines, The Journal of Chemical Physics 109 (18) (1998) 7678–7693. doi:10.1063/1.477414.

[12] M. Deserno, C. Holm, How to mesh up Ewald sums. II. An accurate error estimate for the Particle–Particle–Particle-Mesh algorithm, The Journal of Chemical Physics 109 (18) (1998) 7694–7701. doi:10.1063/1.477415.

[13] A. Arnold, C. Holm, MMM1D: A method for calculating electrostatic interactions in one-dimensional periodic geometries, The Journal of Chemical Physics 123 (12) (2005) 144103. doi:10.1063/1.2052647.

[14] S. Tyagi, M. Süzen, M. Sega, M. C. Barbosa, S. S. Kantorovich, C. Holm, An iterative, fast, linear-scaling method for computing induced charges on arbitrary dielectric boundaries, The Journal of Chemical Physics 132 (15) (2010) 154112. doi:10.1063/1.3376011.

[15] A. Arnold, K. Breitsprecher, F. Fahrenberger, S. Kesselheim, O. Lenz, C. Holm, Efficient algorithms for electrostatic interactions including dielectric contrasts, Entropy 15 (11) (2013) 4569–4588. doi:10.3390/e15114569.

[16] A. Arnold, J. de Joannis, C. Holm, Electrostatics in periodic slab geometries. I, The Journal of Chemical Physics 117 (6) (2002) 2496–2502. doi:10.1063/1.1491955.

[17] J. de Joannis, A. Arnold, C. Holm, Electrostatics in periodic slab geometries. II, The Journal of Chemical Physics 117 (6) (2002) 2503–2512. doi:10.1063/1.1491954.

[18] L. Durlofsky, J. F. Brady, G. Bossis, Dynamic simulation of hydrodynamically interacting particles, Journal of Fluid Mechanics 180 (1) (1987) 21–49. doi:10.1017/S002211208700171X.

[19] I. Tischler, A. Schlaich, C. Holm, Disentanglement of surface and confinement effects for diene metathesis in mesoporous confinement, ACS omega 9 (1) (2023) 598–606. doi:10.1021/acsomega.3c06195.

[20] J. Landsgesell, P. Hebbeker, O. Rud, R. Lunkad, P. Košovan, C. Holm, Grand-reaction method for simulations of ionization equilibria coupled to ion partitioning, Macromolecules 53 (8) (2020) 3007–3020. doi:10.1021/acs.macromol.0c00260.

[21] I. Tischler, F. Weik, R. Kaufmann, M. Kuron, R. Weeber, C. Holm, A thermalized electrokinetics model including stochastic reactions suitable for multiscale simulations of reaction–advection–diffusion systems, Journal of Computational Science 63 (2022) 101770. doi:10.1016/j.jocs.2022.101770.

[22] M. Bauer, J. Hötzer, D. Ernst, J. Hammer, M. Seiz, H. Hierl, J. Hönig, H. Köstler, G. Wellein, B. Nestler, U. Rüde, Code generation for massively parallel phase-field simulations, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Association for Computing Machinery, Denver Colorado, 2019, pp. 1–32. doi:10.1145/3295500.3356186.

[23] M. Bauer, H. Köstler, U. Rüde, Lbmpy: Automatic code generation for efficient parallel lattice Boltzmann methods, Journal of Computational Science 49 (2021) 101269. doi:10.1016/j.jocs.2020.101269.

[24] F. Hennig, M. Holzer, U. Rüde, Advanced Automatic Code Generation for Multiple Relaxation-Time Lattice Boltzmann Methods, SIAM Journal on Scientific Computing 45 (4) (2023) C233–C254. doi:10.1137/22m1531348.

[25] B. Dröge, V. H. Rusu, K. Hoste, C. van Leeuwen, A. O'Cais, T. Röblitz, EESSI: A cross-platform ready-to-use optimised scientific software stack, Software: Practice and Experience 53 (1) (2023) 176–210. doi:10.1002/spe.3075.

[26] V. Karakasis, T. Manitaras, V. H. Rusu, R. Sarmiento-Pérez, C. Bignamini, M. Kraushaar, A. Jocksch, S. Omlin, G. Peretti-Pezzi, J. P. S. C. Augusto, B. Friesen, Y. He, L. Gerhardt, B. Cook, Z.-Q. You, S. Khuvis, K. Tomko, Enabling continuous testing of HPC systems using ReFrame, in: G. Juckeland, S. Chandrasekaran (Eds.), Tools and Techniques for High Performance Computing, Springer International Publishing, Cham, Switzerland, 2020, pp. 49–68. doi:10.1007/978-3-030-44728-1_3.

[27] A. O'Cais, K. Hoste, J.-N. Grad, C. van Leeuwen, L. Peeters, S. Kamath, T. Röblitz, R. Topouchian, B. Dröge, P. S. Neves, R. Weeber, Portable test run of ESPResSo on EuroHPC systems via EESSI, Procedia Computer Science 255 (2025) 122. doi:10.1016/j.procs.2025.02.267.

[28] C. van Leeuwen, K. Hoste, L. Peeters, S. S. Kamath, Portable test suite for shared software stack, Deliverable 1.5, CoE MultiXscale (2025). doi:10.5281/zenodo.17257083.
[29] D. Beyer, P. B. Torres, S. P. Pineda, C. F. Narambuena, J.-N. Grad, P. Košovan, P. M. Blanco, pyMBE: The Python-based molecule builder for ESPResSo, The Journal of Chemical Physics 161 (2) (2024) 022502. doi:10.1063/5.0216389.
[30] D. Beyer, P. B. Torres, S. P. Pineda, S. Kurahatti, J.-N. Grad, P. Košovan, P. M. Blanco, pyMBE v1.0.0 (Oct. 2025). doi:10.5281/zenodo.17295061.
[31] S. Tovey, C. Lohrmann, T. Merkt, D. Zimmer, K. Nikolaou, S. Koppenhöfer, A. Bushmakina, J. Scheunemann, C. Holm, SwarmRL: Building the future of smart active systems, The European Physical Journal E 48 (4) (2025) 16. doi:10.1140/epje/s10189-025-00477-4.
[32] M. Bauer, S. Eibl, C. Godenschwager, N. Kohl, M. Kuron, C. Rettinger, F. Schornbaum, C. Schwarzmeier, D. Thönnes, H. Köstler, U. Rüde, waLBerla: A block-structured high-performance framework for multiphysics simulations, Computers & Mathematics with Applications 81 (2021) 478–501. doi:10.1016/j.camwa.2020.01.007.
[33] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, Am. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, A. Scopatz, SymPy: Symbolic computing in Python, PeerJ Computer Science 3 (2017) e103. doi:10.7717/peerj-cs.103.
[34] B. Dünweg, A. J. C. Ladd, Lattice Boltzmann simulations of soft matter systems, in: C. Holm, K. Kremer (Eds.), Advanced Computer Simulation Approaches for Soft Matter Sciences III, Springer, Berlin, 2009, pp. 89–166. doi:10.1007/12_2008_4.
[35] T. Krüger, Fluid–structure interaction: the immersed boundary method, in: Computer simulation study of collective phenomena in dense suspensions of red blood cells under shear, Vieweg+Teubner Verlag, Wiesbaden, 2012, pp. 37–42. doi:10.1007/978-3-8348-2376-2_6.
[36] Z.-G. Feng, E. E. Michaelides, The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems, Journal of Computational Physics 195 (2) (2004) 602–628. doi:10.1016/j.jcp.2003.10.013.
[37] J. de Graaf, H. Menke, A. J. T. M. Mathijssen, M. Fabritius, C. Holm, T. N. Shendruk, Lattice-Boltzmann hydrodynamics of anisotropic active matter, The Journal of Chemical Physics 144 (13) (2016) 134106. doi:10.1063/1.4944962.
[38] S. Tovey, ZnVis: A visualisation package for particle simulations, GitHub (2025). URL https://github.com/zincware/ZnVis
[39] R. Weeber, J.-N. Grad, Coupling ESPResSo with waLBerla, Deliverable 3.3, CoE MultiXscale (2025). doi:10.5281/zenodo.15855043.
[40] P. Ahlrichs, B. Dünweg, Simulation of a single polymer chain in solution by combining lattice Boltzmann and molecular dynamics, The Journal of Chemical Physics 111 (17) (1999) 8225–8239. doi:10.1063/1.480156.
[41] B. Dünweg, U. D. Schiller, A. J. C. Ladd, Progress in the understanding of the fluctuating lattice Boltzmann equation, Computer Physics Communications 180 (4) (2009) 605–608. doi:10.1016/j.cpc.2009.01.014.
[42] F. Capuani, I. Pagonabarraga, D. Frenkel, Discrete solution of the electrokinetic equations, The Journal of Chemical Physics 121 (2004) 973–986. doi:10.1063/1.1760739.
[43] A. Ayala, S. Tomov, M. Stoyanov, J. Dongarra, Scalability issues in FFT computation, in: V. Malyshkin (Ed.), Parallel Computing Technologies, Springer International Publishing, Cham, 2021, pp. 279–287. doi:10.1007/978-3-030-86359-3_21.
[44] M. Verma, S. Chatterjee, G. Garg, B. Sharma, N. Arya, S. Kumar, A. Saxena, M. K. Verma, Scalable multi-node fast Fourier transform on GPUs, SN Computer Science 4 (5) (2023) 625. doi:10.1007/s42979-023-02109-0.
[45] A. Ayala, S. Tomov, A. Haidar, J. Dongarra, *heFFTe*: Highly efficient FFT for exascale, in: V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, J. Teixeira (Eds.), Computational Science – ICCS 2020, Springer International Publishing, Cham, Switzerland, 2020, pp. 262–275. doi:10.1007/978-3-030-50371-0_19.
[46] NVIDIA, cuFFTMp documentation, NVIDIA (2026). URL https://docs.nvidia.com/cuda/cufftmp/
[47] C. Lohrmann, C. Holm, A novel model for biofilm initiation in porous media flow, Soft Matter 19 (36) (2023) 6920–6928. doi:10.1039/D3SM00575E.
[48] J.-N. Grad, F. Weik, A. Reinauer, H. Kobayashi, I. Tischler, P. M. Blanco, D. Mostarac, S. Bindgen, D. Beyer, J. Hoßbach, M. Kuron, P. Hohenberger, N. Müller, R. Frenner, Y. Gandhi, M. E. Brito, S. Tovey, R. Weeber, ESPResSo v5.0.1 (May 2026). doi:10.5281/zenodo.20450775.
[49] H. C. Edwards, C. R. Trott, D. Sunderland, Kokkos: Enabling manycore performance portability through polymorphic memory access patterns, Journal of Parallel and Distributed Computing 74 (12) (2014) 3202–3216. doi:10.1016/j.jpdc.2014.07.003.
[50] C. R. Trott, D. Lebrun-Grandie, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, J. Wilke, Kokkos 3: Programming model extensions for the exascale era, IEEE Transactions on Parallel and Distributed Systems 33 (4) (2022) 805–817. doi:10.1109/tpds.2021.3097283.
[51] S. Slattery, S. T. Reeve, C. Junghans, D. Lebrun-Grandié, R. Bird, G. Chen, S. Fogerty, Y. Qiu, S. Schulz, A. Scheinberg, A. Isner, K. Chong, S. Moore, T. Germann, J. Belak, S. Mniszewski, Cabana: A performance portable library for particle-based simulations, Journal of Open Source Software 7 (72) (2022) 4115. doi:10.21105/joss.04115.
[52] M. Lauricella, A. Mukherjee, L. Brandt, S. Succi, D. Izbassarov, A. Montessori, accLB: A high-performance lattice Boltzmann code for multiphase turbulence on multi-gpu architectures, Procedia Computer Science 267 (2025) 40–51. doi:10.1016/j.procs.2025.08.231.
[53] J. Latt, C. Coreixas, Multi-GPU acceleration of PALABOS fluid solver using C++ standard parallelism (2026). doi:10.48550/arXiv.2506.09242.
[54] W. Schroeder, K. Martin, B. Lorensen, The Visualization Toolkit. An Object-Oriented Approach to 3D Graphics, 4th Edition, Kitware, 2006.
[55] U. Ayachit, The ParaView Guide. A Parallel Visualization Application, Kitware, Incorporated, Clifton Park, New York, USA, 2015.
[56] J.-N. Grad, I. Tischler, A. Reinauer, H. Kobayashi, M. Kuron, F. Hennig, M. Holzer, P. M. Santos Neves, S. S. Kamath, C. Holm, R. Weeber, Replication data for: Coupling ESPResSo and waLBerla for scalable soft matter simulations with hydrodynamic interactions, Dataset V2, DaRUS (Mar. 2026). doi:10.18419/DARUS-5827.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 95–105

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

Mixed-Precision GPU Acceleration of RTE-RRTMGP-CPP: Towards Radiative Transfer Simulations on Exascale Supercomputers

Stijn Heldens[a,*], Alessio Sclocco[a], Gijs van den Oord[a], Ben van Werkhoven[b], Chiel van Heerwaarden[c], Erwan Raffin[d], Xavier Yepes-Arbós[e]

[a] *Netherlands eScience Center, Amsterdam, the Netherlands*
[b] *Leiden University, Leiden, the Netherlands*
[c] *Wageningen University & Research, Wageningen, the Netherlands*
[d] *Center for Excellence in Performance Programming (CEPP), Bull, Rennes, France*
[e] *Barcelona Supercomputing Center, Barcelona, Spain*

Abstract

Radiative transfer is a fundamental component of weather and climate modeling, often accounting for a significant portion of total simulation time in Earth System Models (ESMs). RTE+RRTMGP-CPP is a C++ frontend as well as a CUDA-based implementation of the Radiative Transfer for Energetics (RTE) solver and the Rapid Radiative Transfer Model (RRTMGP), both widely used to compute longwave and shortwave radiative transfer in vertical columns of atmospheric models. However, despite existing GPU acceleration, high computational intensity remains a bottleneck for scalability on modern supercomputers.

In this work, we present a scalable, mixed-precision implementation of RTE-RRTMGP-CPP. Driven by recent advances in machine learning, GPU manufacturers have integrated specialized low-precision hardware units offering significantly higher throughput and energy efficiency than traditional 64-bit floating-point numbers. By selectively integrating these low-precision data types and arithmetic in performance-critical GPU kernels of RTE-RRTMGP-CPP, we improve computational efficiency while maintaining scientific accuracy within an acceptable tolerance. Using automatic precision tuning, we are able to find multiple configurations of the application that provide Pareto-optimal trade-offs between numerical accuracy and execution speed.

Our results show that mixed-precision tuning accelerates the radiative transfer solver, achieving speedups of up to 2.1× on Leonardo (NVIDIA A100) and 2.2× on LUMI (AMD MI250X) compared to double precision and carefully selected configurations offer attractive performance-accuracy trade-offs compared to uniform single precision. Overall, this work demonstrates how to exploit the low-precision capabilities of modern GPUs and prepare weather and climate models for upcoming exascale systems.

Keywords: mixed-precision computing, GPU acceleration, automatic tuning, high-performance computing, radiative transfer

1. Introduction

Simulating radiative transfer is a crucial component of many weather and climate models within modern Earth System Models (ESM). Radiative transfer calculations are responsible for simulating the interactions of solar and terrestrial radiation with atmospheric gases, clouds, and aerosols. These processes determine important atmospheric properties, such as heating rates and radiative fluxes, which directly influence atmospheric dynamics and long-term

*Corresponding author
Email address: s.heldens@esciencecenter.nl (Stijn Heldens)

10.1016/j.procs.2026.08.023

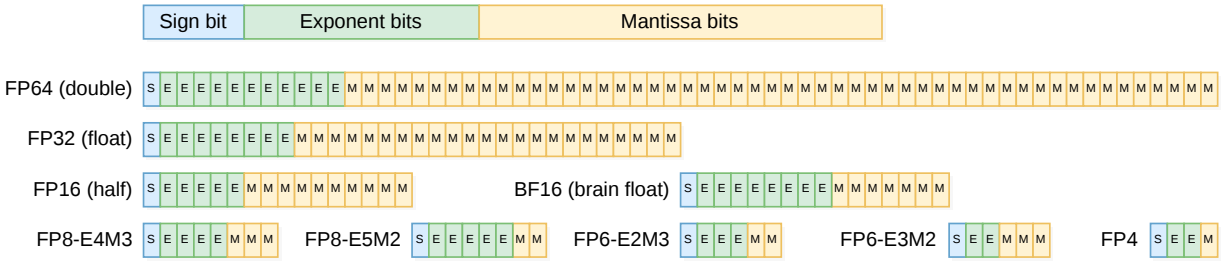


Figure 1. Different floating-point formats supported on modern NVIDIA GPUs [5].

climate evolution. Due to the complexity of the underlying physical models and the large number of atmospheric columns that must be evaluated at every simulation timestep, radiative transfer computations often account for a substantial fraction of the total execution time in weather and climate simulations.

Several efficient radiative transfer solvers have been developed for use in large-scale atmospheric models. Notable among these are the *Radiative Transfer for Energetics* (RTE) solver and the *Rapid Radiative Transfer Model for GCM Applications* (RRTMGP) that provide accurate and computationally efficient representations of longwave and shortwave radiative processes. While the original *RTE+RRTMGP* code [1] is implemented in Fortran, *RTE+RRTMGP-CPP* [2] is an alternative C++/CUDA implementation designed to exploit the massive parallelism offered by GPUs. However, even with GPU acceleration and accompanying techniques, such as temporal sub-sampling or spatial coarsening [3] and using machine learning to replace (parts of) the scheme with an emulator [4], the computational cost of radiative transfer remains high as resolution increases.

Simultaneously, the architecture of large-scale high-performance computing (HPC) systems is undergoing rapid change. GPUs are now ubiquitous in supercomputers and, driven largely by the demands of machine learning workloads, modern GPUs are increasingly equipped with specialized hardware units optimized for low-precision arithmetic (see Figure 1). The use of reduced-precision data formats offers several advantages over the traditional double-precision floating-point format, including higher computational throughput, improved energy efficiency, and a reduced memory footprint. These developments have allowed certain applications, most notably in artificial intelligence, to reach performance levels previously considered unattainable.

These advancements also hold potential for the domain of scientific computing [6]. However, while lower-precision arithmetic can significantly improve performance, it may introduce numerical errors that propagate through the simulation and potentially affect scientific results. This has motivated the field of *mixed-precision computing* [7, 8, 9], where the goal is to use different levels of precision for different computations and variables within the code. For complex physical models like radiative transfer, identifying which parts of the computation can safely operate in reduced precision is nontrivial. Manually exploring the large design space of possible precision configurations is often impractical, particularly for several computational kernels arranged in a pipeline

In this work, we investigate the use of mixed-precision GPU computing to accelerate radiative transfer calculations in RTE-RRTMGP-CPP. We achieve this by identifying the most compute-intensive kernels and making their data types configurable using a C++ library called *Kernel Float* [10]. Next, we use an automatic tuning framework, called *Kernel Tuner* [11, 12], to explore the search space and identify configurations that offer a favorable trade-off between numerical accuracy and computational performance.

Our results show that mixed-precision tuning can accelerate the radiative transfer solver, achieving speedups over a double-precision implementation of up to 2.1× on NVIDIA and 2.2× on AMD GPUs. Additionally, carefully selected mixed-precision configurations provide attractive performance-accuracy trade-offs compared to uniform single-precision execution, either by delivering better performance with similar accuracy or better accuracy with similar performance. Scientific validation shows that a moderate reduction in precision preserves physically relevant quantities such as heating-rate profiles, but highly aggressive reductions lead to errors that are no longer scientifically meaningful. Overall, these results contribute to the ongoing effort to prepare weather and climate models for upcoming exascale supercomputing systems, including the next generation of European supercomputers.

This paper is organized as follows: Section 2 provides background, Section 3 describes our implementation, Section 4 evaluates our work, Section 5 discusses related work, and Section 6 presents directions for future work.

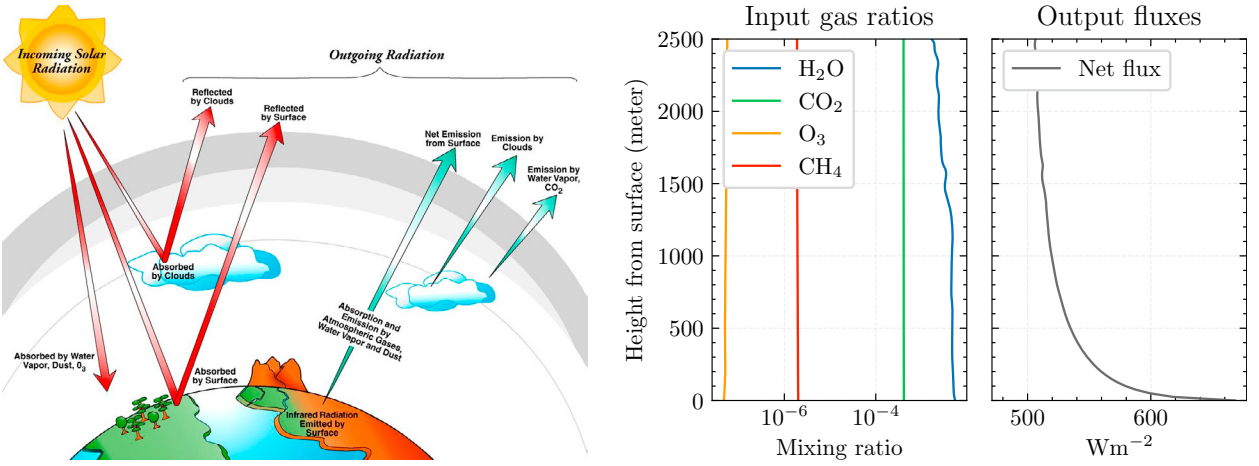


Figure 2. Example of atmospheric radiation. Left: visual representation (Image courtesy of the U.S. Department of Energy Atmospheric Radiation Measurement (ARM) user facility). Right: input gases and resulting fluxes for one atmospheric column in RTE-RRTMGP-CPP.

2. Background

In this section, we provide background on the ESiWACE3 services (Section 2.1), the RTE-RRTMGP-CPP model (Section 2.2), mixed-precision GPU programming (Section 2.3), and automatic performance tuning (Section 2.4).

2.1. ESiWACE3 Services

ESiWACE is the Centre of Excellence in Simulation of Weather and Climate in Europe, currently in its third phase (2023-2026). The main objective of this collaborative project is to advance weather and climate modeling using cutting-edge supercomputing. ESiWACE drives innovation to address global challenges, support policymakers, and contribute to a more sustainable future for our planet. One of ESiWACE's activities is providing HPC services to enhance weather and climate applications through collaborations with external model developers. Support is provided either as six person-months over one year or two person-months over six months. During this period, ESiWACE Research Software Engineers (RSE) work closely with model developers to provide technical guidance and knowledge transfer, ensuring lasting and maintainable improvements to the main code base.

The project offers two services: (i) model optimization, acceleration, and HPC exploitation, and (ii) support for the European ESM community in adopting specialized tools for next-generation (pre-)exascale systems. This includes mixed-precision optimization with AutoRPE for CPUs or Kernel Tuner for GPUs, as well as support for deployment with Spack, containerization, and Automatic Performance Profiling (APP). The work presented in this paper resulted from a collaboration under service (ii), focusing on the application of Kernel Tuner and Kernel Float to the RTE-RRTMGP-CPP radiation model.

2.2. RTE-RRTMGP-CPP

RTE-RRTMGP-CPP, which was derived from RTE+RRTMGP [1], was developed with applications to cloud-resolving simulations in mind, as wel all extensions toward 3D radiation using Monte Carlo ray tracing [13]. In short, *RRTMGP* is the optical solver that converts concentrations of atmospheric gases into optical properties (optical thickness, single-scattering albedo, and asymmetry factor), whereas *RTE* computes radiation fluxes (see Figure 2) for a given source (solar for shortwave and thermal emission for longwave). Broadband fluxes are obtained by aggregating independent computations for each quadrature point. A typical configuration contains 112 points for shortwave and 256 for longwave, indicating the high computational cost associated with radiative transfer calculations. Optical thicknesses can vary by orders of magnitude within the domain, and flux computations mainly involve reduction operations that add small numbers to large fluxes. Thus, the implementation of mixed-precision algorithms must be done with care so as not to introduce errors that affect the scientific conclusions.

```cpp
#include "kernel_float.h"

__global__ void axpy(kernel_float::vec_ptr<TYPE_Y, VECTOR_SIZE> y,
                     kernel_float::vec_ptr<const TYPE_X, VECTOR_SIZE> x,
                     TYPE_A a, const int n) {
    int i = BLOCK_SIZE_X * blockIdx.x + threadIdx.x;
    if (i < n/VECTOR_SIZE) y[i] += a * x[i];
}
```

Listing 1. Implementation of an AXPY kernel using Kernel Float [10] for CUDA and HIP.

2.3. Mixed-Precision GPU Programming

Modern GPUs, increasingly driven by the demands of machine learning, nowadays support several low-precision floating-point formats (see Figure 1). These formats can improve computational throughput, increase energy efficiency, and reduce memory usage. For example, modern GPUs execute 16-bit floating-point operations at twice the rate of 32-bit operations while using half the memory bandwidth and energy [9]. However, many traditional high-performance computing (HPC) applications cannot rely solely on low precision without having an unacceptable loss of numerical accuracy. This has motivated *mixed-precision* computing [7, 8, 9], which combines variables of different precisions to accelerate only those computations that do not require high precision.

However, writing portable mixed-precision GPU code directly in low-level languages, such as CUDA or HIP, is challenging. Support for floating-point formats differs across GPU architectures, and hardware vendors use different naming conventions and hardware intrinsics, especially for low-precision types. *Kernel Float* [10] is a recently developed solution for GPU kernel programming that aims to simplify the development of mixed-precision GPU kernels. The library provides a generic vector type that packs multiple floating-point scalars into a single vector and automatically uses hardware-accelerated instructions when available. This enables portable mixed-precision GPU code and allows switching between data types and GPU vendors (AMD and NVIDIA) with no code changes.

Listing 1 shows an example of a mixed-precision GPU kernel that performs the operation $Y := Y + aX$, where X, Y are vectors and a is a scalar. This code is essentially a blueprint that can use different types and vector lengths. Specifically, the data types `TYPE_Y`, `TYPE_X`, and `TYPE_A` can be chosen freely to select the formats for the variables x, y, and a. Additionally, the constants `BLOCK_SIZE_X` and `VECTOR_SIZE` can be set to specify the thread block dimensions (i.e., number of threads per block) and vector width (i.e., number of elements per thread), respectively.

2.4. Automatic Performance Tuning of GPU Code

Writing GPU kernel code is challenging as it requires programmers to navigate a large design space of implementation choices that impact performance while not affecting the correctness of the code [14]. Examples include thread block dimensions (i.e., threads per block), vector size (i.e., elements per thread), loop unroll factors, use of shared memory, and number of registers per thread. Automatic performance tuning, or *auto-tuning*, enables developers to systematically explore this search space and identify the configuration of parameters that maximizes performance.

Kernel Tuner [12] is an open-source auto-tuner for GPU code that provides a high-level Python interface where users can specify their kernel code, tunable parameters, input data, and expected output data. The tuner handles compilation of the code, measurement of performance, and the search through the space for the optimal parameters.

A recent addition to Kernel Tuner is an accuracy-aware extension [11] that enables floating-point precision parameters (i.e., data types) to be tuned alongside conventional architectural parameters (e.g., block size and vector size). This allows Kernel Tuner to be used with GPU kernels that contain both types of parameters, such as shown in Listing 1. For each configuration, the tuner measures both throughput and numerical error, steering the search toward a balance between computational performance and accuracy. Results show that this unified approach to tuning accuracy and performance outperforms using separate performance and precision tuners [11].

3. Implementation

In this section, we describe how we implemented the mixed-precision GPU version of RTE+RRTMGP-CPP [2] through the following steps: profiling the original application (Section 3.1), adapting the GPU kernel code (Section 3.2), tuning the precision and performance of the individual kernels (Section 3.3), and integrating the tuned kernels back into the application (Section 3.4). Finally, we discuss porting and tuning for AMD GPUs (Section 3.5).

Parameter	Description
FLOAT	Generic floating-point type used throughout the code base. Primary numerical precision type for computations unless a more specific type is required.
FLOAT_COMPUTE_LW/SW/ MAJORG/MINORG	Floating-point type used for internal computations. A separate type is defined for each of the four kernels, allowing precision to be tuned independently.
FLOAT_FLUX	Spectral band fluxes (shortwave and longwave). These represent radiative energy transport and are sensitive to precision choices.
FLOAT_TEMPERATURE	Floating-point type used for temperature fields.
FLOAT_PRESSURE	Floating-point type used for pressure fields.
FLOAT_TAU	Optical depth values, which can vary over several orders of magnitude.
FLOAT_SURFACE	Surface-related radiative quantities such as surface fluxes or albedo-dependent terms. Can use higher precision since each atmospheric column has only a single value.
FLOAT_SOURCE	Source function values, including thermal emission and scattering source terms.
FLOAT_COL_GAS	Column-integrated gas amount.
FLOAT_COL_MIX	Column mixing ratios (gas fraction per layer).
FLOAT_FMINOR/FMAJOR	Absorption coefficients for minor and major gases.
FLOAT_KMINOR/KMAJOR	k-distribution coefficients for minor and major gases.

Table 1. Floating-point type definitions introduced into RTE-RRTMGP-CPP.

3.1. Application Profiling

The first step was to profile the original application using the NVIDIA Nsight Systems profiler. This was done using double precision (64-bit) on a dataset with grid dimensions of 384×384×384 on an NVIDIA A100 GPU. For this particular use case, both the shortwave and longwave radiation calculations were enabled. The performance profile showed that the application called 27 distinct GPU kernels. However, the runtime is highly imbalanced, with a small number of kernels dominating the total execution time. For this work, we focus on integrating mixed precision into four kernels, which together account for 68.8% of the total simulation time: two kernels for solving longwave and shortwave radiation and two kernels for computing gas optical depths.

- **LW solver** (`lw_solver_noscat`): Longwave radiative transfer solver without scattering (31.4% of time).
- **SW solver** (`sw_solver`): Shortwave radiative transfer solver including scattering (18.7%).
- **Major gases** (`gas_optical_depths_major`): Optical depths for major gases (e.g., H_2O, CO_2) (9.8%).
- **Minor gases** (`gas_optical_depths_minor`): Optical depth calculation for minor gases (trace species) (8.9%).

3.2. Kernel Extraction

The next step was to extract each of the four kernels from the original application and modify them to support mixed precision. To achieve this, we integrated Kernel Float [10] into the GPU kernels. With Kernel Float, each floating-point type becomes a *tunable parameter*, allowing an auto-tuner to explore the effect of using different precision levels for different data types within each kernel.

Overall, we introduced the data types shown in Table 1. Not all data types are relevant for every kernel: some are used only by specific kernels, while others are shared across multiple kernels. The data types also serve different purposes: some store physical fields (e.g., `FLOAT_FLUX`, `FLOAT_PRESSURE`), some represent lookup tables (e.g., `FLOAT_KMINOR`), and others are used for intermediate variables within the kernel code (e.g., `FLOAT_COMPUTE_LW`). The type `FLOAT` is special, as it is used for computations in all other kernels that are not tuned in this work.

In addition to the application-wide precision parameters, we also introduced several performance-related tuning parameters for each kernel (see Table 2). These parameters determine performance aspects that can significantly influence runtime performance but do not affect numerical accuracy, such as block size and loop unrolling. During tuning, these kernel-specific performance parameters are explored jointly with the application-level precision parameters in a single search. Note that not all performance parameters apply to every kernel, for example, some kernels do not have a z-axis, do not contain loops to be unrolled, or do not use shared memory.

Parameter	Description
BLOCK_SIZE_X	Thread block dimension along the x-axis.
BLOCK_SIZE_Y	Thread block dimension along the y-axis.
BLOCK_SIZE_Z	Thread block dimension along the z-axis.
VECTOR_SIZE	Number of data elements processed per thread (i.e., atmospheric columns per thread).
LOOP_UNROLL_FACTOR	Unrolling factor applied to loops within the kernel to reduce loop control overhead.
USE_SMEM	Whether to use shared memory.

Table 2. Optimization parameters introduced for each kernel into RTE-RRTMGP-CPP.

3.3. Precision and Performance Tuning Mixed-Precision Kernels

Next, we tune the parameters of the four kernels using *Kernel Tuner*. Each kernel is tuned individually; the integration of the tuned kernels back into the application is discussed in the next section.

In order to measure the accuracy loss introduced by lower-precision data types, we obtain reference data by executing the original application in double precision. When executing the application, the first time each kernel is called, we store the input data to disk immediately before execution and the output data immediately afterward. To quantify the accuracy loss of mixed-precision configurations during tuning, we measure the *normalized root-mean-square error* (NRMSE) between this reference output y_i and the mixed-precision output y'_i:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum \left(y_i - y'_i\right)^2}, \quad \text{N} = \sqrt{\frac{1}{n}\sum y_i^2}, \quad \text{NRMSE} = \frac{\text{RMSE}}{\text{N}} \tag{1}$$

The NRMSE provides a rough indication of the number of decimal digits that are correct in the computed results. For example, an NMRSE of 10^{-k} suggests that approximately k decimal places of the output values are accurate.

For tuning each kernel, we consider four data type options (see Figure 1): FP64 (double precision), FP32 (single precision), FP16 (half precision), and BF16 (*bfloat16*). FP16 and BF16 have the same bit width, but FP16 has more significant digits, whereas BF16 has a larger exponent range and is less likely to overflow. We do not consider smaller formats, as their limited precision makes them impractical. Parameter configurations producing invalid values (NaN or infinity) are discarded. For tuning, we let Kernel Tuner search for the parameter configuration that maximizes performance while keeping the numerical error below a predefined threshold τ on the NRMSE (see Section 4.2).

3.4. Integration of Mixed-Precision Kernels into the Application

The final step was to integrate the tuned kernels back into the application. After the tuning process, we generate a C++ header file containing the selected configuration for each kernel. This includes both the chosen application-wide types (e.g., FLOAT_TAU, FLOAT_PRESSURE) and the selected kernel-specific performance parameters (e.g., BLOCK_SIZE_X, VECTOR_SIZE). The generated header file is used to compile the full RTE+RRTMGP-CPP application.

When generating the header file, special care must be be taken for data types shared across multiple kernels. For example, both the LW and SW solver kernels read surface source values stored with the type FLOAT_SURFACE. Similarly, the gas optical depth kernels compute optical depth values (FLOAT_TAU), which are subsequently used by the LW and SW solver kernels. For these types, we adopt a simple rule: if different kernels require different precisions, we select the higher-precision type. Increasing the precision of a data type cannot worsen numerical accuracy and therefore provides a safe solution. For example, if one kernel prefers FP16 and another FP32, we select FP32.

Finally, there is the choice of the FLOAT type, which is the default floating-point type used for kernels not tuned in this work. Because modifying this type affects the entire code base, we select it manually (see Section 4.2).

3.5. Porting to AMD

RTE+RRTMGP-CPP is implemented using CUDA. However, one of our target platforms, LUMI, utilizes AMD GPUs that cannot natively execute CUDA code. Therefore, we rewrote the CUDA-specific sections using KMM [15], a framework for parallel dataflow execution and efficient memory management in multi-GPU systems. Internally, KMM has a compile-time compatibility layer that translate CUDA calls to HIP [16], thereby achieving execution on AMD GPUs. Although KMM enables code portability between NVIDIA and AMD GPUs, optimal tuning parameters are generally not portable [17]. Therefore, we tuned the implementation for the two GPUs separately.

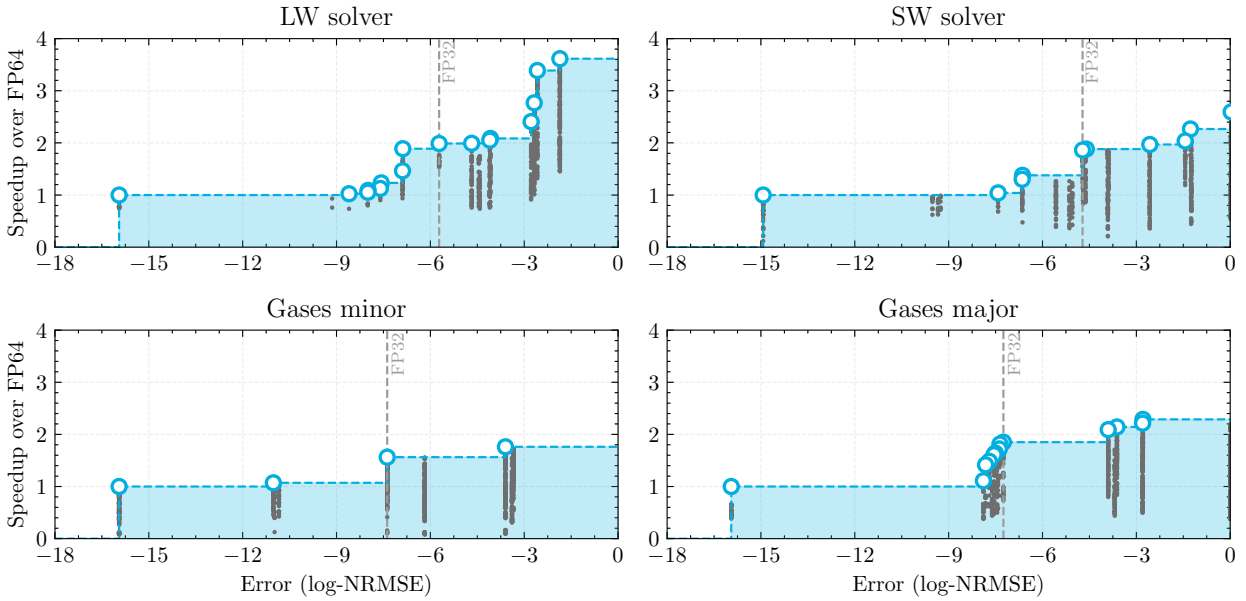


Figure 3. Tuning results on NVIDIA A100 for the individual kernels. Each graph shows 1000 random configurations (black dots), the Pareto-optimal configurations (blue circles), and the Pareto front (blue dashed line). The dash vertical line indicates the error obtained with full FP32.

4. Evaluation

This section presents the results for individual kernels (Section 4.1), the performance of the application as a whole (Section 4.2), and a scientific validation (Section 4.3). Experiments were done on two systems: *Leonardo* and *LUMI*.

Leonardo is a petascale supercomputer located at the CINECA data center in Italy and ranked 10th in the most recent Top500 (November 2025). The system is based on a BullSequana XH2000 architecture. Its Booster module consists of 3,456 nodes of custom BullSequana X2135 "Da Vinci" blade servers. Each node contains a single-socket 32-core Intel Xeon Platinum 8358 CPU, 512 GB of system memory, and four custom **NVIDIA Ampere A100** GPUs, each equipped with 64 GB of HBM2e memory. We used CUDA 12 and GCC 12.2.0.

LUMI is a pre-exascale supercomputer hosted by CSC in Finland. The system is based on HPE Cray EX architecture and ranked 9th in the Top500 (Nov 2025). The GPU partition consists of nodes equipped with a single AMD EPYC processor and four **AMD Instinct MI250X** GPUs, each GPU integrating two compute dies with 64 GB of HBM2e memory in total. We used ROCM 6.3.4 and AMD clang 18.0.0.

4.1. Kernel-Level Results

First, we consider the tuning results for each of the four kernels individually. Figure 3 shows, for each kernel on an NVIDIA A100 GPU, 1000 randomly chosen configurations, with the numerical error (logarithmic NRMSE) on the horizontal axis and the computational performance on the vertical axis. Performance is measured as the speedup relative to the fastest configuration in which all data types use FP64 (double precision). Each graph also includes the Pareto-optimal front: configurations for which no other configuration achieves both better performance and better accuracy. Vertical clusters of configurations appear in the graphs because different parameter configurations can produce identical numerical errors. For example, configurations that have the same data type parameters but different thread block dimensions. Results for the AMD MI250X look similar.

These graphs illustrate the trade-off between performance and numerical accuracy that arises from mixed-precision computing: higher performance can generally only be achieved at the cost of larger numerical errors. For example, for the LW solver, a speedup of up to 3.6× over the double-precision baseline can be achieved. This is significant considering that 31.4% of the total simulation time is spent on this kernel (Section 3.1). However, this configuration

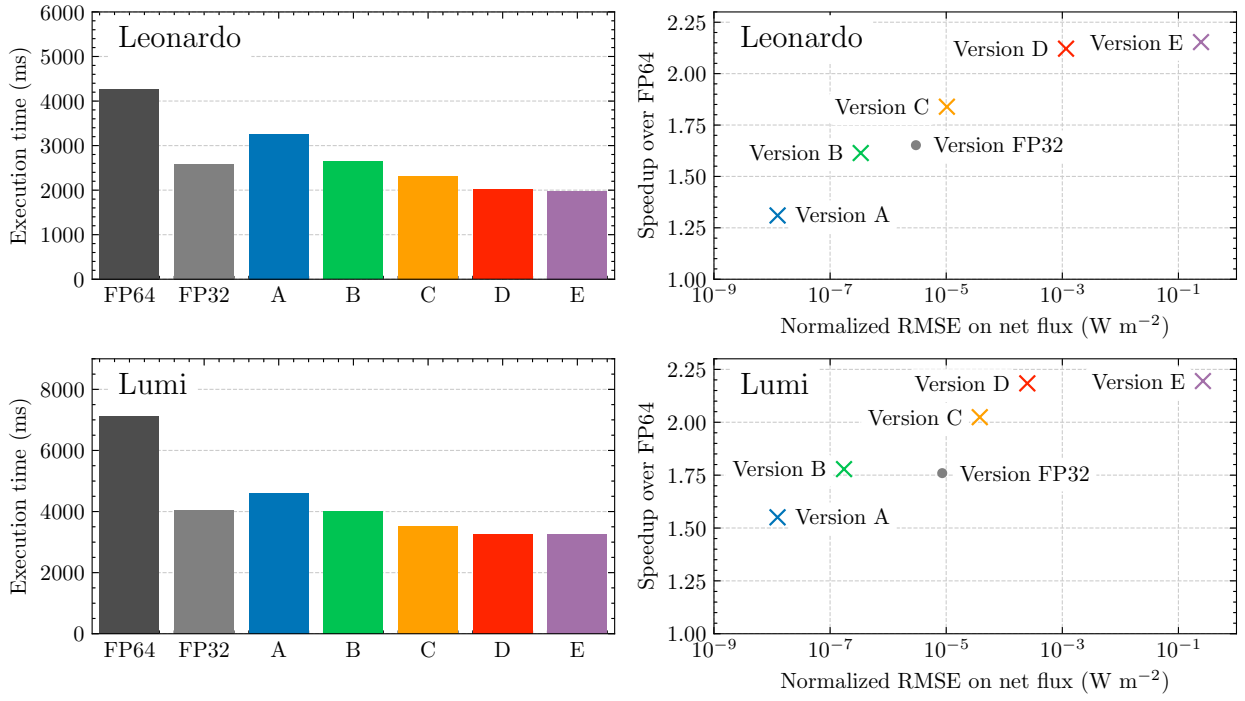


Figure 4. Execution time (left) and numerical error (right) for the mixed-precision versions for Leonardo (top) and LUMI (bottom).

has an NRMSE of $1.4 \cdot 10^{-2}$, which may be unacceptably large for certain applications. Alternatively, a configuration with a smaller error of $1.6 \cdot 10^{-3}$ also exist, although this reduces the speedup to 2.4× over double precision. The other kernels also show performance improvements, although the achievable speedups are smaller than for the LW solver. The maximum speedups are 2.6× for the SW solver, 1.8× for the minor gases, and 2.3× for the major gases kernel.

4.2. Application-Level Results

Now we consider the performance of the full application. First, we tuned the application (See Section 3) to use only FP64 (double precision) and only FP32 (single precision). Next, we tuned the following five versions:

- Version A: Error threshold of $\tau = 10^{-7}$ and `FLOAT=FP64`.
- Version B: Error threshold of $\tau = 10^{-5}$ and `FLOAT=FP64`.
- Version C: Error threshold of $\tau = 10^{-3}$ and `FLOAT=FP32`.
- Version D: Error threshold of $\tau = 10^{-2}$ and `FLOAT=FP32`.
- Version E: Error threshold of $\tau = 10^{-1}$ and `FLOAT=FP32`.

For each version, we executed the application on a dataset with grid dimensions of 384×384×256 at a 100×100×25 m resolution. Figure 4 shows the execution time and numerical error measured for each version. The execution time is measured as the time taken by the solver itself on the GPU and does not include aspects such as initialization of the GPU, reading the input data from disk, or writing the output data to disk. The graph shows how the execution time decreases from around 4.3 seconds (Leonardo) and 7.1 seconds (LUMI) for double precision (FP64) to just 2.0 seconds (Leonardo) and 3.2 seconds (LUMI), corresponding to speedups of 2.1× and 2.2×. Note that all mixed-precision versions are faster than FP64, and versions C, D, and E are also faster than FP32.

To measure the numerical error introduced by low precision, we consider the normalized RMSE over the *net flux* for each grid cell as computed by RTE-RRTMGP-CPP. The net flux is the sum of the shortwave and longwave flux contributions across all spectral points at that location. Figure 4 plots the NRMSE (horizontal axis) versus the speedup over FP64 (vertical axis), showing the trade-off between computational performance and numerical accuracy across the different mixed-precision configurations. Note that FP64 is absent, as its speedup is one and its error is zero.

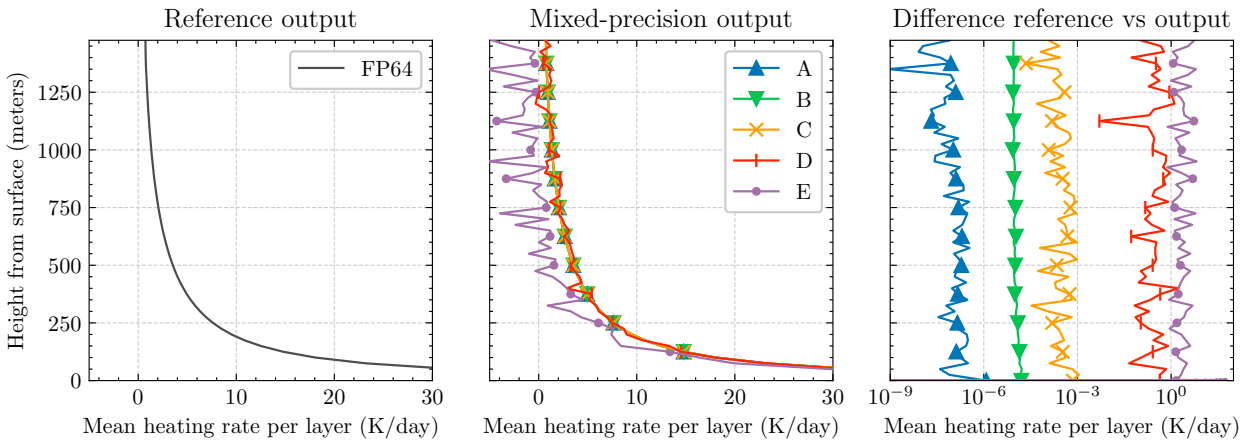


Figure 5. Heating rate per height layer for the mixed-precision versions. Left: The average heating rate at each layer for the FP64 version. Middle: Mixed-precision versions for Leonardo. Right: The absolute mean error at each layer compared to FP64 (Note the logarithmic scale).

The first observation is that the measured NRMSE of the net flux differs from the error threshold set for the individual kernels. For example, version A was tuned for an error threshold of 10^{-7}, but achieves a lower error of $1.2 \cdot 10^{-8}$. This is because the individual kernels produce intermediate quantities, whereas the net flux is obtained by summing all flux contributions, allowing numerical errors to partially cancel out. As a result, the error measured in the net flux does not necessarily correspond to the error measured in the intermediate quantities produced by the individual kernels, but instead reflects the overall impact on the radiative flux computed by the model.

Interestingly, on both Leonardo and LUMI, version B has roughly the same execution time as FP32 (+2.3% Leonardo, -1.1% LUMI), while being an order of magnitude more accurate. On the other hand, version C is faster than FP32 (-10.1% Leonardo, -13.0% LUMI), while being only slightly less accurate. Further investigation revealed that version B has nearly all data types as FP32, except `FLOAT_COMPUTE_SW` which is FP64, meaning that the largest accuracy loss occurs in computations of the SW solver. Version C, on the other hand, has all data types as FP32, except for `FLOAT_SOURCE` and `FLOAT_TAU` which are FP16, implying that reducing the precision of source functions and optical depths yields a notable performance gain at only a small accuracy cost. This shows that precision-aware auto-tuning can uncover attractive performance-accuracy trade-offs compared to uniform FP32 precision .

Finally, we can also see that versions D and E have almost the same execution time (2.12 versus 2.09 seconds on Leonardo), while version E is more than two orders of magnitude less accurate (216.2×). This shows that aggressively reducing precision does not necessarily lead to meaningful performance improvements, and that beyond a certain point the loss in numerical accuracy outweighs the marginal gain in computational throughput.

4.3. Scientific Validation

While the mean relative error on the net flux is a good metric to quantify the overall numerical impact of reduced precision on the radiative fluxes, it does indicate how these errors would propagate to physically relevant quantities in the atmospheric model. To quantify this, we can calculate the *heating rate* at each grid point in the atmosphere. The heating rate gives an indication of the rate at which radiation warms or cools the atmosphere and is therefore an important quantity used by weather and climate models to update the atmospheric temperature. The heating rate H is defined as the vertical divergence of the net radiative flux:

$$H = -\frac{g}{c_p}\frac{\partial F_{\text{net}}}{\partial p} \tag{2}$$

where F_{net} is the net radiative flux, g gravitational acceleration, c_p the heat capacity of air at constant pressure, and p atmospheric pressure. The derivative is approximated by finite differences between adjacent layers.

Figure 5 shows the average heating rate across all atmospheric columns at each atmospheric layer from the Earth's surface to 1500 meters for Leonardo. The figure shows that, on this scale, the FP64 reference results (left graph) closely matches versions A, B, and C (middle graph) and the results are nearly indistinguishable. We see visible

deviations for version D and much larger deviations for version E. This observation can be confirmed by considering the absolute difference in the heating rate compared to FP64 (right graph).

This shows that versions A and B have errors below 10^{-5} K/day, which can be considered negligible. Version C has errors around 10^{-5}-10^{-3} K/day, which are larger but still small compared to the magnitude of the heating rate. Version D has an average error of 0.4 K/day, which may still be acceptable for exploratory simulations. Version E, however, has an average error of around 3 K/day, which is too large to be considered physically acceptable. The numerical errors introduced by mixed precision should also be interpreted in the context of other uncertainties in radiative transfer calculations in Earth system models, such as cloud properties [18]. Overall, the mixed-precision configurations preserve the general heating-rate profiles, except for E, where the precision reduction was too aggressive.

5. Related Work and Discussion

There have been several studies on integrating mixed precision into weather and climate models. Váňa et al. [19] moved parts of the Integrated Forecast System (IFS) from double to single precision, reporting a 40% performance improvement. Lang et al. [20] showed that applying single precision to most, but not all, parts of a forecasting model improves performance and enables higher vertical resolution, ultimately improving forecast quality. Nakano et al. [21] found that using single precision throughout caused unacceptable errors, whereas retaining double precision only for grid-geometry setup reduced runtime by 46% with negligible accuracy loss. Klöwer et al. [22] demonstrated nearly a 4× speedup by running a nonlinear shallow-water model entirely in 16-bit arithmetic with negligible accuracy loss.

Together, these studies show that reduced precision can substantially improve performance in weather and climate applications, but they typically apply a uniform precision level or rely on manual tuning. In contrast, our approach uses automatic tuning to determine the best-performing mixed-precision configuration that satisfies a given error threshold.

Most closely related is the work of Tintó Prims et al. [23], who present a divide-and-conquer algorithm using the Reduced Precision Emulator [24] to identify which variables tolerate lower precision without sacrificing accuracy, evaluated on NEMO 4.0 and ROMS 3.6. Our approach differs in that we optimize for runtime performance rather than accuracy alone, by tuning data types alongside code parameters such as block size and loop unroll factor [11].

An important limitation of our work concerns how numerical error is quantified. We use RMSE as a selection criterion during auto-tuning, comparing outputs produced under different precision configurations. While RMSE is a practical and efficient metric for this purpose, it does not constitute a full scientific validation. Climate-model verification research has shown that numerical agreement alone is insufficient: ensemble-based consistency tests are needed to determine whether changes in arithmetic precision produce statistically distinguishable model behavior [25, 26]. As small perturbations may accumulate over time, a low RMSE indicates close short-term agreement but does not guarantee long-term statistical consistency. Evaluating such consistency is outside the scope of this work.

6. Conclusions and Future Work

We have shown that mixed-precision GPU computing can accelerate the radiative transfer solver, a critical component of weather and climate models. For example, mixed-precision version C provides speedups of 1.8× (Leonardo) and 2.1× (LUMI) over double precision, while providing better performance than uniform single precision at similar accuracy, whereas version B provides similar speedup, but higher accuracy compared to uniform single precision. These results demonstrate that mixed-precision tuning is a promising approach for improving the efficiency of radiative transfer. Additionally, because the mixed-precision strategies developed here are not specific to RTE+RRTMGP-CPP, they could also benefit RTE-RRTMGP and alternative implementations, such as the OpenACC port in ICON-A [27], the RRTMGPxx rewrite using modern C++ [28], and the Julia-based RRTMGP.jl [29].

In future work, we aim to focus on understanding the impact of reduced accuracy versions of RTE-RRTMGP-C++ upon the modeling error over longer simulation periods in more realistic RCEMIP-style settings. We also plan to explore more finer-grained precision assignments, for example by varying precision between different quadrature points or atmospheric columns. As single-column radiation calculations continue to be a performance bottleneck for Earth system models, our approach is scalable and well suited for large-scale simulations in next-generation weather and climate models on current petascale and future exascale systems.

Acknowledgements

The authors acknowledge that this work has received support from the European High Performance Computing Joint Undertaking (EuroHPC JU). ESiWACE3 is funded by the EuroHPC JU and national co-funding bodies (grant agreement No 101093054). Access to the EuroHPC supercomputer Leonardo was granted through the WeatherGenerator project (grant agreement No 101187947). Access to the EuroHPC supercomputer LUMI was granted through SURF (grant EINF-17348).

References

[1] R. Pincus, E. J. Mlawer, J. S. Delamere, Balancing Accuracy, Efficiency, and Flexibility in Radiation Calculations for Dynamical Models, Journal of Advances in Modeling Earth Systems 11 (10) (2019) 3074–3089.
[2] G. van den Oord, A. Sclocco, B. van Stratum, et al., Optimization of RTE+ RRTMGP-C++: a CUDA implementation of the radiation package RTE+ RRTMGP for atmospheric science, in: AGU Fall Meeting Abstracts, Vol. 2021, 2021, pp. A55A–03.
[3] R. J. Hogan, A. Bozzo, A flexible and efficient radiation scheme for the ecmwf model, Journal of Advances in Modeling Earth Systems 10 (8) (2018) 1990–2008.
[4] P. Ukkonen, R. Pincus, R. J. Hogan, K. Pagh Nielsen, E. Kaas, Accelerating radiation computations for dynamical models with targeted machine learning and code optimization, Journal of Advances in Modeling Earth Systems 12 (12) (2020) e2020MS002226.
[5] CUDA programming guide, `https://docs.nvidia.com/cuda/cuda-programming-guide/`, accessed: 2026-03-16 (2026).
[6] T. Palmer, Modelling: Build imprecise supercomputers, Nature 526 (7571) (2015) 32–33.
[7] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, S. Tomov, Accelerating scientific computations with mixed precision algorithms, Computer Physics Communications 180 (12) (2009) 2526–2533.
[8] A. Haidar, H. Bayraktar, S. Tomov, J. Dongarra, N. J. Higham, Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems, Proceedings of the Royal Society A 476 (2243) (2020) 20200110.
[9] N.-M. Ho, W.-F. Wong, Exploiting half precision arithmetic in Nvidia GPUs, in: 2017 IEEE High Performance Extreme Computing Conference (HPEC), IEEE, 2017, pp. 1–7.
[10] S. Heldens, B. van Werkhoven, Kernel Float: Unlocking mixed-precision GPU programming, ACM Trans. Math. Softw. 52 (1) (2026) 1–36.
[11] S. Heldens, B. van Werkhoven, Accuracy-Aware Mixed-Precision GPU Auto-Tuning, IEEE Trans. Parallel Distrib. Syst. 37 (4) (2026) 867–884. `doi:10.1109/TPDS.2026.3659324`.
[12] B. van Werkhoven, Kernel Tuner: A search-optimizing GPU code auto-tuner, Future Generation Computer Systems 90 (2019) 347–358.
[13] M. A. Veerman, B. J. H. van Stratum, C. C. van Heerwaarden, A Case Study of Cumulus Convection Over Land in Cloud-Resolving Simulations With a Coupled Ray Tracer, Geophysical Research Letters 49 (23) (2022) e2022GL100808.
[14] B. van Werkhoven, W. J. Palenstijn, A. Sclocco, Lessons learned in a decade of research software engineering gpu applications, in: Computational Science – ICCS 2020, Springer International Publishing, Cham, 2020, pp. 399–412.
[15] KMM documentation, `https://nlesc.github.io/kmm/`, accessed: 2026-03-24 (2026).
[16] HIP documentation, `https://rocm.docs.amd.com/projects/HIP/en/docs-6.3.3/index.html`, accessed: 2026-03-24 (2025).
[17] M. Lurati, S. Heldens, A. Sclocco, B. van Werkhoven, Bringing Auto-Tuning to HIP: Analysis of Tuning Impact and Difficulty on AMD and Nvidia GPUs, in: Euro-Par 2024: Parallel Processing, Springer, 2024, pp. 91–106.
[18] E. Jahangir, Q. Libois, F. Couvreux, B. Vié, D. Saint-Martin, Uncertainty of sw cloud radiative effect in atmospheric models due to the parameterization of liquid cloud optical properties, Journal of Advances in Modeling Earth Systems 13 (12) (2021) e2021MS002742.
[19] F. Váňa, P. Düben, S. Lang, T. Palmer, M. Leutbecher, D. Salmond, G. Carver, Single precision in weather forecasting models: An evaluation with the ifs, Monthly Weather Review 145 (2) (2017) 495–502.
[20] S. T. Lang, A. Dawson, M. Diamantakis, et al., More accuracy with less precision, Quarterly Journal of the Royal Meteorological Society 147 (741) (2021) 4358–4370.
[21] M. Nakano, H. Yashiro, C. Kodama, H. Tomita, Single precision in the dynamical core of a nonhydrostatic global atmospheric model: Evaluation using a baroclinic wave test case, Monthly Weather Review 146 (2) (2018) 409–416.
[22] M. Klöwer, S. Hatfield, M. Croci, P. D. Düben, T. N. Palmer, Fluid simulations accelerated with 16 bits: Approaching 4x speedup on A64FX by squeezing ShallowWaters. jl into Float16, Journal of Advances in Modeling Earth Systems 14 (2).
[23] O. Tintó Prims, M. C. Acosta, A. M. Moore, et al., How to use mixed precision in ocean models: Exploring a potential reduction of numerical precision in nemo 4.0 and roms 3.6, Geoscientific Model Development 12 (7) (2019) 3135–3148.
[24] A. Dawson, P. D. Düben, rpe v5: an emulator for reduced floating-point precision in large numerical simulations, Geoscientific Model Development 10 (6) (2017) 2221–2230.
[25] C. Zeman, C. Schär, An ensemble-based statistical methodology to detect differences in weather and climate model executables, Geoscientific Model Development 15 (8) (2022) 3183–3203.
[26] T. Price-Broncucia, A. Baker, D. Hammerling, M. Duda, R. Morrison, The ensemble consistency test: from CESM to MPAS and beyond, Geoscientific Model Development 18 (8) (2025) 2349–2372.
[27] M. A. Giorgetta, W. Sawyer, X. Lapillonne, et al., The ICON-A model for direct QBO simulations on GPUs, Geoscientific Model Development 15 (18) (2022) 6985–7016.
[28] B. Hillman, M. Norman, L. Bertagna, A. Donahue, P. Caldwell, RRTMGPxx: a portable radiation code for ultra- high-resolution modeling, Tech. rep. (10 2022). `doi:10.2172/2005353`.
[29] CliMA, RRTMGP.jl, https://github.com/CliMA/RRTMGP.jl, accessed: 2026-05-22 (2026).

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 106–135

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

GPU Acceleration of Ratio-Based Differential Transcriptomics

Ziyad AlBanoby[a,*], Suzanne Jin[b], Cristina Araiz[b], Mathys Grapotte[b], Cedric Notredame[b], Ionas Erb[b], David Vicente[a]

[a] *Barcelona Supercomputing Center, Barcelona, Spain*
[b] *Center for Genomic Regulation, Barcelona, Spain*

Abstract

Differential expression analysis traditionally relies on normalization assumptions that are difficult to verify in practice. The `propr` R package addresses this through differential proportionality, that is, the analysis of ratios of gene expression rather than normalized counts, which reflects mRNA stoichiometry directly and without normalization assumptions. However, the approach carries a significant computational cost that has limited its applicability to large datasets. Here we present a substantially redesigned version of `propr`, featuring a modernized software architecture, package-wide performance improvements, and GPU acceleration of core computational components. These advances reduce runtime approximately 20-fold for large bulk RNA-seq data and 400-fold for single-cell data, enabling rapid, iterative data exploration and bringing large-scale single-cell atlases within practical reach of ratio-based differential expression analysis.

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026

Keywords: GPU acceleration; Logratio Analysis; Differential proportionality; Differential expression analysis; Transcriptomics; Compositional data analysis; Epicure;

1. Introduction

High-throughput sequencing experiments generate count data from molecular samples, allowing quantification of gene expression in a cell (for transcriptome analysis), or species abundances in an environmental sample (for microbiome analysis), to mention two main applications. However, the experimental protocols for this are often highly complex and inherently constrained by technical limitations such as measurement noise and finite sequencing depth, leading to readouts of count totals that do not faithfully reflect the true counts in the biological samples. As a result, the count data generated do not provide a meaningful absolute scale, instead they are more appropriately interpreted in relative terms [1,2]. The observed count for any given gene, for instance, rather than representing its actual molecular abundance, is better described by its proportion of expression within its sample. This relative nature implies that an apparent change in one gene may arise not only from its own variation but also from the changes in other genes.

* Corresponding author
E-mail address: ziyad.albanoby@bsc.es

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.024

Any analysis that involves comparing samples across conditions is directly impacted by these limitations. This is particularly relevant for differential expression analysis, which seeks to identify genes whose expression levels change between experimental groups.

Standard workflows typically address the problem through normalization procedures[3]. In `edgeR`[4] and `DESeq2`[5], normalization factors enter generalized linear models as offsets; in `limma-voom`[6] they rescale counts prior to linear modelling. In either case, the validity of downstream inference depends on assumptions about the biological system that are difficult to verify directly:[7] absent spike-ins or other external references, one typically assumes that most genes are not systematically changing between conditions. The success of standard tools suggests the assumption is often reasonable and works well in practice. It can, however, fail silently, leaving users unaware that an implicit assumption has been violated.

There is thus a need for methods that can deal with situations where normalizations are not valid (for example, due to a global transcriptional shift in tumor cells[8] or the loss of a nucleosome[9]). Such a method, inspired by the perspective of compositional data analysis[10], was introduced in the form of *differential proportionality*[11]. This idea reframes the analysis from individual genes to ratios between pairs of genes, so that sample-specific scaling factors cancel out, making inference independent of normalization, but also allowing direct detection of biologically meaningful changes in stoichiometry. Ratios are of specific interest in transcriptomics, where cellular phenotypes are influenced not only by the expression of individual genes but also by the balance among interacting transcripts, regulatory partners, pathways, and macromolecular complexes. In these contexts, relative expression and stoichiometry can convey biologically meaningful information that marginal abundances alone may obscure, revealing differential changes in stoichiometry of genes that may themselves not be differentially expressed. Differential proportionality thus provides a normalization-free framework both for differential abundance analysis under compositional constraints and for directly studying relative changes as biologically informative signals.[12] The framework was implemented in the `propr` R package,[13] which was originally designed for the identification of proportional gene pairs[14,15] without accounting for different experimental conditions, with a recent addition of a shrinkage procedure for logratio-based partial correlations.[16]

Nonetheless, practical use of differential proportionality has long been constrained by computational cost. Because the method operates on all gene pairs, the number of comparisons grows quadratically with the number of genes. This challenge is further compounded by the intrinsic dependencies in compositional data, which violate the assumptions of many conventional significance tests. While the original framework includes both a parametric and a permutation-based procedure to assess statistical significance, the permutation procedure requires fewer assumptions for the test to be valid, yet entails repeating the full set of pairwise computations, multiplying the computational burden. In practice, users often need to explore several configurations before selecting an appropriate one for a given experiment, further extending runtime. Altogether, this places ratio-based analyses in a substantially more demanding computational regime than standard gene-wise differential expression methods.

We address these limitations through a GPU-accelerated implementation of the `propr` package. The general structure of computations involving logratios is embarrassingly parallel: for every pair of genes, the same operation is performed independently. By leveraging the parallelism of GPU hardware, millions of computations can be carried out concurrently, dramatically reducing runtime and making the ratio-based method feasible at scales that were previously impractical.

In section 2, we describe the baseline package architecture and a typical scientific workflow using it. The design decisions behind the optimized implementation, and the GPU acceleration strategy used to reduce the computational burden of pairwise transcriptomic analysis are presented in section 3. In section 4, the benchmark results comparing CPU and GPU based performance are presented. An Appendix is provided for some of the more technical details regarding the evaluation and implementation of relevant statistics.

2. Package Overview and Baseline Architecture

2.1. Package Scope

The `propr` R package[13] is designed for the analysis of relative omics data in the form of counts, where the total counts in each sample are considered a source of bias. Besides the scale-free evaluation of covariation measures such

as proportionality[14,15] and logratio-based partial correlations,[16] it enables the detection of changes in gene ratios between conditions. The latter, known as differential proportionality,[11] is the focus of the present work, where we put the emphasis on transcriptomic analysis.

Rather than testing genes individually, the idea is to study the change of gene ratios between experimental groups. The package implements several related formulations, including standard, α-transformed, moderated, and weighted variants, so that users can adapt the analysis to sparse data, few samples, or heterogeneous-sample settings. A typical workflow (Figure 1) begins from a count matrix and group labels, computes pairwise proportionality statistics, and then extracts significant gene pairs together with intermediate quantities such as log-ratio means and variances for downstream interpretation.

Due to the strong mean-variance dependence and heterogeneity of variance across genes, linear modelling of differential expression is commonly done for individual genes rather than in a joint multivariate analysis. In our approach, the usual one-way analysis of variance (ANOVA) is adopted to decompose the variance of logratios of feature pairs rather than of the features themselves. Let $\mathbf{y}_i = (Y_{1i}, \ldots, Y_{Ni})$ denote the counts belonging to gene i in samples $1, \ldots N$. The (maximum-likelihood) empirical variance estimator of the logratios between the counts of genes i and j is

$$\mathrm{LRV}_{ij} := \mathrm{LRV}(\mathbf{y}_i, \mathbf{y}_j) = \frac{1}{N} \sum_{k=1}^{N} \left(\log \frac{Y_{ki}}{Y_{kj}} - \mathrm{LRM}_{ij} \right)^2, \tag{1}$$

where LRM denotes the logratio mean

$$\mathrm{LRM}_{ij} := \mathrm{LRM}(\mathbf{y}_i, \mathbf{y}_j) = \frac{1}{N} \sum_{k=1}^{N} \log \frac{Y_{ki}}{Y_{kj}}. \tag{2}$$

Assume each sample can be assigned to one of two groups of sizes n and $N-n$, and let $\mathbf{y}_i^1$ and $\mathbf{y}_i^2$ denote their respective count vectors for gene i. We also denote the respective statistics evaluated on the two groups by LRM^1, LRM^2, LRV^1, and LRV^2. We can decompose the total empirical logratio variance in the data as

$$\mathrm{LRV}_{ij} = \frac{n(N-n)}{N^2} \left(\mathrm{LRM}_{ij}^1 - \mathrm{LRM}_{ij}^2 \right)^2 + \frac{n\mathrm{LRV}_{ij}^1 + (N-n)\mathrm{LRV}_{ij}^2}{N}. \tag{3}$$

Here, the first term represents the between-group variance, and the second term the within-group variance. A statistic quantifying the ratio of within-group variance to total variance is given by

$$\theta_{ij} = \frac{n\mathrm{LRV}_{ij}^1 + (N-n)\mathrm{LRV}_{ij}^2}{N\mathrm{LRV}_{ij}}. \tag{4}$$

As shown in the original article,[11] the θ statistic, when also providing the sample size N, is equivalent to the squared t-statistic of differential expression tests. Since the approach is similar to the gene-wise linear modelling performed in the `limma` R package,[17] several of `limma`'s statistical procedures were adapted for ratios. θ_{ij} can be moderated to account for small sample sizes,[18] the mean-variance trend of count logratios can be taken into account similar to the limma-trend procedure,[18,6] and also `limma` sample weights[19] are incorporated. Users may supply sample-level gene weights, which are interpreted as precisions and therefore combined into approximate ratio weights via their harmonic mean. We find that such approximate weights of the limma-voom type are outperformed by direct detrending of log-ratios. A manuscript with details is currently in preparation.

The use of pseudocounts to account for zeros in traditional differential expression analysis can be avoided in `propr` by specifying an approximation of the logarithm using a parameter $0 < \alpha \leq 1$ (here referred to as α-transformation). A similar, ANOVA-like decomposition can be obtained for the χ^2-distance of power-transformed proportions of counts (see Appendix B), making use of an equivalence between logratio variance and the χ^2 statistic in the limit of powers tending to zero.[20]

2.2. Package Architecture

`propr` implements its most computationally expensive routines in C++, with integration into R provided through Rcpp. In proportionality analysis, the log-ratio-transformed matrix is passed to the C++ functions that generate a

$G \times G$ matrix – where G is the number of genes. The variance of log-ratios (VLR) is computed from the covariance matrix of the log-ratios (`lr2vlr`), from which the proportionality measures ρ, ϕ, and ϕ_s are obtained (`lr2rho`, `lr2phi`, `lr2phs`). Partial-correlation variants (`pcor`,[21] `pcor.shrink`,[22] `pcor.bshrink`,[16]) make use of the `ppcor`[21] or `corpcor`[23] packages, optionally incorporating basis-level shrinkage. For differential proportionality (`propd`), the C++ routines `lrv` and `lrm` compute log-ratio variances and means for the complete dataset and for each group, considering all $G(G-1)/2$ gene pairs. The four theta statistics— θ_d (disjointed), θ_e (emergent), θ_f, and θ_g—are subsequently constructed in R from the within-group and total sums of squares.

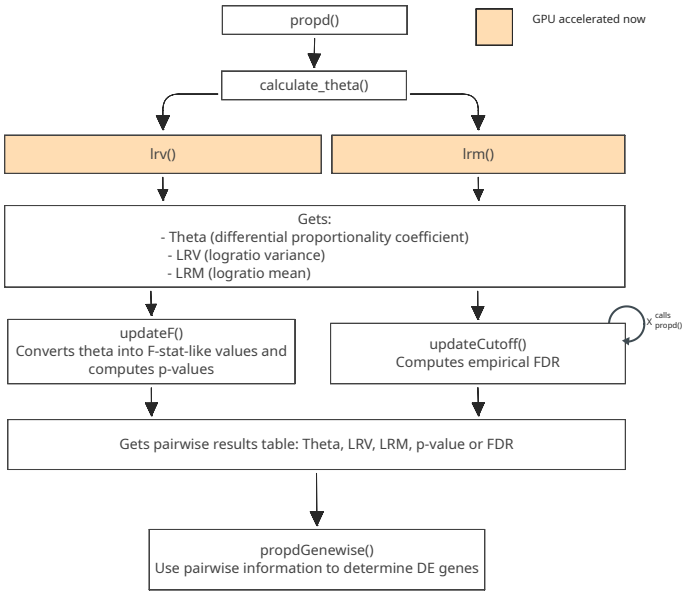


Fig. 1. Execution flow of `propd()`

2.3. Baseline Performance Limitations

The main computational bottleneck in `propr` stems primarily from the quadratic complexity of the pairwise gene computations. Every core routine computes on $G(G-1)/2$ gene pairs. If we consider a standard human RNA-seq data set with 18,244 genes, this would amount to approximately 166 million pairwise comparisons with each further requiring an inner loop over N samples. The cost increases further when false discovery rates (FDR) are estimated by permutations, since each permutation requires recomputing the full set of pairwise statistics, leading to substantially longer end-to-end runtimes as the number of permutations increases, as shown in Figure Appendix D.

In the current implementation considered here, `propr` executes its pairwise computations sequentially utilizing only a single CPU core. Consequently, the package does not exploit modern multi-threading for its most expensive computations. The package does however make use of permutation level parallelism (via `parallel::parLapply`), which distributes independent permutation runs across cores but does not reduce the latency of a single pass. This combination of quadratic algorithmic complexity, serial execution, and high memory usage limits scalability for large-scale analysis, as shown by the runtime and memory profiling benchmarks (Figs. Appendix D and Appendix F).

2.3.1. Profiling and Bottleneck Analysis

To characterize the runtime behavior of `propr` we developed an NVTX (NVIDIA Tools Extension[24]) R profiling package[1] that allowed us to annotate specific code regions in `propr`. These annotations were used to identify dominant

[1] nvtxR: https://github.com/zalbanob/nvtxR

compute regions, compare pre- and post-optimization runs, and attribute runtime costs to specific `propr` operations. Examples of the collected CPU and GPU traces are shown in Figure E.8 in the Appendix.

2.3.2. Package-Level Refactoring

One of the first major changes was to refactor the package to support GPU execution while preserving the original CPU implementation. Keeping the CPU version was important both for users who might not have access to a GPU and for checking the correctness of the GPU implementation. Before this refactoring, the codebase had a fairly flat structure, with little abstraction which led to tight coupling between the Rcpp interface and the CPU kernels. Although this worked for the original CPU-only version, it would have made it much harder to support both CPU and GPU execution through the same interface.

To address this, the package was reorganized into a layered structure that separated the main responsibilities more clearly. Users' R code now goes through a dedicated `LibPropR` interface, which manages device/host execution. From there, computation is passed to the CPU or GPU dispatcher, which sets up kernel parameters, allocates memory for intermediate results, and manages host-to-device memory transfers before calling the corresponding kernels. This change reduces the coupling between the high-level package logic and the backend-specific code, improving overall modularity. Figure 2 illustrates this structure and shows how it makes the current implementation easier to maintain and simplifies the addition of potential future backends.

We also migrated the build system to CMake to better manage the added complexity introduced by GPU support. The original Makefile-based system was designed around the CPU implementation and was not well suited to managing separate CPU and GPU source files, different compiler toolchains, and the additional build steps required for CUDA code.

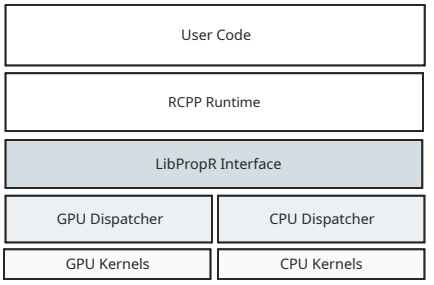


Fig. 2. Improved package structure for `propr`

3. GPU Implementation

3.1. Integration into the Package Workflow

One of the main goals in implementing the GPU backend was to do so without imposing any significant changes to the existing user-facing API or disrupting existing workflows. Consequently, the public API was left unchanged. Backend selection could be enabled globally via `options(propr.backend="GPU")`, on a per-function basis `propr:::phiRcpp(x, backend = "GPU")`, or delegated to the library at runtime, via the `backend = "auto"`, as illustrated in Figure 3. However, this requirement conflicted with the general practice of managing GPU memory: to keep data resident on the GPU for as long as possible and copied back to the host only when necessary.

An approach we considered was R's ALTREP (Alternative Representation) framework,[25] which allows R's standard vector types to be backed by a custom internal representation without exposing new classes to the user. In principle, this approach was attractive, because it appeared to offer a way for data to remain GPU-resident for the execution period within `propr`. In practice, however, ALTREP was not well suited for this case, primarily because the compiled code paths would ultimately have requested pointer access, and the ALTREP object would have been forced to materialize on the host, after which the data would still have to be copied to the GPU before the computation could proceed.

More fundamentally, ALTREP is intended to be an alternative storage representation for standard R objects; it does not provide an execution model within that representation. Achieving the latter would have in practice entailed user-visible classes and overloaded operators, which would have conflicted with the goal of preserving the existing API.

We therefore adopted a simpler approach, based on explicit device memory management in the Rcpp layer. For each GPU operation, we allocate memory for the output, intermediates (if any), and copy the input from the host to device, execute the computation, and copy back our results into standard Rcpp objects. Although this approach does not allow for data to persist on the GPU between successive or subsequent GPU-enabled function calls, it preserves the user-facing API and avoids fragile interactions with the R memory model and GPU data management.

The profiling results in Fig. E.9 quantify this trade-off. In both the 10- and 100-permutation runs, the GPU runtime is dominated by host-device memory transfers rather than kernel execution. In both runs, combined memory-transfer operations account for approximately 92% of the total GPU runtime, while kernel execution time amounts to approximately 8%. These results indicate that the current GPU implementation is primarily transfer-bound rather than compute-bound.

```r
options(propr.backend = "cuda")  # 1 Enable globally
library(propr)
set.seed(42)

# Parameters
nsamples <- 1024
nfeat    <- 1024

# Feature-specific parameters on the log scale
mu    <- rnorm(nfeat, mean = 0, sd = 1)
sigma <- runif(nfeat, min = 0.5, max = 1.5)

# Generate Z ~ N(0,1) and map to X = exp(mu_j + sigma_j * Z_ij)
Z <- matrix(rnorm(nsamples * nfeat), nrow = nsamples, ncol = nfeat)
x <- exp(sweep(Z, 2, sigma, `*`) + rep(mu, each = nsamples))
x <- x * exp(matrix(rnorm(nsamples * nfeat, sd = 1e-9), nrow = nsamples))

res_vlr_cpu <- propr:::vlrRcpp(x, backend = "cpu")  # 2 Enable per function
res_vlr_gpu <- propr:::vlrRcpp(x, backend = "cuda")
stopifnot(isTRUE(all.equal(res_vlr_cpu, res_vlr_gpu, tolerance = 1e-6)))
```

Fig. 3. Toggling GPU acceleration in `propr`. 1 The global option applies to all subsequent calls. 2 The `backend` argument overrides the global setting on a per-call basis.

3.2. GPU Porting

A direct port of the original CPU routines to the GPU would not have been sufficient to produce the gains reported in Section 4. The baseline implementation repeatedly materialized intermediate arrays and performed multiple passes over the same data. On the GPU, such a kernel implementation would have generated excessive global-memory traffic and kernel-launch overhead. For these reasons, the porting effort relied on two key optimizations: online statistical formulations and kernel fusion.

3.2.1. Online statistical formulations

The main pairwise statistics in `propr` were reformulated as streaming computations. Each GPU thread is assigned a unique gene pair for which it accumulates the required quantities while traversing the samples. In practice, this meant expressing means, variances, and covariances through recurrence-based updates and algebraic decompositions, as detailed in Appendix A. For the basic logratio variance and covariance, this takes the form of Welford-style[26] online updates of the running means and centered sums. The weighted case is handled similarly through the corresponding weighted recurrence. For the basic log ratio mean, we used the identity $\log(a/b) = \log(a) - \log(b)$ to compute per-gene

intermediates first and then compute the pairwise ratio by subtraction, reducing the computational work from $O(G^2N)$ to $O(GN + G^2)$, where G is the number of genes and N is the number of samples. The α-transformed variants were similarly reorganized to require one pass, or at most two passes, over the full data and the relevant subgroup, while preserving the original statistical formulation; see Appendix A and Appendix B in the Appendix.

This reformulation allowed us to reduce temporary storage and avoid repeated passes over data, which would otherwise have generated a significant amount of expensive global-memory traffic. A valid concern with this reformulation, however, is its potential impact on numerical accuracy. Our proposed GPU implementation makes use of recurrence-based updates whose numerical behavior has been studied previously for one-pass mean and variance algorithms[27] and, more recently, for numerically stable parallel computation of variance and covariance[28]. Consistent with these results, we empirically verified that the GPU implementation closely reproduce the reference CPU outputs for all directly computed backend quantities. Across the largest mouseStemCellsMatrix and GTExMatrix validation cases, the maximum absolute CPU/GPU difference was below $1.3e^{-3}$ for all $LRV*$, $LRM*$, and $\theta*$ statistics. The largest deviations occurred for theta-like quantities, with max $|\theta_{gpu} - \theta_{cpu}|$ of $1.26e^{-3}$ for mouseStemCellsMatrix and $9.59e^{-4}$ for GTExMatrix, while lrm-family quantities remained below $1.1e^{-5}$. A more detailed breakdown across the different sample sizes and number of genes for both of the aforementioned datasets can be found in Appendix C.

Below, we derive online formulations for the two core pairwise statistics in `propr`: the log-ratio mean (LRM) and log-ratio variance (LRV), which allow for efficient single-pass computation; derivations for the remaining kernels follow the same strategy and hence are provided in Appendix A

Logratio Mean (LRM): For two genes i, j,

$$\mathrm{LRM}_{ij} = \frac{1}{n}\sum_{k=1}^{n}\log\left(\frac{Y_{ki}}{Y_{kj}}\right) = \frac{1}{n}\sum_{k=1}^{n}\log(Y_{ki}) - \frac{1}{n}\sum_{k=1}^{n}\log(Y_{kj}) = \boxed{\mu_i - \mu_j}$$

Thus, we first compute the mean log of each gene's counts in the first pass and then obtain each pairwise LRM_{ij} by subtraction. This allows us to reduce the computational complexity from $O(G^2N)$ to $O(GN + G^2)$, using $O(G)$ extra space.

Logratio Variance (LRV):

For each sample k, let $r_{k,ij} = \log(x_{ki}/x_{kj})$ and $\mu_{ij} = \frac{1}{n}\sum_{k=1}^{n} r_{k,ij}$. Then

$$V_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(r_{k,ij} - \mu_{ij})^2.$$

With $S_1 = \sum_{k=1}^{n} r_{k,ij}$ and $S_2 = \sum_{k=1}^{n} r_{k,ij}^2$,

$$\sum_{k=1}^{n}(r_{k,ij} - \mu_{ij})^2 = \sum_{k=1}^{n} r_{k,ij}^2 - 2\mu_{ij}\sum_{k=1}^{n} r_{k,ij} + n\mu_{ij}^2 = S_2 - \frac{S_1^2}{n},$$

so

$$V_{ij} = \frac{S_2 - S_1^2/n}{n-1}.$$

To compute this stably in one pass, we define

$$\mu_{ij}^{(k)} = \frac{1}{k}\sum_{t=1}^{k} r_{t,ij}, \qquad M_{ij}^{(k)} = \sum_{t=1}^{k}(r_{t,ij} - \mu_{ij}^{(k)})^2,$$

and update, for each new $r_{k,ij}$,

$$\delta_k = r_{k,ij} - \mu_{ij}^{(k-1)}, \qquad \mu_{ij}^{(k)} = \mu_{ij}^{(k-1)} + \frac{\delta_k}{k}, \qquad M_{ij}^{(k)} = M_{ij}^{(k-1)} + \delta_k(r_{k,ij} - \mu_{ij}^{(k)}),$$

starting from $\mu_{ij}^{(1)} = r_{1,ij}$ and $M_{ij}^{(1)} = 0$. After n samples,

$$V_{ij} = \frac{M_{ij}^{(n)}}{n-1}.$$

3.2.2. Kernel fusion

In several routines, a naive GPU implementation of their CPU counterparts would still have followed the same staged structure, materializing large intermediate arrays in global memory only to read them back immediately in the following stage. This, however, would have constituted a large overhead on the GPU due to round-trips to global memory, which is expensive to access. Thus, wherever possible, dependent stages were fused, allowing intermediate computed quantities to be consumed directly as they were produced or recovered through inexpensive reductions.

This is particularly relevant for routines whose CPU implementation proceeds through several intermediate matrix forms. For example, as shown in Appendix A.10, the proportionality coefficient ϕ does not require explicit materialization of the CLR matrix: once the VLR matrix has been computed, the required CLR variances can be recovered from row-wise and global reductions over VLR alone. Similarly, the GPU implementation of VLR evaluates it directly using the same streaming formulation for the basic logratio variance, rather than reproducing the CPU implementation approach that explicitly materializes the covariance matrix. In this way, intermediate quantities are either consumed immediately inside the same kernel or reduced to a much smaller set of partial values before the next stage.

This reduced memory traffic and kernel-launch overhead, while increasing computational intensity. Together, these optimizations were central to the performance of the GPU implementation. The resulting speedups came not only from GPU parallelism, but also from restructuring the computation to better match the hardware by streaming pairwise interactions, avoiding unnecessary intermediates, and reducing data movement. These benefits were especially noticeable for larger workloads.NVIDIA Nsight Compute Performance analysis is presented in Appendix H for the kernel that dominated the CPU runtime.

3.3. Parallelization Approach

Since our computational problem is based on symmetric pairwise interactions between distinct genes, the natural output space is the strict upper triangle of a $G \times G$ matrix rather than the full matrix. Creating threads over the entire matrix would waste work on diagonal and redundant lower-triangular entries. We therefore use a triangular indexing scheme that maps each thread id directly to one valid pair (r, c) with $r < c$. This process is illustrated in Figure 4

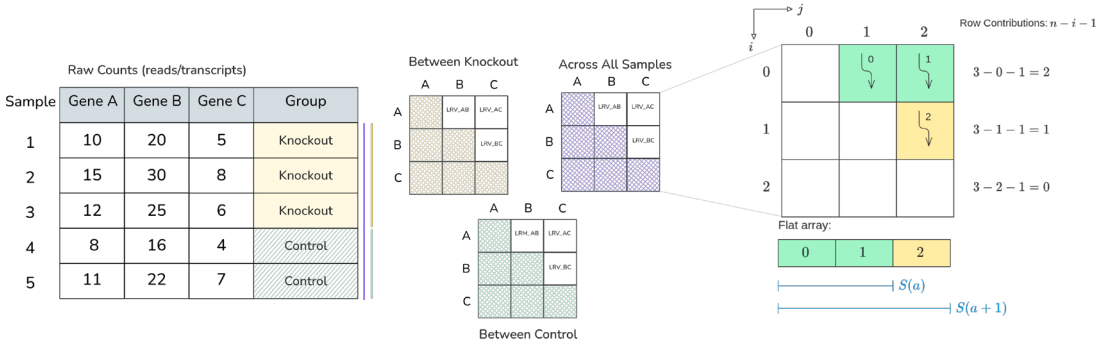


Fig. 4. Construction of the pairwise log-ratio variance matrices used by `propd`. The input count table is split by the supplied group labels, here Knockout and Control, and `calculate_theta` computes one gene-pair LRV matrix across all samples and one LRV matrix within each group. These matrices are then combined as within-group sums of squares relative to the total LRV to compute the differential proportionality statistic, θ_d. To the right we show how `propr`'s pairwise LRV computation between genes (i, j) is mapped to threads using their thread id's

Mapping thread id to upper-triangular matrix coordinates. We assign one thread to each pair (r, c) in the strict upper triangle of a $G \times G$ output matrix, where $0 \le r < c < G$. The total number of such pairs is $G(G-1)/2$, so thread $t \in \{0, \ldots, G(G-1)/2 - 1\}$ is a flattened index over these pairs, enumerated row by row. Let $S(r) = r(2G - r - 1)/2$ count the upper-triangular entries preceding row r. The row index for thread t is the unique integer r that satisfies $S(r) \le t < S(r+1)$. Inverting the closed form of $S(r)$ gives r. We ignore the larger root, because r must lie within $[0, G)$ and is $\sim 2G > G$. Once r is known, the offset within our row is $\delta = t - S(r)$ and the column index is $c = r + 1 + \delta$.

Hence, the full mapping from thread id t to matrix coordinates (r, c) is

$$r = \left\lfloor \frac{(2G-1) - \sqrt{(2G-1)^2 - 8t}}{2} \right\rfloor, \quad c = r + 1 + t - \frac{r(2G-r-1)}{2}.$$

4. Performance Evaluation

4.1. Experimental Setup

Two data sets are used for the benchmark, bulk RNA-seq data in human cortex (male vs female) from the GTEx project[29] in the version available through the `recount3` R package,[30] and a single-cell data set in mouse that was used previously in a study of cell-cycle regulation.[31] For the latter, samples from different cell-cycle phases (S vs M/G2) are compared. In both data sets, weakly expressed genes were filtered out. Both data sets represent typical use cases for a normalisation-free differential expression analysis enabled by `propr`. To obtain a performance evaluation in terms of sample size and number of genes, the two data matrices are analyzed in full and in their reduced versions with randomly selected subsets of both genes and samples, see Fig.5. The sample sizes of the groups were kept balanced.

For benchmarking and performance analysis, all experiments were performed on the MareNostrum 5 supercomputer using a single NVIDIA H100 GPU equipped with 64 GiB of HBM2e memory and an Intel(R) Xeon(R) Platinum 8460Y+ CPU with 512 GB of RAM.

4.2. Package-Level Performance Results

The end-to-end benchmark results in Figure 5, as well as the routine-level benchmark results in Figure G.11, show that the GPU backend outperforms the CPU implementation both for the overall package workflow and on a per kernel basis. This improvement is observed for both benchmark data sets, GTExMatrix and mouseStemCellsMatrix. For GTExMatrix, the speedup is modest in smaller cases and increases with larger sample groups. For mouseStemCellsMatrix, the gains are larger overall because of the larger problem sizes.

Routine-level benchmarks show a similar pattern. While the speedup is not uniform across all routines, the largest gains happen in kernels that underly the differential proportionality computation. This is seen with `lrm` and `lrv` as they perform $O(G^2)$ pairwise computations and thus we see speedups that exceeds 1,000× for large inputs. `cor`, `cov`, `clr`, `lin`, `rho`, `phi`, and `vlr` also gain speed with increased input size, from about 5–50× to 100–600×. `llt`, `urt`, `lab`, and `ctz` stay near 5–50×.

The benchmarking and simulation runs produced several gigabytes of output data. These results were stored in Feather format using the `R` package `arrow`[32] which enabled fast and efficient file storage and subsequent loading for analysis, which would have otherwise been difficult using default text-based formats such as CSV due to slower read/write speeds and larger file sizes. This made post-processing, aggregation, and GPU–CPU correctness checks more efficient.

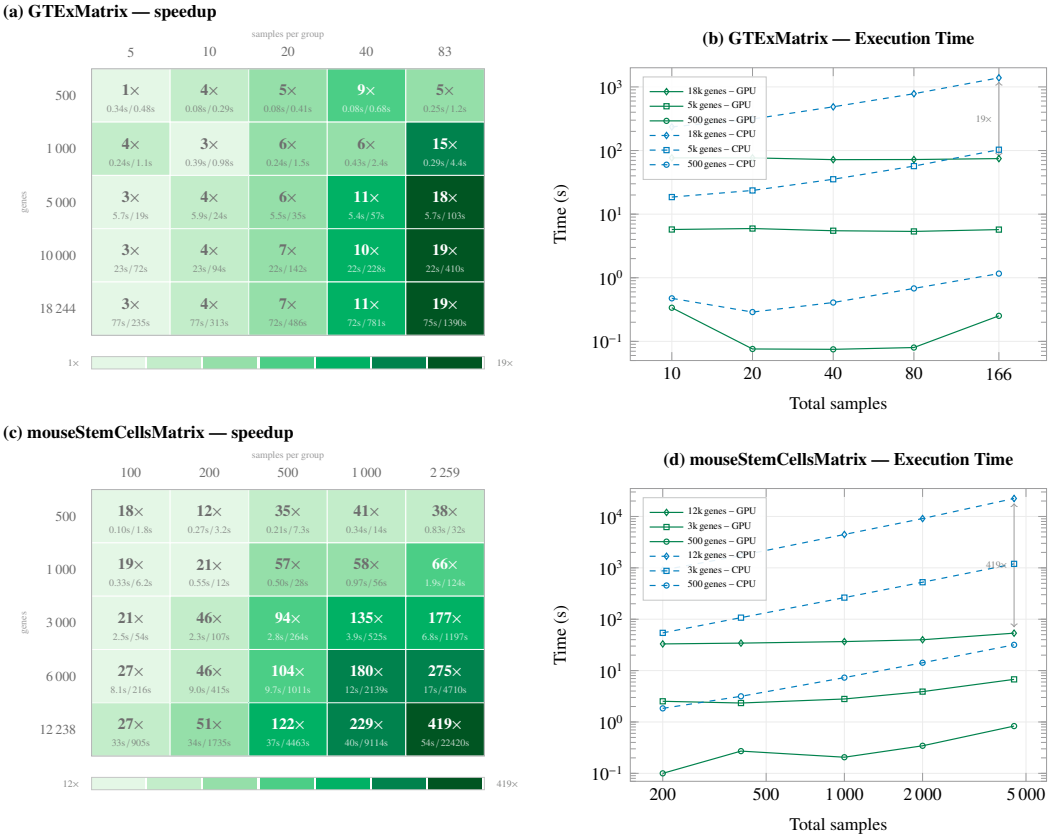


Fig. 5. GPU vs CPU benchmark comparison. (a, c) Heatmaps illustrating the relative speedup of GPU execution over CPU execution for the GTExMatrix and mouseStemCellsMatrix datasets, respectively. Speedup factors are shown for varying number of genes (y-axis) and samples per group (x-axis), with raw execution times (GPU/CPU) provided in each cell. Darker green shades indicate higher acceleration. (b, d) Execution time (seconds) as a function of total samples for the GTExMatrix and mouseStemCellsMatrix datasets, respectively. Solid lines represent GPU performance, while dashed lines represent CPU performance. Different markers and shades of green/blue denote varying number of genes. The vertical gaps (e.g., 19× and 419×) highlight the significant reduction in computational latency achieved by GPU acceleration as data scale increases.

5. Discussion

Here, we presented an optimized implementation of the `propr` R package with a focus on differential proportionality analysis, combining package-level refactoring with GPU acceleration of the most computationally intensive pairwise routines. This substantially reduces the cost of the core pairwise computations, making differential proportionality practical for routine transcriptomic analysis. At transcriptome scale, it addresses questions of relative balance between features that remain hidden in conventional gene-wise analyses. Although the pairwise ratio approach is tailored to quantifying stoichiometry of mRNA abundances and as such appears naturally in biological questions, it also comes with challenges. The first is technical: the number of comparisons increases quadratically with the number of genes or transcripts, with the corresponding requirements in resources like memory and computation time. Our GPU implementation largely addresses this, but gene-wise approaches remain more efficient in certain respects, notably multiple testing burden and memory consumption. The second concerns interpretability: downstream analysis of many ratios can pose new problems. Visualizing ratios as edges in a gene graph helps, and other exploratory tools for feature ratios are available.[33] However, standard tools for downstream analysis like gene-set enrichment analysis have only been partially adapted to ratios.[34] Our procedure to recast differential proportionality as gene-wise differential expression will address this gap (manuscript in preparation). Together, these developments make ratio-based approaches more accessible and scalable, broadening the toolkit available for transcriptomic studies where the assumptions of global normalization are difficult to justify.

Acknowledgements

The authors gratefully acknowledge the EuroHPC Joint Undertaking for awarding access to MareNostrum 5, at the Barcelona Supercomputing Center (BSC), Spain, under project EHPC-DEV-2025D04-122. The authors also thank the EPICURE[2] Support Team at BSC for their technical assistance in porting and optimizing the code for GPUs.

References

1. Quinn, T.P., Erb, I., Richardson, M.F., Crowley, T.M.. Understanding sequencing data as compositions: an outlook and review. *Bioinformatics* 2018;**34**(16):2870–2878.
2. Gloor, G.B., Macklaim, J.M., Pawlowsky-Glahn, V., Egozcue, J.J.. Microbiome Datasets Are Compositional: And This Is Not Optional. *Frontiers in Microbiology* 2017;**8**.
3. Robinson, M.D., Oshlack, A.. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology* 2010; **11**:R25.
4. Robinson, M.D., McCarthy, D.J., Smyth, G.K.. edger: a bioconductor package for differential expression analysis of digital gene expression data. *bioinformatics* 2010;**26**(1):139–140.
5. Love, M.I., Huber, W., Anders, S.. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology* 2014; **15**(12):550.
6. Law, C.W., Chen, Y., Shi, W., Smyth, G.K.. voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology* 2014;**15**:R29.
7. Lovén, J., Orlando, D.A., Sigova, A.A., Lin, C.Y., Rahl, P.B., Burge, C.B., et al. Revisiting Global Gene Expression Analysis. *Cell* 2012; **151**(3):476–482.
8. Lin, C.Y., Lovén, J., Rahl, P.B., Paranal, R.M., Burge, C.B., Bradner, J.E., et al. Transcriptional amplification in tumor cells with elevated c-myc. *Cell* 2012;**151**(1):56–67.
9. Hu, Z., Chen, K., Xia, Z., Chavez, M., Pal, S., Seol, J.H., et al. Nucleosome loss leads to global transcriptional up-regulation and genomic instability during yeast aging. *Genes & development* 2014;**28**(4):396–408.
10. Greenacre, M., Grunsky, E., Bacon-Shone, J., Erb, I., Quinn, T.. Aitchison's compositional data analysis 40 years on: A reappraisal. *Statistical Science* 2023;**38**(3):386–410.
11. Erb, I., Quinn, T., Lovell, D., Notredame, C.. Differential proportionality–a normalization-free approach to differential gene expression. *Proceedings of CoDaWork 2017, The 7th Compositional Data Analysis Workshop* 2017;URL https://doi.org/10.1101/134536.
12. Quinn, T.P., Erb, I., Gloor, G., Notredame, C., Richardson, M.F., Crowley, T.M.. A field guide for the compositional analysis of any-omics data. *GigaScience* 2019;**8**(9):giz107.
13. Quinn, T.P., Richardson, M.F., Lovell, D., Crowley, T.M.. propr: An R-package for Identifying Proportionally Abundant Features Using Compositional Data Analysis. *Scientific Reports* 2017;**7**(1):16252.
14. Lovell, D., Pawlowsky-Glahn, V., Egozcue, J.J., Marguerat, S., Bähler, J.. Proportionality: A Valid Alternative to Correlation for Relative Data. *PLoS Computational Biology* 2015;**11**(3).
15. Erb, I., Notredame, C.. How should we measure proportionality on relative gene expression data? *Theory in Biosciences* 2016;**135**:21–36.
16. Jin, S., Notredame, C., Erb, I.. Compositional covariance shrinkage and regularised partial correlations. *Statistics and Operations Research Transactions (SORT)* 2023;**47**(2):245–268.
17. Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* 2015;**43**(7):e47.
18. Smyth, G.K.. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology* 2004;**3**:Article3.
19. Liu, R., Holik, A.Z., Su, S., Jansz, N., Chen, K., Leong, H.S., et al. Why weight? modelling sample and observational level variability improves power in rna-seq analyses. *Nucleic Acids Research* 2015;**43**(15):e97–e97.
20. Greenacre, M.. Power transformations in correspondence analysis. *Computational Statistics & Data Analysis* 2009;**53**(8):3107–3116.
21. Kim, S.. ppcor: An r package for a fast calculation to semi-partial correlation coefficients. *Communications for Statistical Applications and Methods* 2015;**22**(6):665–674.
22. Schäfer, J., Strimmer, K.. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology* 2005;**4**(1):32.
23. Schäfer, J., Opgen-Rhein, R., Zuber, V., Ahdesmäki, M., Silva, A.P.D., Strimmer, K.. *corpcor: Efficient Estimation of Covariance and (Partial) Correlation*; 2021. R package version 1.6.10. Accessed: 2026-03-30; URL https://CRAN.R-project.org/package=corpcor.
24. NVIDIA Corporation, . NVIDIA tools extension library (NVTX). 2023. Available at: https://docs.nvidia.com/tools-extension/index.html. Accessed: 2026-03-30.
25. Tierney, L., Gabe, G., Kalibera, T., Lawrence, M.. ALTREP: Alternative representations for R objects. In: *useR! Conference*. Brussels, Belgium; 2017, .

[2] https://epicure-hpc.eu/

26. Welford, B.P.. Note on a method for calculating corrected sums of squares and products. *Technometrics* 1962;**4**(3):419–420.
27. Ling, R.F.. Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association* 1974;**69**(348):859–866.
28. Schubert, E., Gertz, M.. Numerically stable parallel computation of (co-) variance. In: *Proceedings of the 30th international conference on scientific and statistical database management*. 2018, p. 1–12.
29. Lonsdale, J., Thomas, J., Salvatore, M., Phillips, R., Lo, E., Shad, S., et al. The genotype-tissue expression (gtex) project. *Nature genetics* 2013;**45**(6):580–585.
30. Wilks, C., Zheng, S.C., Chen, F.Y., Charles, R., Solomon, B., Ling, J.P., et al. recount3: summaries and queries for large-scale rna-seq expression and splicing. *Genome biology* 2021;**22**(1):323.
31. Riba, A., Oravecz, A., Durik, M., Jiménez, S., Alunni, V., Cerciat, M., et al. Cell cycle gene regulation dynamics revealed by rna velocity and deep-learning. *Nature communications* 2022;**13**(1):2865.
32. Richardson, N., Cook, I., Crane, N., Dunnington, D., François, R., Keane, J., et al. *arrow: Integration to 'Apache Arrow'*; 2024. R package version 14.0.0. Accessed: 2026-03-30; URL https://CRAN.R-project.org/package=arrow.
33. Fedarko, M.W., Martino, C., Morton, J.T., González, A., Rahman, G., Marotz, C.A., et al. Visualizing'omic feature rankings and log-ratios using qurro. *NAR genomics and bioinformatics* 2020;**2**(2):lqaa023.
34. Quinn, T.P.. *Understanding Sequencing Data as Compositions*. Ph.D. thesis; Deakin University; Deakin, Australia; 2019. URL https://hdl.handle.net/10779/DRO/DU:23918562.v1.
35. West, D.H.D.. Updating mean and variance estimates: an improved method. *Communications of the ACM* 1979;**22**(9):532–535.
36. Greenacre, M.. Measuring subcompositional incoherence. *Mathematical Geosciences* 2011;**43**(6):681–693.

Appendix A. Statistical Derivations

A.1. Log Ratio Variance

Logratio Variance (LRV): For each sample k, let $r_{k,ij} = \log(x_{ki}/x_{kj})$ be the log ratio between the counts of gene i and gene j. For the first k samples, define the running mean and centered sum of squares by

$$\mu_{ij}^{(k)} = \frac{1}{k}\sum_{t=1}^{k} r_{t,ij}, \qquad M_{ij}^{(k)} = \sum_{t=1}^{k}(r_{t,ij} - \mu_{ij}^{(k)})^2$$

After n samples, the desired sample variance is

$$V_{ij} = \frac{M_{ij}^{(n)}}{n-1}$$

For completeness, we recall the derivation of Welford's online update for $\mu_{ij}^{(k)}$ and $M_{ij}^{(k)}$ in this notation. Suppose $\mu_{ij}^{(k-1)}$ and $M_{ij}^{(k-1)}$ are known, and we observe a new value $r_{k,ij}$. Define

$$\delta_k = r_{k,ij} - \mu_{ij}^{(k-1)}.$$

Then the mean update follows directly from the definition:

$$\mu_{ij}^{(k)} = \frac{(k-1)\mu_{ij}^{(k-1)} + r_{k,ij}}{k} = \mu_{ij}^{(k-1)} + \frac{\delta_k}{k}$$

Let

$$h_k = \mu_{ij}^{(k)} - \mu_{ij}^{(k-1)} = \frac{\delta_k}{k}$$

Using the definition of $M_{ij}^{(k)}$,

$$\begin{aligned} M_{ij}^{(k)} &= \sum_{t=1}^{k-1}(r_{t,ij} - \mu_{ij}^{(k)})^2 + (r_{k,ij} - \mu_{ij}^{(k)})^2 \\ &= \sum_{t=1}^{k-1}\left[r_{t,ij} - \mu_{ij}^{(k-1)} + \mu_{ij}^{(k-1)} - \mu_{ij}^{(k)}\right]^2 + (r_{k,ij} - \mu_{ij}^{(k)})^2 \\ &= \sum_{t=1}^{k-1}\left[(r_{t,ij} - \mu_{ij}^{(k-1)}) - h_k\right]^2 + (r_{k,ij} - \mu_{ij}^{(k)})^2 \end{aligned}$$

Expanding the first term gives

$$\sum_{t=1}^{k-1}(r_{t,ij} - \mu_{ij}^{(k)})^2 = M_{ij}^{(k-1)} - 2h_k\sum_{t=1}^{k-1}(r_{t,ij} - \mu_{ij}^{(k-1)}) + (k-1)h_k^2$$

Since deviations from the mean sum to zero, $\sum_{t=1}^{k-1}(r_{t,ij} - \mu_{ij}^{(k-1)}) = 0$ so

$$\sum_{t=1}^{k-1}(r_{t,ij} - \mu_{ij}^{(k)})^2 = M_{ij}^{(k-1)} + (k-1)h_k^2 = M_{ij}^{(k-1)} + \frac{k-1}{k^2}\delta_k^2$$

We now consider the second term. For a new sample $r_{k,ij}$,

$$r_{k,ij} - \mu_{ij}^{(k)} = r_{k,ij} - \left(\mu_{ij}^{(k-1)} + \frac{\delta_k}{k}\right) = \frac{k-1}{k}\delta_k$$

Therefore substituting back into our main equation,

$$\begin{aligned} M_{ij}^{(k)} &= M_{ij}^{(k-1)} + \frac{k-1}{k^2}\delta_k^2 + \frac{(k-1)^2}{k^2}\delta_k^2 \\ &= M_{ij}^{(k-1)} + \frac{k-1}{k}\delta_k^2 \\ &= M_{ij}^{(k-1)} + \delta_k(r_{k,ij} - \mu_{ij}^{(k)}) \end{aligned}$$

Thus the online updates are

$$\delta_k = r_{k,ij} - \mu_{ij}^{(k-1)}, \qquad \mu_{ij}^{(k)} = \mu_{ij}^{(k-1)} + \frac{\delta_k}{k}$$

$$M_{ij}^{(k)} = M_{ij}^{(k-1)} + \delta_k(r_{k,ij} - \mu_{ij}^{(k)}), \qquad \text{given that } \mu_{ij}^{(1)} = r_{1,ij}, \quad M_{ij}^{(1)} = 0.$$

Thus after n samples we obtain:

$$V_{ij} = \frac{M_{ij}^{(n)}}{n-1}$$

A.2. α-Transformed Log Ratio Mean

To compute the α-LRM, `propr` transforms the dataset for sample k and gene pairs i, j as $X_{ki} = Y_{ki}^{\alpha}$ and $X_{kj} = Y_{kj}^{\alpha}$. Here, N is the total number of samples in the full dataset, and n is the number of samples in the current group. A group is a set of samples with the same label in the 'group' vector supplied to `propr` for example, all control samples or all knockout samples. We then compute the full-data means for every gene across all N samples:

$$\mu_i^{\text{full}} = \frac{1}{N}\sum_{k=1}^{N} X_{ki}, \qquad \mu_j^{\text{full}} = \frac{1}{N}\sum_{k=1}^{N} X_{kj} \tag{A.1}$$

Then we compute the scaled difference between the means:

$$M_{ij} = \frac{\frac{1}{n}\sum_{k=1}^{n} X_{ki}}{\mu_i^{\text{full}}} - \frac{\frac{1}{n}\sum_{k=1}^{n} X_{kj}}{\mu_j^{\text{full}}} \tag{A.2}$$

After which we compute the group-adjusted difference:

$$C_{ij} = \underbrace{\frac{\sum_{k=1}^{n}(X_{ki} - X_{kj})}{n}}_{\text{Current group}} + \underbrace{\frac{\sum_{k=1}^{N}(X_{ki} - X_{kj}) - \sum_{k=1}^{n}(X_{ki} - X_{kj})}{N - n}}_{\text{Complement group}} \tag{A.3}$$

where the complement group is the set of full-data samples not in the current group. Finally, we compute the LRM:

$$\text{LRM}_{ij} = \frac{1}{\alpha}\left(\frac{C_{ij}}{2} + M_{ij}\right) \tag{A.4}$$

A.2.1. Efficient Two-Pass Computation

We can compute this more efficiently in two passes, making more effective use of our data passes. While this is ideally computable in a single pass—since our subset samples are a subset of the total group—we can use a binary bit vector $(0 \ldots 1, 1, 1, \ldots 0)$ to indicate subgroup membership, allowing a segmented summation in a single pass. However, to maintain backwards compatibility, we strictly maintain the full dataset and sample vectors and perform a two-pass approach.

Pass 1: Full-dataset statistics.

$$\mu_i^{\text{full}} = \frac{1}{N}\sum_{k=1}^{N} X_{ki}, \qquad \mu_j^{\text{full}} = \frac{1}{N}\sum_{k=1}^{N} X_{kj}, \qquad T = \sum_{k=1}^{N}\left(X_{ki} - X_{kj}\right) \tag{A.5}$$

Pass 2: Subgroup statistics.

$$S_i = \sum_{k=1}^{n} X_{ki}, \qquad S_j = \sum_{k=1}^{n} X_{kj}, \qquad D = \sum_{k=1}^{n}\left(X_{ki} - X_{kj}\right) \tag{A.6}$$

Final computation.

$$C_{ij} = \frac{D}{n} + \frac{T - D}{N - n}, \qquad M_{ij} = \frac{1}{n}\left(\frac{S_i}{\mu_i^{\text{full}}} - \frac{S_j}{\mu_j^{\text{full}}}\right), \qquad \text{LRM}_{ij} = \frac{1}{\alpha}\left(\frac{C_{ij}}{2} + M_{ij}\right) \tag{A.7}$$

A.2.2. Complexity Reduction via Per-Gene Intermediates

In a similar manner to how we computed the Log Ratio Mean, with a small reordering we can adopt a two-phase approach wherein the first phase computes per-gene intermediates and the second uses these for pairwise statistics, reducing complexity from $O(G^2(N_{\text{full}} + N_{\text{sub}}))$ to $O(G^2 + G(N_{\text{full}} + N_{\text{sub}}))$.

Notice that T, D, and C can be separated into per-gene components:

$$T_i = \sum_{k=1}^{N} X_{ki}, \qquad T_j = \sum_{k=1}^{N} X_{kj} \tag{A.8}$$

$$D_i = \sum_{k=1}^{n} X_{ki}, \qquad D_j = \sum_{k=1}^{n} X_{kj} \tag{A.9}$$

$$C_i = \frac{D_i}{n} + \frac{T_i - D_i}{N - n}, \qquad C_j = \frac{D_j}{n} + \frac{T_j - D_j}{N - n} \tag{A.10}$$

The LRM can then be expressed as the difference of two per-gene partial LRMs:

$$\widetilde{\text{LRM}}_i = \frac{1}{\alpha}\left(\frac{C_i}{2} + \frac{N}{n}\frac{S_i}{T_i}\right), \qquad \text{LRM}_{ij} = \widetilde{\text{LRM}}_i - \widetilde{\text{LRM}}_j \tag{A.11}$$

Thus, in the first phase we compute $\widetilde{\mathrm{LRM}}_i$ for all genes, and in the second phase we compute LRM_{ij} for all gene pairs at the cost of $O(G)$ space from the first phase.

A.3. Weighted Log Ratio Mean

Due to the dependence on the weights, the weighted variant is much harder to split. If we were to attempt a split, we would still need to materialize a $G \times G \times n$ weight matrix and then perform our LRM computation, giving a computational cost of $O(nG^2)$. Since both the numerator and denominator depend on the product $W_{ki} \cdot W_{kj}$ and are normalized by the sum of the weights, the split is not possible. We therefore opt for a single kernel:

$$\mathrm{LRM}_{ij} = \frac{\sum_{k=1}^{n} W_{ki} \cdot W_{kj} \cdot \log\left(\frac{Y_{ki}}{Y_{kj}}\right)}{\sum_{k=1}^{n} W_{ki} \cdot W_{kj}} \tag{A.12}$$

This computation can be rewritten as two separate matrix multiplies and an element-wise division:

$$\mathrm{LRM} = (M - M^{\top}) \oslash (W^{\top} W), \qquad \text{where} \quad M = (W \odot L)^{\top} W, \quad L = \log(Y) \tag{A.13}$$

This makes use of the property $\log(a/b) = \log(a) - \log(b)$ and could potentially be implemented as a fused matrix-multiply-like kernel.

A.4. α-Transformed Weighted Log Ratio Mean

The α-transformed weighted variant combines the power transformation $X_{ki} = Y_{ki}^{\alpha}$ with per-sample harmonic-mean weights. For each gene pair (i, j), we first define the pairwise weight at sample k as the harmonic mean of the individual gene weights:

$$w_{ij,k} = \frac{2\, w_{ki} \cdot w_{kj}}{w_{ki} + w_{kj}} \tag{A.14}$$

computed separately over the full dataset (N samples) and the subgroup (n samples). Our GPU kernel computation then proceeds in two phases, with each thread assigned to the computation of one gene pair.

Phase 1: Full-dataset statistics. We accumulate statistics over the full dataset (Which we denote by uppercase letters):

$$S_W = \sum_{k=1}^{N} W_{ij,k}, \qquad T_i = \sum_{k=1}^{N} W_{ij,k} \cdot X_{ki}, \qquad T_j = \sum_{k=1}^{N} W_{ij,k} \cdot X_{kj} \tag{A.15}$$

from which we derive the weighted means and the full-set weighted difference:

$$\mu_i = \frac{T_i}{S_W}, \qquad \mu_j = \frac{T_j}{S_W}, \qquad T = T_i - T_j \tag{A.16}$$

Phase 2: Subgroup statistics. We accumulate the corresponding subgroup statistics (which we denote by lowercase letters):

$$s_w = \sum_{k=1}^{n} w_{ij,k}, \qquad d_i = \sum_{k=1}^{n} w_{ij,k} \cdot x_{ki}, \qquad d_j = \sum_{k=1}^{n} w_{ij,k} \cdot x_{kj} \tag{A.17}$$

and set $t = d_i - d_j$.

Final computation. We combine these to form the pairwise α-weighted LRM:

$$C_{ij} = \frac{t}{s_w} + \frac{T - t}{S_W - s_w}, \qquad M_{ij} = \frac{1}{s_w}\left(\frac{d_i}{\mu_i} - \frac{d_j}{\mu_j}\right), \qquad \mathrm{LRM}_{ij} = \frac{1}{\alpha}\left(\frac{C_{ij}}{2} + M_{ij}\right) \tag{A.18}$$

A.5. Weighted Log Ratio Variance

The weighted log ratio variance is defined as:

$$V_{ij} = \frac{\sum_k w_{ij,k}\left(r_{k,ij} - \mu_r^w\right)^2}{\sum_k w_{ij,k} - \frac{\sum_k w_{ij,k}^2}{\sum_k w_{ij,k}}}, \qquad \text{where} \quad r_{k,ij} = \log \frac{x_{ki}}{x_{kj}} \tag{A.19}$$

with the harmonic weight and weighted mean given by:

$$w_{ij,k} = \frac{2\, w_{ki}\, w_{kj}}{w_{ki} + w_{kj}}, \qquad \mu_{ij}^w = \frac{\sum_k w_{ij,k} \log \frac{x_{ki}}{x_{kj}}}{\sum_k w_{ij,k}} \tag{A.20}$$

A.5.1. Online Formulation

Let $r_{k,ij} = \log(x_{ki}/x_{kj})$ and, for notational simplicity, write $w_k = w_{ij,k}$. We introduce the weighted accumulators:

$$S_w = \sum_{k=1}^{n} w_k, \qquad S_{w^2} = \sum_{k=1}^{n} w_k^2, \qquad S_{wr} = \sum_{k=1}^{n} w_k\, r_{k,ij}, \qquad S_{wr^2} = \sum_{k=1}^{n} w_k\, r_{k,ij}^2 \tag{A.21}$$

The weighted mean is $\mu_{ij}^w = S_{wr}/S_w$. Expanding the weighted sum of squares:

$$\sum_{k=1}^{n} w_k (r_{k,ij} - \mu_{ij}^w)^2 = S_{wr^2} - 2\mu_{ij}^w S_{wr} + (\mu_{ij}^w)^2 S_w = S_{wr^2} - \frac{S_{wr}^2}{S_w} \tag{A.22}$$

The denominator of the weighted variance is $S_w - S_{w^2}/S_w = (S_w^2 - S_{w^2})/S_w$, giving:

$$V_{ij} = \frac{S_{wr^2} - S_{wr}^2/S_w}{S_w - S_{w^2}/S_w} = \frac{S_w\left(S_{wr^2} - S_{wr}^2/S_w\right)}{S_w^2 - S_{w^2}} \tag{A.23}$$

A.5.2. West's Online Algorithm

We apply West's algorithm[35] —the weighted analogue of Welford's recurrence—and define the partial accumulators after k samples:

$$S_w^{(k)} = \sum_{t=1}^{k} w_t, \qquad S_{wr}^{(k)} = \sum_{t=1}^{k} w_t\, r_t \tag{A.24}$$

$$S_{wr^2}^{(k)} = \sum_{t=1}^{k} w_t\, r_t^2, \qquad M_{ij}^{(k)} = S_{wr^2}^{(k)} - \frac{(S_{wr}^{(k)})^2}{S_w^{(k)}} \tag{A.25}$$

The increment satisfies:

$$M_{ij}^{(k)} - M_{ij}^{(k-1)} = w_k\, \delta_k \left(r_{k,ij} - \mu_{ij}^{(k)}\right) \tag{A.26}$$

where $\delta_k = r_{k,ij} - \mu_{ij}^{(k-1)}$ and $\mu_{ij}^{(k)} = \mu_{ij}^{(k-1)} + (w_k/S_w^{(k)})\,\delta_k$.

This gives West's recurrence, starting from $\mu_{ij}^{(0)} = 0$, $M_{ij}^{(0)} = 0$, $S_w^{(0)} = 0$, $S_{w^2}^{(0)} = 0$. At each step:

$$S_w^{(k)} = S_w^{(k-1)} + w_k, \qquad S_{w^2}^{(k)} = S_{w^2}^{(k-1)} + w_k^2 \tag{A.27}$$

$$\delta_k = r_{k,ij} - \mu_{ij}^{(k-1)}, \qquad \mu_{ij}^{(k)} = \mu_{ij}^{(k-1)} + \frac{w_k}{S_w^{(k)}}\,\delta_k \tag{A.28}$$

$$M_{ij}^{(k)} = M_{ij}^{(k-1)} + w_k\, \delta_k (r_{k,ij} - \mu_{ij}^{(k)}) \tag{A.29}$$

After processing all n samples, $M_{ij}^{(n)} = S_{wr^2} - S_{wr}^2/S_w$ exactly, without forming S_{wr^2} or S_{wr}^2/S_w separately. The one-pass estimator is:

$$\boxed{V_{ij} = \frac{S_w\, M_{ij}^{(n)}}{S_w^2 - S_{w^2}}} \tag{A.30}$$

A.6. α-Log Ratio Variance

The α log ratio variance is defined as:

$$V_{ij} = \frac{1}{\alpha^2}\,\mathrm{Var}\left(\frac{x_i^\alpha - \mathrm{E}[x_i^\alpha]}{\mathrm{E}[X_i^\alpha]} - \frac{x_j^\alpha - \mathrm{E}[x_j^\alpha]}{\mathrm{E}[X_j^\alpha]}\right) \tag{A.31}$$

$$V_{ij} = \frac{1}{\alpha^2(n-1)}\sum_{k=1}^{n}\left(\frac{x_{ki}^\alpha - \mu_i}{M_i} - \frac{x_{kj}^\alpha - \mu_j}{M_j}\right)^2, \qquad \mu_l = \frac{1}{n}\sum_{k=1}^{n} x_{kl}^\alpha, \quad M_l = \frac{1}{N}\sum_{k=1}^{N} X_{kl}^\alpha \tag{A.32}$$

A.6.1. Derivation

The full-data means M_i and M_j are computed in a preliminary pass. Define the centered and scaled difference for sample k:

$$d_k = \frac{x_{ki}^\alpha - \mu_i}{M_i} - \frac{x_{kj}^\alpha - \mu_j}{M_j} \tag{A.33}$$

Since each term is centered by its own mean, $\sum_{k=1}^{n} d_k = 0$. Expanding the square gives three terms:

$$\sum_{k=1}^{n} d_k^2 = \frac{\mathrm{ss}_i}{M_i^2} + \frac{\mathrm{ss}_j}{M_j^2} - \frac{2\,c_{ij}}{M_i\,M_j} \tag{A.34}$$

where:

$$\mathrm{ss}_i = \sum_{k=1}^{n}(x_{ki}^\alpha - \mu_i)^2, \qquad \mathrm{ss}_j = \sum_{k=1}^{n}(x_{kj}^\alpha - \mu_j)^2, \qquad c_{ij} = \sum_{k=1}^{n}(x_{ki}^\alpha - \mu_i)(x_{kj}^\alpha - \mu_j) \tag{A.35}$$

Applying the centering identity with the accumulators:

$$a_i = \sum_{k=1}^{n}(x_{ki}^\alpha)^2, \qquad a_j = \sum_{k=1}^{n}(x_{kj}^\alpha)^2, \qquad a_{ij} = \sum_{k=1}^{n} x_{ki}^\alpha\, x_{kj}^\alpha \tag{A.36}$$

and noting that $\mu_i = (\sum_k x_{ki}^\alpha)/n$, gives:

$$\mathrm{ss}_i = a_i - n\mu_i^2, \qquad \mathrm{ss}_j = a_j - n\mu_j^2, \qquad c_{ij} = a_{ij} - n\mu_i\mu_j \tag{A.37}$$

A.6.2. Online Computation

For the means μ_i and μ_j, we apply Welford's recurrence independently to the streams $x_{1i}^\alpha, x_{2i}^\alpha, \ldots$ and $x_{1j}^\alpha, x_{2j}^\alpha, \ldots$:

$$\delta_{ki} = x_{ki}^\alpha - \mu_i^{(k-1)}, \qquad \mu_i^{(k)} = \mu_i^{(k-1)} + \frac{\delta_{ki}}{k} \tag{A.38}$$

and likewise for $\mu_j^{(k)}$. For the co-moment c_{ij}, we use the bivariate extension of Welford's recurrence:

$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + \delta_{ki}\left(x_{kj}^\alpha - \mu_j^{(k)}\right) \tag{A.39}$$

with $c_{ij}^{(0)} = 0$. After all n samples, $c_{ij}^{(n)} = a_{ij} - n\mu_i\mu_j$ exactly. The sums of squares a_i and a_j are accumulated as running sums of $(x_{ki}^\alpha)^2$ and $(x_{kj}^\alpha)^2$, and the central sums are recovered at the end via $\mathrm{ss}_i = a_i - n(\mu_i^{(n)})^2$ and $\mathrm{ss}_j = a_j - n(\mu_j^{(n)})^2$.

The one-pass estimator is:

$$\boxed{V_{ij} = \frac{1}{\alpha^2(n-1)}\left(\frac{\mathrm{ss}_i}{M_i^2} + \frac{\mathrm{ss}_j}{M_j^2} - \frac{2\,c_{ij}^{(n)}}{M_i\,M_j}\right)} \tag{A.40}$$

A.7. α-Weighted Log Ratio Variance

The α weighted log ratio variance is:

$$V_{ij} = \frac{1}{\alpha^2}\frac{\sum_k w_{ij,k}\left(d_k - \mu_d^w\right)^2}{\sum_k w_{ij,k} - \frac{\sum_k w_{ij,k}^2}{\sum_k w_{ij,k}}}, \qquad d_k = \frac{x_{ki}^\alpha - \mathrm{E}_w[x_i^\alpha]}{\mathrm{E}_W[X_i^\alpha]} - \frac{x_{kj}^\alpha - \mathrm{E}_w[x_j^\alpha]}{\mathrm{E}_W[X_j^\alpha]} \tag{A.41}$$

equivalently:

$$V_{ij} = \frac{\sum_k w_{ij,k}\left(\frac{x_{ki}^\alpha - \mu_i^w}{M_i^w} - \frac{x_{kj}^\alpha - \mu_j^w}{M_j^w}\right)^2}{\alpha^2\left(\sum_k w_{ij,k} - \frac{\sum_k w_{ij,k}^2}{\sum_k w_{ij,k}}\right)} \tag{A.42}$$

where:

$$w_{ij,k} = \frac{2\,w_{ki}\,w_{kj}}{w_{ki} + w_{kj}}, \qquad \mu_l^w = \frac{\sum_{k=1}^n w_{ij,k}\,x_{kl}^\alpha}{\sum_{k=1}^n w_{ij,k}}, \qquad M_l^w = \frac{\sum_{k=1}^N W_{ij,k}\,X_{kl}^\alpha}{\sum_{k=1}^N W_{ij,k}} \tag{A.43}$$

A.7.1. Derivation

Write $w_k = w_{ij,k}$ for the harmonic weight of sample k. The full-data weighted means M_i^w and M_j^w are computed in a preliminary pass over the full dataset with their own harmonic weights W_k. The centered and scaled difference is:

$$d_k = \frac{x_{ki}^\alpha - \mu_i^w}{M_i^w} - \frac{x_{kj}^\alpha - \mu_j^w}{M_j^w} \tag{A.44}$$

Since the within-group means are weighted means under the same weights, $\sum_k w_k\,d_k = 0$. Expanding the weighted sum of squares:

$$\sum_{k=1}^n w_k\,d_k^2 = \frac{\mathrm{ss}_i^w}{(M_i^w)^2} + \frac{\mathrm{ss}_j^w}{(M_j^w)^2} - \frac{2\,c_{ij}^w}{M_i^w\,M_j^w} \tag{A.45}$$

where the weighted central sums are:

$$\mathrm{ss}_i^w = \sum_{k=1}^n w_k(x_{ki}^\alpha - \mu_i^w)^2 \tag{A.46}$$

$$\mathrm{ss}_j^w = \sum_{k=1}^n w_k(x_{kj}^\alpha - \mu_j^w)^2 \tag{A.47}$$

$$c_{ij}^w = \sum_{k=1}^n w_k(x_{ki}^\alpha - \mu_i^w)(x_{kj}^\alpha - \mu_j^w) \tag{A.48}$$

A.7.2. Online Formulation

We define the following accumulators:

$$s_w = \sum_{k=1}^{n} w_k, \qquad s_{w^2} = \sum_{k=1}^{n} w_k^2 \tag{A.49}$$

$$s_{wx_i} = \sum_{k=1}^{n} w_k\, x_{ki}^{\alpha}, \qquad s_{wx_j} = \sum_{k=1}^{n} w_k\, x_{kj}^{\alpha} \tag{A.50}$$

$$s_{wx_i^2} = \sum_{k=1}^{n} w_k (x_{ki}^{\alpha})^2, \qquad s_{wx_j^2} = \sum_{k=1}^{n} w_k (x_{kj}^{\alpha})^2 \tag{A.51}$$

$$s_{wx_ix_j} = \sum_{k=1}^{n} w_k\, x_{ki}^{\alpha}\, x_{kj}^{\alpha} \tag{A.52}$$

Noting that $\mu_i^w = s_{wx_i}/s_w$, the weighted centering identity gives:

$$\mathrm{ss}_i^w = s_{wx_i^2} - \frac{s_{wx_i}^2}{s_w}, \qquad \mathrm{ss}_j^w = s_{wx_j^2} - \frac{s_{wx_j}^2}{s_w}, \qquad c_{ij}^w = s_{wx_ix_j} - \frac{s_{wx_i}\, s_{wx_j}}{s_w} \tag{A.53}$$

The denominator is the same reliability-weights term from the weighted LRV, $(s_w^2 - s_{w^2})/s_w$. All seven accumulators are updated in a single pass, and the central sums are recovered at the end. The one-pass estimator is:

$$\boxed{V_{ij} = \frac{s_w}{\alpha^2(s_w^2 - s_{w2})}\left(\frac{\mathrm{ss}_i^w}{(M_i^w)^2} + \frac{\mathrm{ss}_j^w}{(M_j^w)^2} - \frac{2\, c_{ij}^w}{M_i^w\, M_j^w}\right)} \tag{A.54}$$

Performing the computation in a streaming manner allows us to compute our required statistics in a single pass over the input, with at most two passes—one over the full dataset and another over the subset samples. The second pass could have been eliminated entirely, but was kept for backwards compatibility so as not to break the API for existing users.

A.8. Variation of Log Ratios (VLR)

The VLR between genes i and j is defined as:

$$\mathrm{VLR}_{ij} = \mathrm{Var}\left(\log \frac{x_{ki}}{x_{kj}}\right) = \frac{1}{n-1}\sum_{k=1}^{n}\left(\log \frac{x_{ki}}{x_{kj}} - \mu_{ij}\right)^2, \qquad \mu_{ij} = \frac{1}{n}\sum_{k=1}^{n} \log \frac{x_{ki}}{x_{kj}} \tag{A.55}$$

This is mathematically identical to the basic LRV case. The GPU implementation recognizes this and implements it as such. The CPU implementation computes VLR indirectly using the identity $\mathrm{Var}(a-b) = \mathrm{Var}(a) + \mathrm{Var}(b) - 2\,\mathrm{Cov}(a,b)$. It first materializes the covariance matrix $(G \times G)$ and extracts the diagonal values to compute $\sigma_{ii} + \sigma_{jj} - 2\sigma_{ij}$. It is important to note that both still perform the same amount of asymptotic work of $\Theta(ng^2)$

Let $l_{ki} = \log x_{ki}$ and $\mu_i^l = \frac{1}{n}\sum_k l_{ki}$. Expanding:

$$\begin{aligned}
\mathrm{VLR}_{ij} &= \mathrm{Var}(l_{*i} - l_{*j}) \\
&= \frac{1}{n-1}\sum_{k=1}^{n}[(l_{ki} - \mu_i^l) - (l_{kj} - \mu_j^l)]^2 \\
&= \frac{1}{n-1}\sum_{k=1}^{n}(l_{ki} - \mu_i^l)^2 + \frac{1}{n-1}\sum_{k=1}^{n}(l_{kj} - \mu_j^l)^2 - \frac{2}{n-1}\sum_{k=1}^{n}(l_{ki} - \mu_i^l)(l_{kj} - \mu_j^l) \\
&= \mathrm{Var}(l_{*i}) + \mathrm{Var}(l_{*j}) - 2\,\mathrm{Cov}(l_{*i}, l_{*j})
\end{aligned} \tag{A.56}$$

A.9. Centered Log-Ratio (CLR) Transform

Let G denote the total number of genes. For sample k and gene j, the CLR transform is:

$$\mathrm{CLR}_{kj} = \log(x_{kj}) - \frac{1}{G}\sum_{g=1}^{G}\log(x_{kg}) = \log(x_{kj}) - \mu_k, \qquad \mu_k = \frac{1}{G}\sum_{g=1}^{G} l_{kg} \tag{A.57}$$

Because the centering is applied independently within each sample, we first perform a row-wise reduction to compute the sample mean μ_k and then subtract the result from every log-transformed value in the corresponding row.

A.10. Proportionality Coefficient-ϕ

The proportionality coefficient ϕ between genes i and j is:

$$\phi_{ij} = \frac{\mathrm{VLR}_{ij}}{v_j}, \qquad v_j = \mathrm{Var}(\mathrm{CLR}_{*j}) \tag{A.58}$$

A value near zero indicates strong proportionality. `propr` implements two variants: a symmetric one and a non-symmetric one. When the symmetric variant is requested, we reproduce the lower triangle values into the upper triangle. The diagonal is always 0.

`propr` computes this via stages in the CPU backend: first the full VLR matrix, then the CLR matrix, and finally the per-gene CLR variance v_j to divide VLR_{ij}. However, we can avoid materializing the CLR matrix entirely by recovering the CLR variances directly from the VLR matrix.

A.10.1. Recovering CLR Variance from the VLR Matrix

Recall from the VLR derivation (A.56) that:

$$\mathrm{VLR}_{ij} = \sigma_{ii} + \sigma_{jj} - 2\sigma_{ij} \tag{A.59}$$

where $\sigma_{ij} = \mathrm{Cov}(l_{*i}, l_{*j})$ are entries of the log-covariance matrix Σ with $l_{ki} = \log X_{ki}$.

Row sum of the VLR matrix. Define the row sum for gene i:

$$R_i \overset{\text{def}}{=} \sum_{j=1}^{G}\mathrm{VLR}_{ij} = \sum_{j=1}^{G}\left(\sigma_{ii} + \sigma_{jj} - 2\sigma_{ij}\right) = G\,\sigma_{ii} + \mathrm{tr}(\Sigma) - 2\sum_{j=1}^{G}\sigma_{ij} \tag{A.60}$$

Dividing by G and rearranging:

$$\sigma_{ii} - \frac{2}{G}\sum_{j=1}^{G}\sigma_{ij} = \frac{R_i}{G} - \frac{\mathrm{tr}(\Sigma)}{G} \tag{A.61}$$

Expanding the CLR variance. Since $\mathrm{CLR}_{ki} = l_{ki} - \frac{1}{G}\sum_{g=1}^{G} l_{kg}$, the CLR variance for gene i is:

$$v_i = \mathrm{Var}\left(l_{*i} - \frac{1}{G}\sum_{g=1}^{G} l_{*g}\right) \tag{A.62}$$

Using $\mathrm{Var}(a-b) = \mathrm{Var}(a) + \mathrm{Var}(b) - 2\,\mathrm{Cov}(a,b)$ with $a = l_{*i}$ and $b = \frac{1}{G}\sum_{g=1}^{G} l_{*g}$:

$$\begin{aligned} v_i &= \sigma_{ii} + \frac{1}{G^2}\mathrm{Var}\left(\sum_{g=1}^{G} l_{*g}\right) - 2\,\mathrm{Cov}\left(l_{*i}, \frac{1}{G}\sum_{g=1}^{G} l_{*g}\right) \\ &= \sigma_{ii} + \frac{1}{G^2}\sum_{g=1}^{G}\sum_{h=1}^{G}\sigma_{gh} - \frac{2}{G}\sum_{g=1}^{G}\sigma_{ig} \end{aligned} \tag{A.63}$$

where we used the linearity of covariance: $\mathrm{Cov}(A, B+C) = \mathrm{Cov}(A,B) + \mathrm{Cov}(A,C)$ and factoring out $\frac{1}{G}$.

Reordering and substituting the row-sum identity (A.61):

$$v_i = \underbrace{\sigma_{ii} - \frac{2}{G}\sum_{g=1}^{G}\sigma_{ig}}_{\text{from (A.61)}} + \frac{1}{G^2}\sum_{g=1}^{G}\sum_{h=1}^{G}\sigma_{gh} = \frac{R_i}{G} - \frac{\operatorname{tr}(\Sigma)}{G} + \frac{1}{G^2}\sum_{g,h}\sigma_{gh} \tag{A.64}$$

Eliminating the covariance matrix. To avoid materializing Σ, we eliminate $\operatorname{tr}(\Sigma)$ and $\sum_{g,h}\sigma_{gh}$. Define the total sum of the VLR matrix:

$$T \overset{\text{def}}{=} \sum_{j=1}^{G}\sum_{l=1}^{G}\mathrm{VLR}_{jl} = \sum_{j,l}\left(\sigma_{jj} + \sigma_{ll} - 2\sigma_{jl}\right) = 2G\operatorname{tr}(\Sigma) - 2\sum_{j,l}\sigma_{jl} \tag{A.65}$$

Solving for $\sum_{j,l}\sigma_{jl}$ and scaling:

$$\frac{1}{G^2}\sum_{j,l}\sigma_{jl} = \frac{\operatorname{tr}(\Sigma)}{G} - \frac{T}{2G^2} \tag{A.66}$$

Substituting into (A.64):

$$v_i = \frac{R_i}{G} - \frac{\operatorname{tr}(\Sigma)}{G} + \frac{\operatorname{tr}(\Sigma)}{G} - \frac{T}{2G^2} \tag{A.67}$$

$$\boxed{v_i = \frac{R_i}{G} - \frac{T}{2G^2}} \tag{A.68}$$

where $R_i = \sum_{j=1}^{G}\mathrm{VLR}_{ij}$ is the i-th row sum and $T = \sum_{j,l}\mathrm{VLR}_{jl}$ is the total sum of the VLR matrix. This allows us to recover the CLR variance from two reductions over VLR without ever materializing the CLR matrix.

A.11. Online Covariance

The sample covariance between genes i and j is:

$$\mathrm{Cov}_{ij} = \frac{1}{n-1+d}\sum_{k=1}^{n}(x_{ki}-\mu_i)(x_{kj}-\mu_j), \qquad \mu_i = \frac{1}{n}\sum_{k=1}^{n}x_{ki}, \quad \mu_j = \frac{1}{n}\sum_{k=1}^{n}x_{kj} \tag{A.69}$$

where $d = \text{norm\_type} \in \{0, 1\}$ selects between dividing by $n-1$ (sample) or n (population).

A.11.1. Centering Identity

Applying the centering identity to the numerator:

$$\sum_{k=1}^{n}(x_{ki}-\mu_i)(x_{kj}-\mu_j) = \sum_{k=1}^{n}x_{ki}\,x_{kj} - n\mu_i\mu_j \tag{A.70}$$

A.11.2. Welford's Bivariate Recurrence

We introduce the partial accumulators:

$$S_1^{(k)} = \sum_{t=1}^{k}x_{ti}\,x_{tj}, \qquad \mu_i^{(k)} = \frac{1}{k}\sum_{t=1}^{k}x_{ti}, \qquad \mu_j^{(k)} = \frac{1}{k}\sum_{t=1}^{k}x_{tj} \tag{A.71}$$

and the partial co-moment:

$$C_{ij}^{(k)} = S_1^{(k)} - k\,\mu_i^{(k)}\mu_j^{(k)} \tag{A.72}$$

The difference between steps $k-1$ and k is:

$$C_{ij}^{(k)} - C_{ij}^{(k-1)} = x_{ki}\,x_{kj} - k\,\mu_i^{(k)}\mu_j^{(k)} + (k-1)\,\mu_i^{(k-1)}\mu_j^{(k-1)} \tag{A.73}$$

Writing $\delta_{ki} = x_{ki} - \mu_i^{(k-1)}$, we have $\mu_i^{(k)} = \mu_i^{(k-1)} + \delta_{ki}/k$ (and similarly for j). Expanding the product $k\,\mu_i^{(k)}\mu_j^{(k)}$:

$$\begin{aligned} k\,\mu_i^{(k)}\mu_j^{(k)} &= k\left(\mu_i^{(k-1)} + \frac{\delta_{ki}}{k}\right)\left(\mu_j^{(k-1)} + \frac{\delta_{kj}}{k}\right) \\ &= k\,\mu_i^{(k-1)}\mu_j^{(k-1)} + \mu_j^{(k-1)}\delta_{ki} + \mu_i^{(k-1)}\delta_{kj} + \frac{\delta_{ki}\delta_{kj}}{k} \end{aligned} \tag{A.74}$$

Substituting back and using $x_{ki} = \mu_i^{(k-1)} + \delta_{ki}$, $x_{kj} = \mu_j^{(k-1)} + \delta_{kj}$, then expanding and collecting terms:

$$C_{ij}^{(k)} - C_{ij}^{(k-1)} = \delta_{ki}\,\delta_{kj} - \frac{\delta_{ki}\,\delta_{kj}}{k} = \delta_{ki}\,\delta_{kj}\,\frac{k-1}{k} \tag{A.75}$$

Noting that $x_{kj} - \mu_j^{(k)} = \delta_{kj} - \delta_{kj}/k = \delta_{kj}(k-1)/k$, this simplifies to:

$$C_{ij}^{(k)} - C_{ij}^{(k-1)} = \delta_{ki}(x_{kj} - \mu_j^{(k)}) \tag{A.76}$$

A.11.3. Update Rules

Starting from $\mu_i^{(0)} = \mu_j^{(0)} = 0$, $C_{ij}^{(0)} = 0$, at each step:

$$\delta_{ki} = x_{ki} - \mu_i^{(k-1)} \tag{A.77}$$

$$\mu_i^{(k)} = \mu_i^{(k-1)} + \frac{\delta_{ki}}{k} \tag{A.78}$$

$$\mu_j^{(k)} = \mu_j^{(k-1)} + \frac{x_{kj} - \mu_j^{(k-1)}}{k} \tag{A.79}$$

$$C_{ij}^{(k)} = C_{ij}^{(k-1)} + \delta_{ki}(x_{kj} - \mu_j^{(k)}) \tag{A.80}$$

It is important that μ_j is updated *before* the co-moment so that the new mean $\mu_j^{(k)}$ appears in the second factor. After all n samples, $C_{ij}^{(n)} = \sum_k x_{ki}x_{kj} - n\mu_i\mu_j$ exactly, without ever forming S_1 or $n\mu_i\mu_j$ separately. The one-pass estimator is:

$$\boxed{\mathrm{Cov}_{ij} = \frac{C_{ij}^{(n)}}{n - 1 + d}} \tag{A.81}$$

A.12. Online Pearson Correlation

The Pearson correlation between genes i and j is:

$$r_{ij} = \frac{\sum_{k=1}^{n}(x_{ki} - \mu_i)(x_{kj} - \mu_j)}{\sqrt{\sum_{k=1}^{n}(x_{ki} - \mu_i)^2 \;\cdot\; \sum_{k=1}^{n}(x_{kj} - \mu_j)^2}}, \qquad \mu_* = \frac{1}{n}\sum_{k=1}^{n} x_{k*} \tag{A.82}$$

The numerator is exactly the co-moment $C_{ij}^{(n)}$ derived in Section A.11. The denominator has two factors—the univariate sums of centered squares:

$$S_i = \sum_{k=1}^{n}(x_{ki} - \mu_i)^2, \qquad S_j = \sum_{k=1}^{n}(x_{kj} - \mu_j)^2 \tag{A.83}$$

As shown in the LRV derivation, applying the centering identity and defining the partial accumulator $S_i^{(k)} = \sum_{t=1}^{k} x_{ti}^2 - k(\mu_i^{(k)})^2$, the difference simplifies to:

$$S_i^{(k)} - S_i^{(k-1)} = \delta_{ki}(x_{ki} - \mu_i^{(k)}) \tag{A.84}$$

where $\delta_{ki} = x_{ki} - \mu_i^{(k-1)}$ and $\mu_i^{(k)} = \mu_i^{(k-1)} + \delta_{ki}/k$.

Since all three recurrences (C_{ij}, S_i, S_j) share the same delta values and running means, we maintain five accumulators (μ_i, μ_j, S_i, S_j, C_{ij}) in a single pass. The combined update at sample k is:

$$\delta_{ki} = x_{ki} - \mu_i^{(k-1)}, \qquad \mu_i^{(k)} = \mu_i^{(k-1)} + \frac{\delta_{ki}}{k}, \qquad S_i^{(k)} = S_i^{(k-1)} + \delta_{ki}(x_{ki} - \mu_i^{(k)}) \tag{A.85}$$

$$\delta_{kj} = x_{kj} - \mu_j^{(k-1)}, \qquad \mu_j^{(k)} = \mu_j^{(k-1)} + \frac{\delta_{kj}}{k}, \qquad S_j^{(k)} = S_j^{(k-1)} + \delta_{kj}(x_{kj} - \mu_j^{(k)}) \tag{A.86}$$

$$C_{ij}^{(k)} = C_{ij}^{(k-1)} + \delta_{ki}(x_{kj} - \mu_j^{(k)}) \tag{A.87}$$

After processing all n samples, $S_i^{(n)} = \sum_k x_{ki}^2 - n\mu_i^2$, $S_j^{(n)} = \sum_k x_{kj}^2 - n\mu_j^2$, and $C_{ij}^{(n)} = \sum_k x_{ki}x_{kj} - n\mu_i\mu_j$, without ever forming any of these products separately. The one-pass estimator is:

$$\boxed{r_{ij} = \frac{C_{ij}^{(n)}}{\sqrt{S_i^{(n)}} \cdot \sqrt{S_j^{(n)}}}} \tag{A.88}$$

A.13. Fisher Z-Transform and Variance of Lin's Concordance Correlation Coefficient (CCC)

Given a pre-computed concordance correlation matrix ρ_c and the Pearson correlation matrix r (both $G \times G$), `propr` computes the Fisher z-transform and its associated variance for hypothesis testing. The GPU kernel `linRcpp` fuses the online Pearson correlation (Section A.12) with the z-transform and variance computation into a single pass over the data, producing a single output matrix with the Fisher Z-transform stored in lower triangle and the variance of the Z-Transform in the upper triangle. The output matrix Z is defined element-wise as:

$$Z_{ij} = \begin{cases} 1 & \text{if } i = j \\ \operatorname{atanh}(\rho_c^{ij}) & \text{if } i > j \quad \text{(lower triangle)} \\ \dfrac{(1 - r_{ij}^2)\,(\rho_c^{ij})^2}{(1 - (\rho_c^{ij})^2)\, r_{ij}^2\,(K-2)} & \text{if } i < j \quad \text{(upper triangle)} \end{cases} \tag{A.89}$$

where K is the number of samples.

A.13.1. Lower Triangle: Fisher Z-Transform

The Fisher z-transform maps the CCC from $(-1, 1)$ to the real line, stabilizing the variance for inference:

$$z_{ij} = \operatorname{atanh}(\rho_c^{ij}) = \frac{1}{2} \ln\left(\frac{1 + \rho_c^{ij}}{1 - \rho_c^{ij}}\right) \tag{A.90}$$

In the GPU implementation, the CCC is clamped to $[-1 + \epsilon,\ 1 - \epsilon]$ with $\epsilon = 10^{-7}$ before applying the transform to avoid numerical singularities at ± 1.

A.13.2. Upper Triangle: Variance of the Z-Transform

The upper triangle stores the variance of the Fisher z-transform, which quantifies the uncertainty of z_{ij} under the assumption that the true concordance is ρ_c^{ij}:

$$\operatorname{Var}(z_{ij}) = \frac{(1 - r_{ij}^2)\,(\rho_c^{ij})^2}{(1 - (\rho_c^{ij})^2)\, r_{ij}^2\,(K-2)} \tag{A.91}$$

where r_{ij} is the Pearson correlation between genes i and j. This variance estimate allows the construction of confidence intervals for the CCC via $z_{ij} \pm z_{\alpha/2}\sqrt{\operatorname{Var}(z_{ij})}$, which can be back-transformed to the CCC scale using $\tanh(\cdot)$.

A.13.3. GPU Implementation

The kernel computes the Pearson correlation r_{ij} online via the five-accumulator Welford recurrence described in Section A.12 ($\mu_i, \mu_j, S_i, S_j, C_{ij}$), fused into a tiled matrix-multiply-like structure. The pre-computed CCC matrix ρ_c is read from global memory. After the main accumulation loop completes, the epilogue computes the Pearson correlation as:

$$r_{ij} = C_{ij}^{(K)} \cdot \frac{1}{\sqrt{S_i^{(K)}}} \cdot \frac{1}{\sqrt{S_j^{(K)}}} \tag{A.92}$$

clamped to $[-1, 1]$, and then branches on the triangle position to emit either $\operatorname{atanh}(\rho_c^{ij})$ or $\operatorname{Var}(z_{ij})$. The diagonal is set to 1.

Appendix B. ANOVA-like decomposition of χ^2 distance for sparse data

While only the logarithmic function provides full scale-freeness of the approach, the use of an approximation based on the Box-Cox transformation provides a useful trade-off between avoidance of zero-imputations and maintaining the analysis approximately scale-free. One can approximate the logarithm in the presence of zeros making use of the limit $\log(x) = \lim_{\alpha\to 0} \frac{x^\alpha - 1}{\alpha}$ by using a finite approximation for a small user-defined α. In this approximation, log-ratio variance can be written in form of a (squared) χ^2-distance that approaches LRV for $\alpha \to 0$. A simple re-arrangement of terms lets LRV take the form

$$\mathrm{LRV}_{ij} = \frac{1}{N}\sum_{i=1}^{N}\left(\log\frac{Y_{ki}}{\prod_{k=1}^{N} Y_{ki}^{1/N}} - \log\frac{Y_{kj}}{\prod_{k=1}^{N} Y_{kj}^{1/N}}\right)^2. \tag{B.1}$$

Note that the χ^2 distance used in Correspondence Analysis, when weighting it uniformly, can be written as

$$\chi^2(\mathbf{y}_i, \mathbf{y}_j) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{Y_{ki}}{\frac{1}{N}\sum_{k=1}^{N} Y_{ki}} - \frac{Y_{kj}}{\frac{1}{N}\sum_{k=1}^{N} Y_{kj}}\right)^2. \tag{B.2}$$

This similarity between LRV and χ^2-distance becomes an equality in the limit[36]

$$\lim_{\alpha\to 0}\frac{1}{\alpha^2}\chi^2(\mathbf{y}_i^\alpha, \mathbf{y}_j^\alpha) = \mathrm{LRV}_{ij}. \tag{B.3}$$

To obtain a logarithm-free distance measure on a subgroup of samples with size n, we need an additional centering of the variables within the group. Define

$$d_{ij}^1(\mathbf{y}_i^\alpha, \mathbf{y}_j^\alpha) = \frac{1}{n\alpha^2}\sum_{k=1}^{n}\left(\frac{Y_{ki}^\alpha - \frac{1}{n}\sum_{l=1}^{n} Y_{li}^\alpha}{\frac{1}{N}\sum_{m=1}^{N} Y_{mi}^\alpha} - \frac{Y_{kj}^\alpha - \frac{1}{n}\sum_{l=1}^{n} Y_{lj}^\alpha}{\frac{1}{N}\sum_{m=1}^{N} Y_{mj}^\alpha}\right)^2, \tag{B.4}$$

and similarly for d_{ij}^2, where sample indices are from the second group and n is replaced by $N - n$. Note that, if we define the same distance on all the samples, we obtain the χ^2-distance, for which, in full analogy to (3), the following decomposition holds:

$$\chi^2(\mathbf{y}_i^\alpha, \mathbf{y}_j^\alpha) = \frac{n d_{ij}^1 + (N-n) d_{ij}^2}{N} + \frac{n(N-n)}{N^2}\left(m_{ij}^1 - m_{ij}^2\right)^2, \tag{B.5}$$

with

$$m_{ij}^1(\mathbf{y}_i^\alpha, \mathbf{y}_j^\alpha) = \frac{1}{N\alpha}\sum_{k=1}^{N}\left(\frac{Y_{ki}^\alpha}{\frac{1}{N}\sum_{m=1}^{N} Y_{mi}^\alpha} - \frac{Y_{kj}^\alpha}{\frac{1}{N}\sum_{m=1}^{N} Y_{mj}^\alpha}\right), \tag{B.6}$$

and similarly for m_{ij}^2. However, if we want the analogous limit to hold for the logratio mean as well, we cannot use (B.6) as a definition. Note that we always have $m_{ij}^1 + m_{ij}^2 = 0$. To avoid this artifact, here we use a corrected version

that approximates the sum over the logratio means by

$$C_{ij} = \frac{1}{n\alpha}\sum_{k=1}^{n}(Y_{ki}^{\alpha} - Y_{kj}^{\alpha}) + \frac{1}{(N-n)\alpha}\sum_{k=n+1}^{N}(Y_{ki}^{\alpha} - Y_{kj}^{\alpha}). \tag{B.7}$$

The corrected version for the α-transformed logratio mean of the first group is then obtained as

$$\mathrm{LRM}_{ij}^{1\alpha} = \frac{C_{ij}}{2} + m_{ij}^{1}, \tag{B.8}$$

with m_{ij}^{1} replaced by m_{ij}^{2} for the second group. Note that the difference between the means and thus the decomposition (B.5) is not affected.

Appendix C. Numerical Accuracy

G	N_1 vs N_2	n_{pairs}	$\max \lvert \text{lrv}_{\text{gpu}} - \text{lrv}_{\text{cpu}} \rvert$	$\max \lvert \text{lrv}_{1,\text{gpu}} - \text{lrv}_{1,\text{cpu}} \rvert$	$\max \lvert \text{lrv}_{2,\text{gpu}} - \text{lrv}_{2,\text{cpu}} \rvert$	$\max \lvert \text{lrm}_{\text{gpu}} - \text{lrm}_{\text{cpu}} \rvert$	$\max \lvert \text{lrm}_{1,\text{gpu}} - \text{lrm}_{1,\text{cpu}} \rvert$	$\max \lvert \text{lrm}_{2,\text{gpu}} - \text{lrm}_{2,\text{cpu}} \rvert$	$\max \lvert \theta_{\text{gpu}} - \theta_{\text{cpu}} \rvert$	$\max \lvert \theta_{e,\text{gpu}} - \theta_{e,\text{cpu}} \rvert$	$\max \lvert \theta_{f,\text{gpu}} - \theta_{f,\text{cpu}} \rvert$	$\max \lvert \theta_{g,\text{gpu}} - \theta_{g,\text{cpu}} \rvert$	$\max \lvert \theta_{\text{mod,gpu}} - \theta_{\text{mod,cpu}} \rvert$
500	100vs100	124750	5.720×10^{-5}	3.691×10^{-5}	5.828×10^{-5}	9.625×10^{-7}	8.891×10^{-7}	7.970×10^{-7}	1.852×10^{-4}	1.270×10^{-4}	1.270×10^{-4}	1.184×10^{-4}	1.335×10^{-4}
500	200vs200	124750	1.114×10^{-4}	4.619×10^{-5}	8.842×10^{-5}	1.555×10^{-6}	1.028×10^{-6}	9.993×10^{-7}	4.988×10^{-4}	4.728×10^{-4}	4.728×10^{-4}	1.962×10^{-4}	3.398×10^{-4}
500	500vs500	124750	1.231×10^{-4}	1.068×10^{-4}	9.691×10^{-5}	1.675×10^{-6}	1.674×10^{-6}	1.254×10^{-6}	2.839×10^{-4}	1.595×10^{-4}	1.595×10^{-4}	1.439×10^{-4}	2.650×10^{-4}
500	1000vs1000	124750	1.599×10^{-4}	1.288×10^{-4}	1.250×10^{-4}	2.880×10^{-6}	1.600×10^{-6}	1.752×10^{-6}	3.914×10^{-4}	2.897×10^{-4}	2.897×10^{-4}	2.056×10^{-4}	3.750×10^{-4}
500	2259vs2240	124750	2.530×10^{-4}	1.820×10^{-4}	1.886×10^{-4}	4.707×10^{-6}	3.345×10^{-6}	3.196×10^{-6}	6.758×10^{-4}	4.194×10^{-4}	4.194×10^{-4}	3.830×10^{-4}	6.644×10^{-4}
1000	100vs100	499500	7.733×10^{-5}	5.292×10^{-5}	5.828×10^{-5}	1.078×10^{-6}	9.669×10^{-7}	8.386×10^{-7}	4.233×10^{-4}	2.669×10^{-4}	2.669×10^{-4}	1.640×10^{-4}	2.342×10^{-4}
1000	200vs200	499500	1.114×10^{-4}	6.301×10^{-5}	8.842×10^{-5}	1.555×10^{-6}	1.160×10^{-6}	1.021×10^{-6}	4.988×10^{-4}	4.728×10^{-4}	4.728×10^{-4}	2.552×10^{-4}	3.549×10^{-4}
1000	500vs500	499500	1.315×10^{-4}	1.068×10^{-4}	1.103×10^{-4}	2.227×10^{-6}	1.827×10^{-6}	1.415×10^{-6}	3.440×10^{-4}	1.841×10^{-4}	1.841×10^{-4}	1.599×10^{-4}	3.184×10^{-4}
1000	1000vs1000	499500	2.086×10^{-4}	1.328×10^{-4}	1.525×10^{-4}	2.880×10^{-6}	2.966×10^{-6}	2.023×10^{-6}	5.221×10^{-4}	3.780×10^{-4}	3.780×10^{-4}	3.340×10^{-4}	5.019×10^{-4}
1000	2259vs2240	499500	2.842×10^{-4}	2.339×10^{-4}	2.073×10^{-4}	4.726×10^{-6}	3.511×10^{-6}	3.313×10^{-6}	7.175×10^{-4}	5.030×10^{-4}	5.030×10^{-4}	4.529×10^{-4}	7.059×10^{-4}
3000	100vs100	4498500	1.028×10^{-4}	5.444×10^{-5}	6.539×10^{-5}	1.625×10^{-6}	1.523×10^{-6}	1.544×10^{-6}	4.986×10^{-4}	5.095×10^{-4}	5.095×10^{-4}	2.119×10^{-4}	3.576×10^{-4}
3000	200vs200	4498500	1.312×10^{-4}	8.490×10^{-5}	8.842×10^{-5}	1.555×10^{-6}	1.406×10^{-6}	1.250×10^{-6}	5.612×10^{-4}	4.728×10^{-4}	4.728×10^{-4}	3.776×10^{-4}	4.930×10^{-4}
3000	500vs500	4498500	2.322×10^{-4}	1.161×10^{-4}	1.148×10^{-4}	2.773×10^{-6}	1.827×10^{-6}	2.085×10^{-6}	6.484×10^{-4}	4.051×10^{-4}	4.051×10^{-4}	3.643×10^{-4}	6.323×10^{-4}
3000	1000vs1000	4498500	2.469×10^{-4}	1.903×10^{-4}	1.829×10^{-4}	2.907×10^{-6}	2.988×10^{-6}	2.584×10^{-6}	5.300×10^{-4}	4.216×10^{-4}	4.216×10^{-4}	3.340×10^{-4}	5.250×10^{-4}
3000	2259vs2240	4498500	4.162×10^{-4}	2.570×10^{-4}	2.572×10^{-4}	5.018×10^{-6}	3.529×10^{-6}	3.800×10^{-6}	8.163×10^{-4}	5.030×10^{-4}	5.030×10^{-4}	4.786×10^{-4}	8.125×10^{-4}
6000	100vs100	17997000	1.168×10^{-4}	7.300×10^{-5}	7.350×10^{-5}	1.761×10^{-6}	1.548×10^{-6}	1.584×10^{-6}	5.564×10^{-4}	5.095×10^{-4}	5.095×10^{-4}	3.581×10^{-4}	4.171×10^{-4}
6000	200vs200	17997000	1.662×10^{-4}	9.555×10^{-5}	9.732×10^{-5}	1.741×10^{-6}	1.516×10^{-6}	1.358×10^{-6}	7.581×10^{-4}	5.282×10^{-4}	5.282×10^{-4}	3.776×10^{-4}	6.832×10^{-4}
6000	500vs500	17997000	2.322×10^{-4}	1.245×10^{-4}	1.387×10^{-4}	2.773×10^{-6}	2.681×10^{-6}	2.085×10^{-6}	7.221×10^{-4}	4.838×10^{-4}	4.838×10^{-4}	4.033×10^{-4}	7.049×10^{-4}
6000	1000vs1000	17997000	3.120×10^{-4}	2.092×10^{-4}	1.968×10^{-4}	3.482×10^{-6}	2.988×10^{-6}	2.600×10^{-6}	9.414×10^{-4}	5.886×10^{-4}	5.886×10^{-4}	4.803×10^{-4}	9.323×10^{-4}
6000	2259vs2240	17997000	4.162×10^{-4}	3.141×10^{-4}	3.933×10^{-4}	8.973×10^{-6}	3.707×10^{-6}	3.800×10^{-6}	1.139×10^{-3}	8.004×10^{-4}	8.004×10^{-4}	6.051×10^{-4}	1.135×10^{-3}
12238	100vs100	74878203	1.168×10^{-4}	7.716×10^{-5}	8.331×10^{-5}	1.800×10^{-6}	1.872×10^{-6}	1.584×10^{-6}	8.695×10^{-4}	9.275×10^{-4}	9.275×10^{-4}	3.630×10^{-4}	6.552×10^{-4}
12238	200vs200	74878203	1.662×10^{-4}	1.054×10^{-4}	1.055×10^{-4}	2.000×10^{-6}	1.609×10^{-6}	1.710×10^{-6}	7.581×10^{-4}	5.282×10^{-4}	5.282×10^{-4}	3.776×10^{-4}	6.857×10^{-4}
12238	500vs500	74878203	2.322×10^{-4}	1.632×10^{-4}	1.417×10^{-4}	3.355×10^{-6}	3.105×10^{-6}	2.256×10^{-6}	7.518×10^{-4}	5.656×10^{-4}	5.656×10^{-4}	4.033×10^{-4}	7.325×10^{-4}
12238	1000vs1000	74878203	3.152×10^{-4}	2.191×10^{-4}	2.079×10^{-4}	3.968×10^{-6}	2.988×10^{-6}	3.773×10^{-6}	9.414×10^{-4}	5.886×10^{-4}	5.886×10^{-4}	4.803×10^{-4}	9.328×10^{-4}
12238	2259vs2240	74878203	5.934×10^{-4}	3.326×10^{-4}	3.959×10^{-4}	1.040×10^{-5}	4.354×10^{-6}	5.258×10^{-6}	1.261×10^{-3}	9.237×10^{-4}	9.237×10^{-4}	6.317×10^{-4}	1.257×10^{-3}

Table C.1. Mouse Stem cell dataset: maximum absolute CPU/GPU differences. Each entry is $\max |x_{\text{gpu}} - x_{\text{cpu}}|$ over all matched gene pairs (Partner, Pair) in that run configuration.

G	N_1 vs N_2	n_{pairs}	$\max \lvert \text{lrv}_{\text{gpu}} - \text{lrv}_{\text{cpu}} \rvert$	$\max \lvert \text{lrv}_{1,\text{gpu}} - \text{lrv}_{1,\text{cpu}} \rvert$	$\max \lvert \text{lrv}_{2,\text{gpu}} - \text{lrv}_{2,\text{cpu}} \rvert$	$\max \lvert \text{lrm}_{\text{gpu}} - \text{lrm}_{\text{cpu}} \rvert$	$\max \lvert \text{lrm}_{1,\text{gpu}} - \text{lrm}_{1,\text{cpu}} \rvert$	$\max \lvert \text{lrm}_{2,\text{gpu}} - \text{lrm}_{2,\text{cpu}} \rvert$	$\max \lvert \theta_{\text{gpu}} - \theta_{\text{cpu}} \rvert$	$\max \lvert \theta_{e,\text{gpu}} - \theta_{e,\text{cpu}} \rvert$	$\max \lvert \theta_{f,\text{gpu}} - \theta_{f,\text{cpu}} \rvert$	$\max \lvert \theta_{g,\text{gpu}} - \theta_{g,\text{cpu}} \rvert$	$\max \lvert \theta_{\text{mod,gpu}} - \theta_{\text{mod,cpu}} \rvert$
500	5vs5	124750	1.014×10^{-4}	9.251×10^{-5}	7.390×10^{-5}	2.735×10^{-6}	2.664×10^{-6}	2.663×10^{-6}	3.029×10^{-4}	2.306×10^{-4}	2.306×10^{-4}	1.165×10^{-4}	7.223×10^{-5}
500	10vs10	124750	1.295×10^{-4}	7.909×10^{-5}	7.134×10^{-5}	2.263×10^{-6}	3.116×10^{-6}	2.811×10^{-6}	2.498×10^{-4}	1.664×10^{-4}	1.664×10^{-4}	1.639×10^{-4}	1.230×10^{-4}
500	20vs20	124750	1.189×10^{-4}	8.749×10^{-5}	7.492×10^{-5}	2.862×10^{-6}	3.075×10^{-6}	2.823×10^{-6}	2.683×10^{-4}	1.788×10^{-4}	1.788×10^{-4}	1.078×10^{-4}	1.902×10^{-4}
500	40vs40	124750	1.359×10^{-4}	1.110×10^{-4}	1.196×10^{-4}	2.785×10^{-6}	2.932×10^{-6}	2.703×10^{-6}	1.530×10^{-4}	1.060×10^{-4}	1.060×10^{-4}	8.949×10^{-5}	1.233×10^{-4}
500	83vs83	124750	2.263×10^{-4}	1.554×10^{-4}	1.496×10^{-4}	5.341×10^{-6}	5.354×10^{-6}	4.263×10^{-6}	2.843×10^{-4}	1.783×10^{-4}	1.783×10^{-4}	1.866×10^{-4}	2.584×10^{-4}
1000	5vs5	499500	1.063×10^{-4}	1.169×10^{-4}	1.293×10^{-4}	2.930×10^{-6}	2.745×10^{-6}	2.683×10^{-6}	5.665×10^{-4}	4.686×10^{-4}	4.686×10^{-4}	1.500×10^{-4}	1.084×10^{-4}
1000	10vs10	499500	1.295×10^{-4}	8.805×10^{-5}	9.166×10^{-5}	2.850×10^{-6}	3.116×10^{-6}	2.942×10^{-6}	3.057×10^{-4}	2.756×10^{-4}	2.756×10^{-4}	1.639×10^{-4}	1.429×10^{-4}
1000	20vs20	499500	1.189×10^{-4}	8.749×10^{-5}	8.416×10^{-5}	3.236×10^{-6}	3.075×10^{-6}	2.826×10^{-6}	3.250×10^{-4}	1.821×10^{-4}	1.821×10^{-4}	1.919×10^{-4}	1.935×10^{-4}
1000	40vs40	499500	1.851×10^{-4}	1.110×10^{-4}	1.196×10^{-4}	3.144×10^{-6}	3.359×10^{-6}	2.885×10^{-6}	2.397×10^{-4}	1.568×10^{-4}	1.568×10^{-4}	1.577×10^{-4}	2.154×10^{-4}
1000	83vs83	499500	2.263×10^{-4}	1.722×10^{-4}	1.649×10^{-4}	5.341×10^{-6}	5.354×10^{-6}	4.718×10^{-6}	3.080×10^{-4}	2.544×10^{-4}	2.544×10^{-4}	1.866×10^{-4}	2.765×10^{-4}
5000	5vs5	12497500	1.447×10^{-4}	1.573×10^{-4}	1.676×10^{-4}	3.652×10^{-6}	2.925×10^{-6}	3.144×10^{-6}	1.271×10^{-3}	1.071×10^{-3}	1.071×10^{-3}	5.691×10^{-4}	1.143×10^{-4}
5000	10vs10	12497500	1.376×10^{-4}	1.327×10^{-4}	1.668×10^{-4}	4.051×10^{-6}	3.802×10^{-6}	3.775×10^{-6}	9.377×10^{-4}	6.449×10^{-4}	6.449×10^{-4}	2.928×10^{-4}	1.396×10^{-4}
5000	20vs20	12497500	1.756×10^{-4}	1.251×10^{-4}	2.333×10^{-4}	3.249×10^{-6}	3.288×10^{-6}	3.321×10^{-6}	6.205×10^{-4}	4.266×10^{-4}	4.266×10^{-4}	3.521×10^{-4}	2.781×10^{-4}
5000	40vs40	12497500	2.157×10^{-4}	1.988×10^{-4}	2.465×10^{-4}	3.871×10^{-6}	3.558×10^{-6}	3.614×10^{-6}	3.949×10^{-4}	3.704×10^{-4}	3.704×10^{-4}	2.540×10^{-4}	2.642×10^{-4}
5000	83vs83	12497500	3.342×10^{-4}	2.569×10^{-4}	3.586×10^{-4}	6.338×10^{-6}	5.354×10^{-6}	5.500×10^{-6}	9.593×10^{-4}	4.526×10^{-4}	4.526×10^{-4}	5.066×10^{-4}	5.665×10^{-4}
10000	5vs5	49995000	1.493×10^{-4}	1.666×10^{-4}	1.763×10^{-4}	3.865×10^{-6}	3.233×10^{-6}	3.144×10^{-6}	2.369×10^{-3}	1.274×10^{-3}	1.274×10^{-3}	1.095×10^{-3}	1.293×10^{-4}
10000	10vs10	49995000	1.588×10^{-4}	1.703×10^{-4}	1.950×10^{-4}	4.073×10^{-6}	3.802×10^{-6}	3.799×10^{-6}	9.377×10^{-4}	1.377×10^{-3}	1.377×10^{-3}	1.349×10^{-3}	1.963×10^{-4}
10000	20vs20	49995000	2.137×10^{-4}	1.747×10^{-4}	2.333×10^{-4}	3.612×10^{-6}	3.686×10^{-6}	3.321×10^{-6}	6.693×10^{-4}	4.984×10^{-4}	4.984×10^{-4}	3.521×10^{-4}	2.755×10^{-4}
10000	40vs40	49995000	2.644×10^{-4}	2.317×10^{-4}	2.862×10^{-4}	4.442×10^{-6}	3.676×10^{-6}	4.548×10^{-6}	5.707×10^{-4}	3.704×10^{-4}	3.704×10^{-4}	2.651×10^{-4}	3.344×10^{-4}
10000	83vs83	49995000	4.103×10^{-4}	2.780×10^{-4}	4.757×10^{-4}	6.338×10^{-6}	5.800×10^{-6}	5.500×10^{-6}	9.593×10^{-4}	4.748×10^{-4}	4.748×10^{-4}	5.066×10^{-4}	7.312×10^{-4}
18244	5vs5	166412646	1.500×10^{-4}	1.739×10^{-4}	1.763×10^{-4}	4.724×10^{-6}	4.036×10^{-6}	3.144×10^{-6}	2.369×10^{-3}	1.274×10^{-3}	1.274×10^{-3}	1.095×10^{-3}	1.309×10^{-4}
18244	10vs10	166412646	1.658×10^{-4}	1.703×10^{-4}	1.950×10^{-4}	4.073×10^{-6}	3.889×10^{-6}	4.130×10^{-6}	9.377×10^{-4}	1.377×10^{-3}	1.377×10^{-3}	1.349×10^{-3}	1.979×10^{-4}
18244	20vs20	166412646	2.137×10^{-4}	1.747×10^{-4}	2.771×10^{-4}	3.809×10^{-6}	3.961×10^{-6}	3.632×10^{-6}	8.334×10^{-4}	5.031×10^{-4}	5.031×10^{-4}	3.521×10^{-4}	3.571×10^{-4}
18244	40vs40	166412646	2.683×10^{-4}	2.317×10^{-4}	3.301×10^{-4}	4.573×10^{-6}	4.482×10^{-6}	4.548×10^{-6}	5.841×10^{-4}	4.370×10^{-4}	4.370×10^{-4}	3.538×10^{-4}	3.807×10^{-4}
18244	83vs83	166412646	4.273×10^{-4}	3.054×10^{-4}	5.232×10^{-4}	6.338×10^{-6}	6.577×10^{-6}	5.982×10^{-6}	9.593×10^{-4}	5.888×10^{-4}	5.888×10^{-4}	5.066×10^{-4}	7.326×10^{-4}

Table C.2. GTE data: maximum absolute CPU/GPU differences. Each entry is $\max |x_{\text{gpu}} - x_{\text{cpu}}|$ over all matched gene pairs (Partner, Pair) in that output file.

Appendix D. Runtime analysis and Timing Breakdown

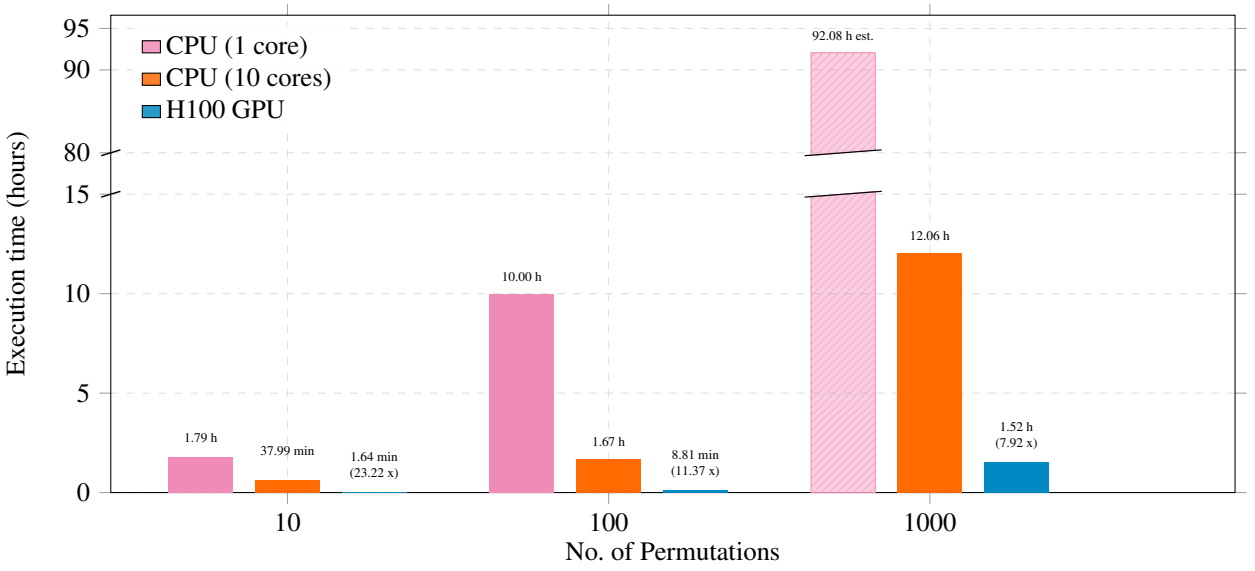
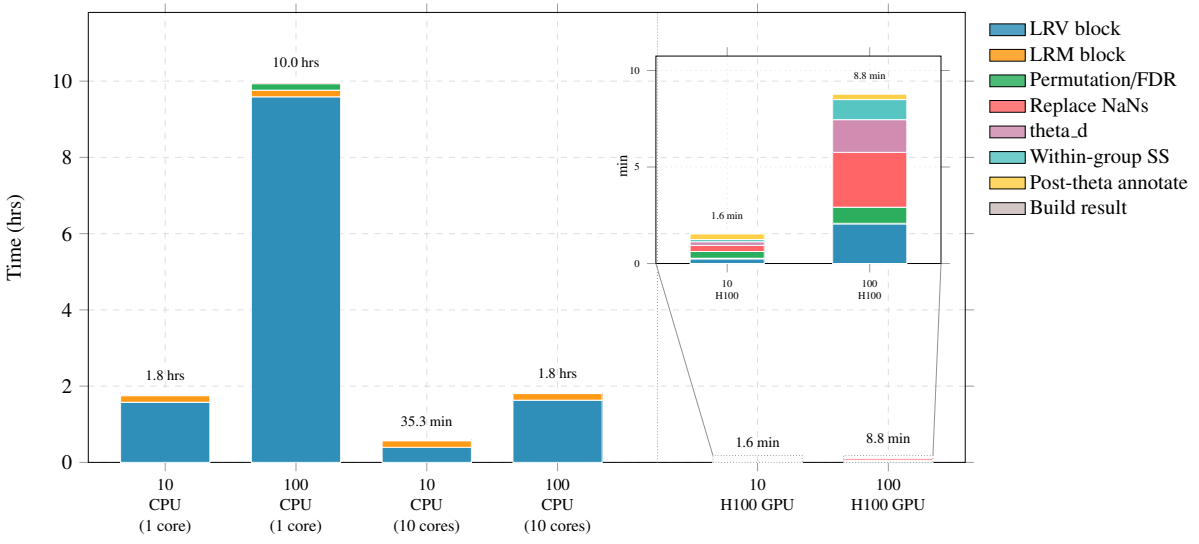


Fig. D.6. End-to-end runtime of `propr`'s FDR computation for 18,244 genes using 10, 100, and 1000 permutations. CPU runs are shown for one core and ten cores, and compared with the GPU backend (H100). Due to the impracticality and infeasibility of running the 1000 permutations case on a single CPU core we estimate it by comparing how different parts of the runtime scale between the 10- and 100-permutations case.



End-End time breakdown for increasing No. of Permutations across backends

Fig. D.7. Runtime breakdown for 10- and 100-permutations runs across CPU and GPU backends (H100 GPU). On the CPU runs, LRV and LRM dominate the runtime, together accounting for 96.7-98.3% of total execution time. On the other hand, in the GPU runs, LRV and LRM contribute a much smaller share of runtime, 17.4% for 10 permutations and 23.7% for 100 permutations

.

Appendix E. Nsight Systems Traces

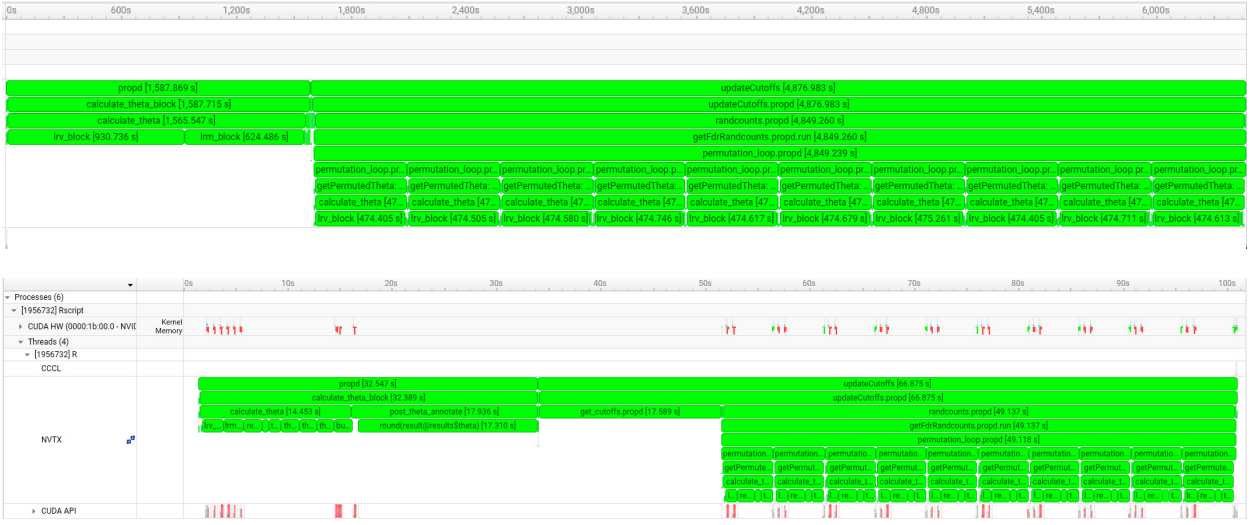


Fig. E.8. Nsight Systems trace for a `propr` CPU run on 18,244 genes and 166 samples, computing the FDR using 10 permutations (top), and for the same configuration using a single H100 GPU (bottom). As is seen in the CPU trace LRV and LRM dominate execution time, while the GPU trace substantially shortens stages in which LRV/LRM are involved and shifts work to the permutation kernels.

Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Category	Operation
71.1%	4.670 s	31	150.648 ms	158.682 ms	2.656 µs	174.805 ms	40.514 ms	MEMORY_OPER	[CUDA memcpy Device-to-Host]
21.5%	1.412 s	40	35.291 ms	369.584 µs	1.024 µs	128.449 ms	57.347 ms	MEMORY_OPER	[CUDA memcpy Host-to-Device]
7.1%	465.910 ms	23	20.257 ms	19.462 ms	19.458 ms	37.742 ms	3.812 ms	CUDA_KERNEL	void propr::detail::cuda::lrv_basic<propr::cuda::traits::lrv_basic>(float *, unsigned long, float *, int, int)
0.2%	12.039 ms	11	1.094 ms	1.092 ms	1.086 ms	1.112 ms	7.830 µs	CUDA_KERNEL	void propr::detail::cuda::bucket_counts_shared_cutoffs_shared_buckets_kernel<(propr::detail::cuda::
0.1%	5.724 ms	3	1.908 ms	1.908 ms	1.905 ms	1.911 ms	3.247 µs	CUDA_KERNEL	void propr::detail::cuda::lrm_basic_phase_2<propr::cuda::traits::lrm_basic>(float *, float *, int)
0.0%	1.613 ms	1	1.613 ms	1.613 ms	1.613 ms	1.613 ms	0 ns	CUDA_KERNEL	propr::detail::cuda::count_joint_zeros(const int *, unsigned long, int, int *)
0.0%	814.143 µs	1	814.143 µs	814.143 µs	814.143 µs	814.143 µs	0 ns	CUDA_KERNEL	propr::detail::cuda::labRcpp(int *, int *, unsigned long)
0.0%	109.184 µs	29	3.765 µs	3.776 µs	800 ns	5.504 µs	977 ns	MEMORY_OPER	[CUDA memset]
0.0%	34.752 µs	3	11.584 µs	9.536 µs	9.472 µs	15.744 µs	3.602 µs	CUDA_KERNEL	void propr::detail::cuda::lrm_basic_phase_1<propr::cuda::traits::lrm_basic>(float *, unsigned long, flo
0.0%	16.480 µs	1	16.480 µs	16.480 µs	16.480 µs	16.480 µs	0 ns	CUDA_KERNEL	void propr::detail::cuda::count_per_feature<(int)256>(const float *, unsigned long, int, int, int *)
0.0%	5.056 µs	2	2.528 µs	2.528 µs	2.528 µs	2.528 µs	0 ns	CUDA_KERNEL	void cub::CUB_200700_900_NS::DeviceScanKernel<cub::CUB_200700_900_NS::DeviceScanPolicy<u
0.0%	3.808 µs	2	1.904 µs	1.904 µs	1.600 µs	2.208 µs	429 ns	CUDA_KERNEL	void cub::CUB_200700_900_NS::DeviceScanInitKernel<cub::CUB_200700_900_NS::ScanTileState<ur

Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Category	Operation
65.8%	32.587 s	211	154.439 ms	153.959 ms	2.624 µs	174.643 ms	16.023 ms	MEMORY_OPER	[CUDA memcpy Device-to-Host]
26.0%	12.864 s	310	41.498 ms	508.623 µs	1.024 µs	129.730 ms	59.157 ms	MEMORY_OPER	[CUDA memcpy Host-to-Device]
8.0%	3.971 s	203	19.559 ms	19.468 ms	19.461 ms	37.744 ms	1.283 ms	CUDA_KERNEL	void propr::detail::cuda::lrv_basic<propr::cuda::traits::lrv_basic>(float *, unsigned long, float *, int, int)
0.2%	110.542 ms	101	1.094 ms	1.093 ms	1.085 ms	1.134 ms	7.517 µs	CUDA_KERNEL	void propr::detail::cuda::bucket_counts_shared_cutoffs_shared_buckets_kernel<(propr::detail::cuda::
0.0%	5.733 ms	3	1.911 ms	1.911 ms	1.910 ms	1.912 ms	981 ns	CUDA_KERNEL	void propr::detail::cuda::lrm_basic_phase_2<propr::cuda::traits::lrm_basic>(float *, float *, int)
0.0%	1.605 ms	1	1.605 ms	1.605 ms	1.605 ms	1.605 ms	0 ns	CUDA_KERNEL	propr::detail::cuda::count_joint_zeros(const int *, unsigned long, int, int *)
0.0%	813.791 µs	1	813.791 µs	813.791 µs	813.791 µs	813.791 µs	0 ns	CUDA_KERNEL	propr::detail::cuda::labRcpp(int *, int *, unsigned long)
0.0%	794.719 µs	209	3.802 µs	3.808 µs	800 ns	5.536 µs	366 ns	MEMORY_OPER	[CUDA memset]
0.0%	34.560 µs	3	11.520 µs	9.440 µs	9.408 µs	15.712 µs	3.630 µs	CUDA_KERNEL	void propr::detail::cuda::lrm_basic_phase_1<propr::cuda::traits::lrm_basic>(float *, unsigned long, flo
0.0%	16.288 µs	1	16.288 µs	16.288 µs	16.288 µs	16.288 µs	0 ns	CUDA_KERNEL	void propr::detail::cuda::count_per_feature<(int)256>(const float *, unsigned long, int, int, int *)
0.0%	5.024 µs	2	2.512 µs	2.512 µs	2.464 µs	2.560 µs	67 ns	CUDA_KERNEL	void cub::CUB_200700_900_NS::DeviceScanKernel<cub::CUB_200700_900_NS::DeviceScanPolicy<u
0.0%	3.776 µs	2	1.888 µs	1.888 µs	1.760 µs	2.016 µs	181 ns	CUDA_KERNEL	void cub::CUB_200700_900_NS::DeviceScanInitKernel<cub::CUB_200700_900_NS::ScanTileState<ur

Fig. E.9. Nsight Systems profiles for a 10-permutation run (top) and 100-permutation (bottom) show that the runtime is dominated by memory operations – [CUDA memcpy Device-to-Host] and [CUDA memcpy Host-To-Device] – rather than computation as we copy back and forth from the GPU as data moves between the R and the `propr`'s Rcpp layer. The combined memory-transfer operations in the 10-permutation case account for approximately 92.6% of total GPU runtime (**not** entire execution runtime), while kernel execution accounts for only about 7–8%. The same holds for the 100-permutation case, where combined memory-transfer operations account for approximately 91.8% of the GPU runtime and kernels again contribute only about 8%.

Appendix F. Memory Runtime analysis

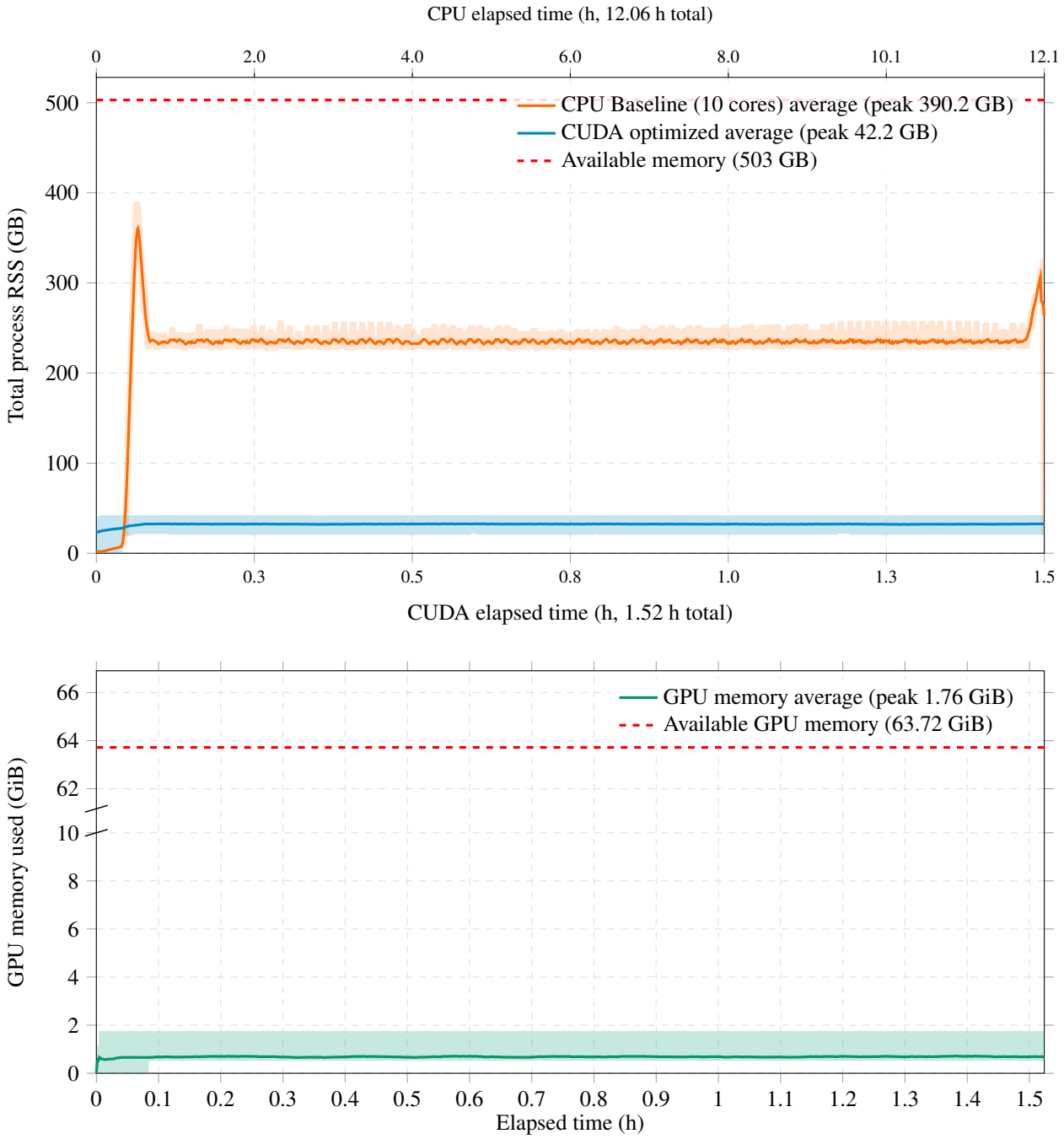


Fig. F.10. Host memory (top) and GPU memory (bottom) execution footprint of `propd` run for 18,244 genes and 1000 permutations. The Host memory plot shows the CPU baseline and CUDA-optimized runs on separate aligned time axes because their wall times substantially differ. The plotted RSS is the summed resident memory of the sampled R processes as the CPU execution run was run with `ncores=10`; solid lines show rolling averages and shaded bands show the rolling min–max envelope. The CUDA implementation reduces the peak host RSS from 390.2 GiB to 42.2 GiB, with peak GPU memory reaching 1.76 GiB.

Appendix G. Performance by Component

Below, we break down the runtime improvements of many of the core routines of `propr`.

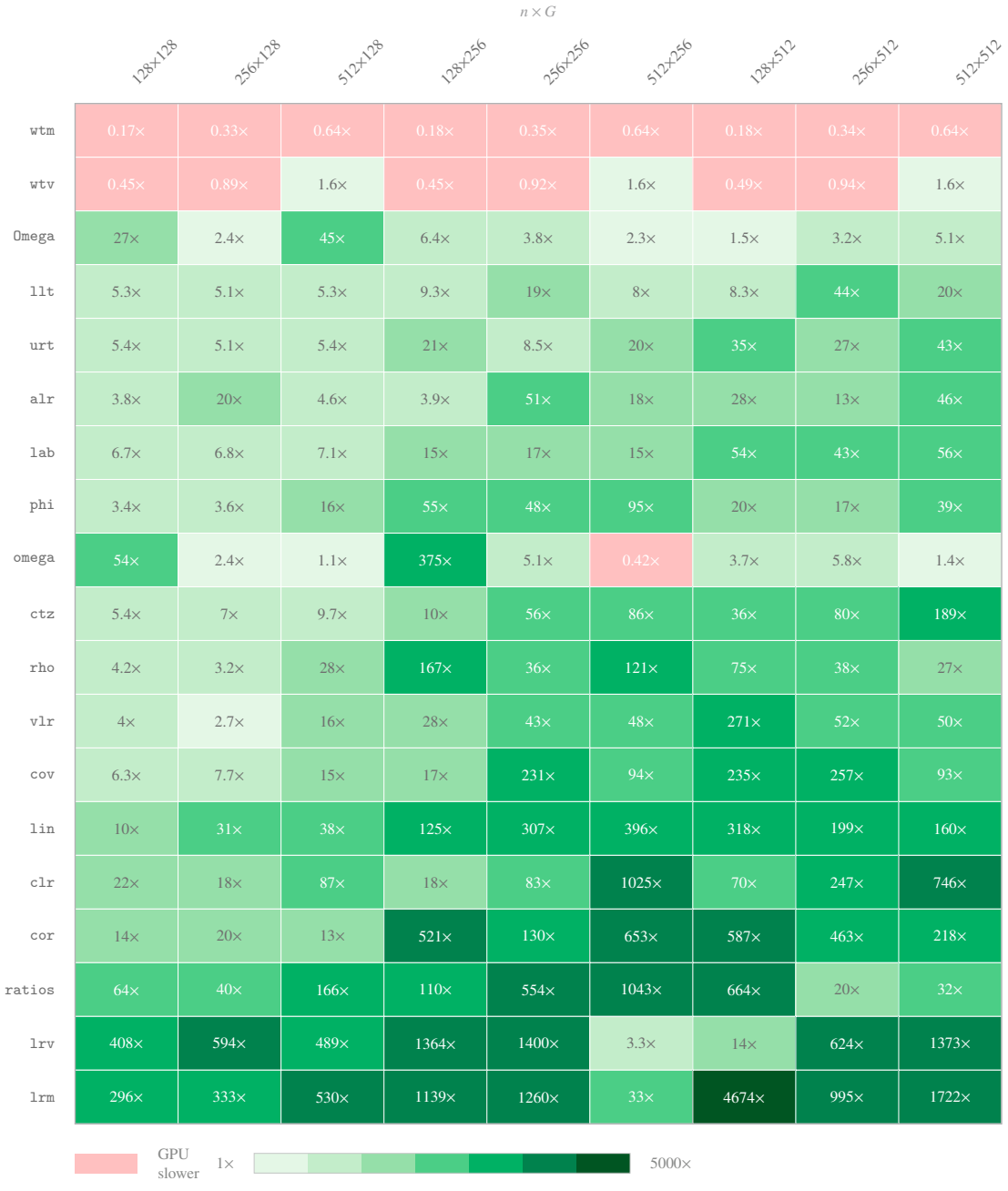


Fig. G.11. Performance comparison for core function routines between the CPU kernels and their GPU counterparts

Appendix H. Nsight Compute Profile For LRV

As demonstrated above, the LRV kernel contributed the most to the overall runtime of the `propr`. For this reason, we present a more detailed NVIDIA Nsight Compute profile of this kernel to help readers better understand the performance behavior of the GPU implementation. Since the original CPU execution runtime was largely dominated by LRV, its compute throughput, memory throughput, roofline plot, and occupancy provide useful insight into how efficiently GPU resources are being used by the now introduced GPU kernel.

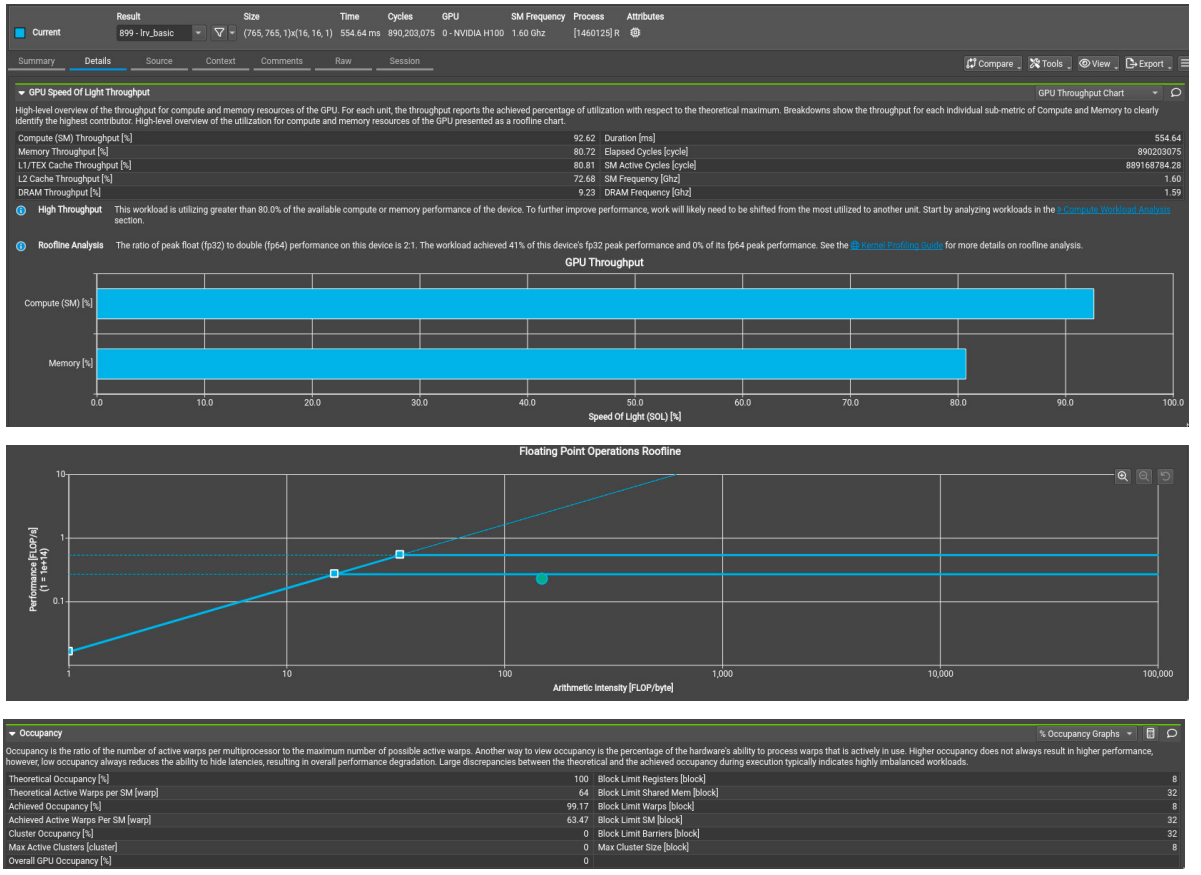


Fig. H.12. NVIDIA Nsight Compute performance summary for the `lrv_basic` CUDA kernel on an NVIDIA H100 GPU. The Speed-of-Light analysis shows very high utilization, with 92.62% SM compute throughput and 80.72% memory throughput, indicating that both compute and memory resources are being used very effectively. The roofline analysis shows that the kernel achieves approximately 41% of the theoretical FP32 peak performance. Although this is below the absolute hardware peak, it is a respectable result for a practical GPU kernel, especially given the very high SM throughput and occupancy. Occupancy is also very high, with 99.17% achieved occupancy and nearly the maximum number of active warps per SM. This suggests that the GPU has sufficient parallel work and is not limited by poor utilization.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 136–146

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Fourth EuroHPC User Days 2026

An FFT-oriented performance analysis and optimization of Quantum ESPRESSO on top-tier EuroHPC systems

Fabrizio Ferrari Ruffino[a,*], Francesco Andreucci[c,*], Pietro Delugas[c], Angela Acocella[b], Giacomo Rossi[d], Fabio Affinito[b], Ivan Carnimeo[c], Ivan Girotto[e], Sergio Orlandini[b], Laura Bellentani[b,*]

[a]*CNR-IOM Istituto dell'Officina dei Materiali, area SISSA, via Bonomea 265, 34136 Trieste, Italy*
[b]*CINECA, Via Magnanelli 6/3, 40033 Casalecchio di Reno, BO, Italy*
[c]*SISSA, Scuola Internazionale Superiore di Studi Avanzati, via Bonomea 265, 34136 Trieste, Italy*
[d]*Advanced Micro Devices S.p.A., Via Riccardo Lombardi No 19/16 and 19/18, 20153 Milano, Italy*
[e]*ICTP, International Centre for Theoretical Physics, Strada Costiera 11, 34151 Trieste, Italy*

Abstract

Density functional theory (DFT), combined with the plane-waves (PW) representation, forms the basis of a large class of first-principles materials simulation codes. In these applications, Fast Fourier Transforms (FFTs) constitute the core computational kernel that strongly impacts overall performance. In the Quantum ESPRESSO suite (QE), an open-source code widely used for material design, FFTs are tightly integrated with the application's parallelization strategy and data distribution. This coupling enables efficient application-aware optimization, although it makes direct benchmarking against standard distributed FFT libraries nontrivial. However, the emergence of highly optimized distributed FFT APIs such as `cuFFTMp`, together with topology-aware communication libraries such as `NCCL`, motivates a reassessment of this application-specific implementation to evaluate its performance against current HPC standards and identify optimization opportunities on modern architectures.

In this work, we analyze and benchmark `FFTXlib`, the distributed FFT library used in QE, against NVIDIA's `cuFFTMp` on the pre-exascale systems Leonardo Booster and MareNostrum5 ACC. From this study, we derive insights that can inform future optimizations and guide new developments of PW-DFT codebases. We further introduce an optimization based on an adaptive batching strategy that relies on the specific data distribution of `FFTXlib` and significantly improves the overall strong-scaling behaviour. We demonstrate its impact through production-scale benchmarks of representative workloads (a 1413-atom water system and a 1532-atom carbon nanotube system) on three top-tier EuroHPC platforms, including NVIDIA and AMD accelerators (Leonardo Booster, MareNostrum5 ACC, LUMI-G).

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026

Keywords: GPU; HPC; FFTXlib; PWSCF; QE; Quantum ESPRESSO

* Corresponding authors.
E-mail address: ferrariruffino@iom.cnr.it (FFR) - fandreuc@sissa.it (FA) - l.bellentani@cineca.it (LB)

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.025

1. Introduction

Computational materials science has been transformed by Density Functional Theory (DFT), the leading first-principles framework for predictive simulations, evolving alongside advances in high-performance computing (HPC) and machine learning [1, 2]. Many large-scale simulations rely on plane-wave pseudopotential software [3, 4, 5, 6, 7], where Fast Fourier Transforms (FFTs) enable efficient transformations of thousands of wavefunctions between reciprocal and real space during Hamiltonian evaluation.

In these codes, FFTs are tightly coupled to the application's parallel structure and interact with two key levels of parallelism: electronic orbitals (bands) and plane waves. Since typical calculations involve hundreds to thousands of bands, FFTs are executed concurrently on large sets of wavefunctions and often represent a significant fraction of the total runtime. Consequently, efficient FFT parallelisation and accelerator offloading, with a focus on maximising throughput, are critical for scalability, performance, and performance portability on current and emerging HPC architectures.

The Quantum ESPRESSO suite (QE) is one of the most widely used applications for first-principles materials simulations [8, 3]. Its development has closely followed both methodological advances in electronic-structure theory and the evolution of HPC architectures, with substantial optimisation efforts carried out within European projects such as MaX, ADMIRE, and DARE [9, 10, 11, 12, 13]. As part of this effort, QE relies on its `FFTXlib` library, which implements distributed FFTs tailored to the application's multi-level parallelism and data layouts.

The emergence of accelerator-based supercomputers and highly optimised distributed FFT libraries such as `cuFFTMp` and `heFFTe` [14, 15, 16, 17] motivates a reassessment of `FFTXlib` on modern systems. The latest published analysis of `FFTXlib` [18] predates the GPU-enabled evolution of QE [19, 20, 21], providing a natural baseline for an updated evaluation of its scalability, throughput, and performance portability on current accelerator-based architectures.

In this work, we assess `FFTXlib` on two EuroHPC systems with distinct architectures, Leonardo Booster (CINECA) and MareNostrum5 ACC (BSC-CNS), using `cuFFTMp` [16] as a reference distributed FFT implementation. This comparison quantifies the impact of QE-specific features, including spherical reciprocal-space cutoffs and distributed band batching, on realistic workloads. Despite being less hardware-specialised than state-of-the-art FFT libraries, `FFTXlib` retains a significant advantage by reducing both computational and communication costs.

Building on this analysis, we introduce an adaptive batching strategy that dynamically aggregates FFT operations to maximise communication throughput and improve the utilisation of multinode FFT distributions. By exploiting high-bandwidth communication backends such as `NCCL` [22], the proposed approach reduces sensitivity to data distribution, sustains high communication efficiency across a wide range of scales, and significantly improves strong-scaling performance.

Finally, we validate these optimizations through two production-scale DFT calculations of a 1413-atom water system and a 1532-atom porphyrin-functionalized carbon nanotube, both representative of modern electronic-structure workloads with high computational and communication demands (using the SCAN meta-GGA functional [23], a state-of-the-art exchange-correlation approximation).

In addition to Leonardo Booster and MareNostrum5 ACC, we report results on the AMD GPU-based LUMI-G system (CSC), demonstrating improved performance portability across NVIDIA- and AMD-based accelerator platforms.

2. FFTs in PW-DFT calculations

In the framework of Density Functional Theory (DFT), the electronic properties of a physical system are determined by functionals of the spatially dependent electronic density. The latter is obtained by solving a set of self-consistent Kohn–Sham (KS) equations [24]:

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V_{KS}(\mathbf{r}, \rho(\mathbf{r}))\right] \phi_i(\mathbf{r}) = \epsilon_i \, \phi_i(\mathbf{r}), \qquad i = 1, \ldots, N_b, \tag{1}$$

where V_{KS} is the KS potential and ϕ_i are the KS orbitals (or bands), which form a set of N_b electronic states. To simplify the notation, k-point sampling is omitted throughout this work. The presented methods and optimisations are nevertheless applicable to general k-point calculations. The electronic density is given by:

$$\rho(\mathbf{r}) = \sum_{i=1}^{N_b} |\phi_i(\mathbf{r})|^2. \tag{2}$$

Starting from an initial guess, the KS equations are solved iteratively until convergence of the electronic density and total energy is achieved. The KS potential in Eq. (1) consists of several contributions, some of which are more efficiently evaluated in reciprocal space and others in real space. As a result, Fast Fourier Transforms (FFTs) are used to switch between these representations, enabling both the calculation and the application of operators in the domain where they are computationally most convenient (typically where they are local). In plane-wave implementations, each orbital is expanded in a basis of N_{pw} plane waves:

$$\phi_i(\mathbf{r}) = \sum_{\mathbf{G}}^{N_{\text{pw}}} C_i(\mathbf{G}) \frac{e^{i\mathbf{G}\cdot\mathbf{r}}}{\sqrt{\Omega}}, \qquad i = 1, \ldots, N_b, \tag{3}$$

where Ω is the volume of the unit cell and each $\mathbf{G}$ is a reciprocal space lattice vector, which corresponds to a point on the reciprocal FFT grid. The size of this grid is determined by the energy cutoff and the crystal lattice vectors of the physical system, together defining a spherical region in reciprocal space, which determines the number of plane waves N_{pw} according to: $(\hbar^2/2m)\,|\mathbf{G}^2| < E_{\text{cutoff}}$.

Since the electronic density involves products of orbitals in direct space (corresponding to convolutions in reciprocal space), its significant Fourier components extend to twice as long wave-vectors as those of the wavefunctions themselves. For this reason, the density requires a cutoff four times larger than the wavefunction cutoff. Fig. 1 illustrates the domains in real and reciprocal space. The spherical cutoff defining the reciprocal-space domain of the electronic density is inscribed within the cubic grid, while the corresponding cutoff for the Kohn–Sham (KS) orbitals has half the radius.

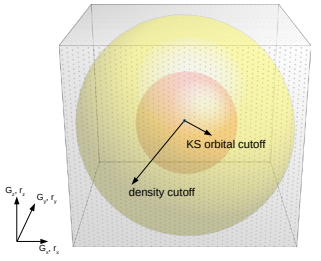


Fig. 1. Assuming a cubic grid in real space, the electronic density in reciprocal space is defined inside the bigger sphere inscribed in the direct grid. KS orbitals, instead, are defined in the smaller darker sphere (typically with half of the density radius).

This cutoff-based spherical representation enables a tailored decomposition of reciprocal space, where only the region within the cutoff sphere is stored and processed, significantly reducing both memory usage and computational cost. For this reason, KS orbitals are typically stored in reciprocal space. Consequently, the application of a local potential operator $\hat{V}$ defined in real space to a KS orbital involves transforming the orbital to real space, applying the operator, and transforming it back via FFTs:

$$[\hat{V} \cdot C_i](\mathbf{G}) = \mathcal{F}[\ \hat{V}(\mathbf{r})\ \phi_i(\mathbf{r})\ ] = \mathcal{F}[\ \hat{V}(\mathbf{r})\ \mathcal{F}^{-1}[C_i(\mathbf{G})]\ ], \qquad i = 1, \ldots, N_b, \tag{4}$$

where the dot · represents the convolution product.
These FFTs associated with KS orbitals typically dominate the overall execution time in local and semi-local PW-DFT simulations (the performance bottlenecks related to non-local hybrid functionals are different and out of the scope of this work). Although the electronic density is represented with a larger cutoff, its FFT cost is comparatively small due to the limited number of such transforms. In contrast, KS orbital FFTs are performed for a large number of bands, often up to several thousands, making them the primary computational bottleneck. For this reason, in this work we focus on KS orbital FFTs, corresponding to a cutoff radius equal to one quarter of the real-space grid size.

3. FFT data distribution in Quantum ESPRESSO

The dominant memory requirement in QE arises from the buffer storing KS orbitals expanded in plane waves. Its memory footprint typically scales as:

$$\text{buffer-size} \sim N_{\text{k}} \cdot N_{\text{b}} \cdot N_{\text{pw}} \;/\; N_{\text{ranks}} \,, \tag{5}$$

Where N_{ranks} is the number of MPI processes. The number of bands N_b typically ranges from hundreds to several thousands, the number of k-points N_{k} from one to a few thousands (e.g. in metallic systems), the grid size N_{pw} might approach 1000^3 points. As a result, individual FFTs are relatively small by modern HPC standards [14, 25], but overall they are numerous and distributed across many processes due to the plane-wave based parallelism at the core of QE.

The data distribution in reciprocal space is non-standard with respect to most FFT implementations, and tailored to the plane-wave representation. Each plane wave corresponds to a grid point within a spherical cutoff. To balance the workload in QE, outside `FFTXlib`, the portion of the grid inside the cutoff sphere is partitioned into one-dimensional z-sticks of varying lengths, which are then distributed uniformly across processes. When it comes to FFT execution, these sticks are extended along the full z-direction, forming a cylindrical domain that intersects the cutoff region in the xy-plane, as shown in Fig. 2. During an inverse FFT, data associated with each (x, y) point within the cutoff circle is transformed along the entire z-axis, after which data transposition among processes is performed in order to match the standard slab decomposition of the real space domain. This is done by means of All-to-All communications and rectangular memcpy backends applied to the cylindrical buffer. Finally, the FFT transformation is completed across all the square xy-slabs.

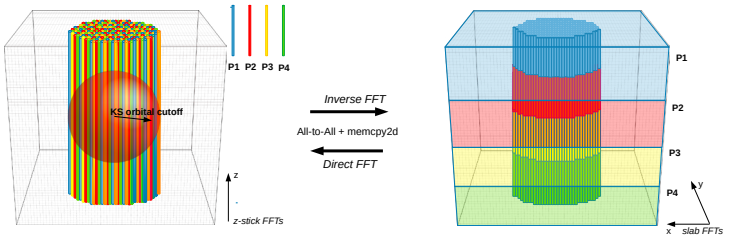


Fig. 2. Domain decomposition in a parallel execution (four processes) of an inverse FFT applied to a single KS orbital by means of the `FFTXlib` tailored slab approach. Starting from reciprocal space, the 1D z-sticks crossing the cutoff sphere are uniformly distributed among processes, and Fourier transformed along the z-direction (upper panel). Then, they are divided into four chunks and distributed in direct space according to the slab partition (lower panel). Finally, each xy-slab is Fourier transformed along the whole xy-domain.

In addition to spatial decomposition, a further level of parallelism arises from the large number of Kohn-Sham orbitals, which can be processed independently [Eq. (4)]. This enables concurrent execution of multiple FFTs, shifting the optimization focus from single-transform latency to overall throughput. As introduced by Romero et al. [19], orbitals are organized into batches and sub-batches, allowing multiple 1D and 2D FFT operations, as well as communication buffers, to be merged together. FFT computations and communications are then executed asynchronously across different sub-batches, as illustrated in Fig. 3. So far (QE v7.5), the standard configuration is hard-coded in `FFTXlib` and consists of batches of 16 bands, organized into 4 sub-batches of 4 bands each.

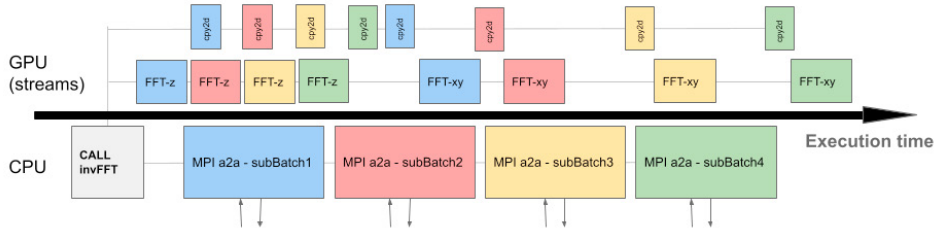


Fig. 3. Simplified flow diagram of an inverse batched FFT execution (as it was implemented in `FFTXlib` [19]). Here a batch made of four sub-batches containing multiple KS orbitals is processed. Each color is associated to one sub-batch. The 3d Fourier transform is decomposed into communication blocks, scalar FFT computation and data movement. While communications related to one sub-batch are executed, computation and data movement of the other sub-batches are performed asynchronously on dedicated GPU streams.

Batching in QE is primarily applied across all processes, reflecting its strong reliance on plane-wave parallelism. Band parallelism, i.e. the distribution of entire orbitals across processes or groups of processes, is also supported and currently under active optimization. Nevertheless, the following analysis focuses on FFT performance rather than higher-level parallelization strategies within DFT simulations. Consequently, the results are largely independent of the specific band distribution scheme and remain valid whether FFTs are executed within band groups or in configurations without band parallelism. Furthermore, the analysis enables an indirect comparison (limited to FFTs) of different band distribution strategies by examining FFT performance across varying numbers of nodes.

4. A performance assessment of FFTXlib

There are two main reasons for using a tailored library such as `FFTXlib` in Quantum ESPRESSO . First, `FFTXlib` is tightly aligned with the QE data distribution, which heavily relies on workload partitioning over plane waves; integrating an external library would require substantial and intrusive refactoring of a mature codebase. Second, as discussed in Sect. 3, `FFTXlib` incorporates PW-DFT–specific features, most notably the adjustable spherical cutoff in reciprocal space and distributed band batching, which are generally not available in standard libraries and yield significant performance benefits.

To quantify these effects, we benchmark `FFTXlib` against the `cuFFTMp` library [16] on the NVIDIA-based systems Leonardo Booster and MareNostrum5 ACC (Table 1). We developed a dedicated miniapp that interfaces `FFTXlib` with configurable cutoff and batching parameters, and integrates `cuFFTMp` using `NVSHMEM` library [26] for GPU memory management. The software stack relies on the NVIDIA implementation of `OpenMPI`, `HPC-X MPI` [27] for inter-GPU communications.

HPC cluster	Leonardo Booster	MareNostrum5 ACC	LUMI-G
Centre	CINECA	BSC-CNS	CSC
Nodes	3456	1120	2978
Processors/node	1x Intel Xeon Platinum 8358 32 cores 512 GB	2x Intel Sapphire Rapids 8460Y 40 cores 512 GB	1x AMD EPYC 7A53 64 cores 512 GB
GPUs/node	4x NVIDIA A100 64GB	4x NVIDIA H100 64GB	4x MI250x dual-die 128GB
Network	NVLink 3.0 (200 GB/s) + Mellanox InfiniBand HDR (200 Gbps)	NVLink (300 GB/s) + Mellanox InfiniBand NDR (400 Gbps)	Infinity Fabric (up to 400 GB/s) + Slingshot-11 (200 Gbps)
Network topology	Fat-tree	DragonFly+	Dragonfly

Table 1. System specifications of Leonardo Booster, MareNostrum5 ACC partitions; LUMI-G partition is added in view of the QE benchmark of Sect. 6

A fair comparison must account for differences in reciprocal-space decomposition. As discussed in Sect. 2, in PW-DFT simulations the relevant reciprocal-space domain is typically confined within a sphere whose radius, for most FFTs, is approximately one quarter of the real-space grid size. This enables a tailored decomposition that excludes

the unused region outside the cutoff sphere, thereby reducing both computation and communication overhead. Accordingly, we compare `FFTXlib` executions performed both with and without the spherical domain decomposition against standard slab-based `cuFFTMp` executions. This isolates the impact of the application-specific cutoff while also reflecting the intrinsic differences between stick- and slab-based data layouts. GPU executions with distributed band batching are also included, with results normalized per band. As a second reference, we consider ideal pure band parallelism using `cuFFT`, where independent scalar 3D FFTs are executed locally on each GPU and execution times are normalized by the number of processes. We adopt this dual reference framework because FFT sizes in QE, by current HPC standards, lie at the boundary between single-GPU and distributed FFT regimes.

Fig. 4 reports the inverse FFT time-to-solution for different grid sizes on Leonardo and MareNostrum 5. Without `FFTXlib`-specific features, `cuFFTMp` achieves better performance. However, enabling the spherical cutoff and distributed batching allows `FFTXlib` to outperform `cuFFTMp`: the cutoff reduces both computational workload and communication volume, while batching improves GPU utilization and communication efficiency.

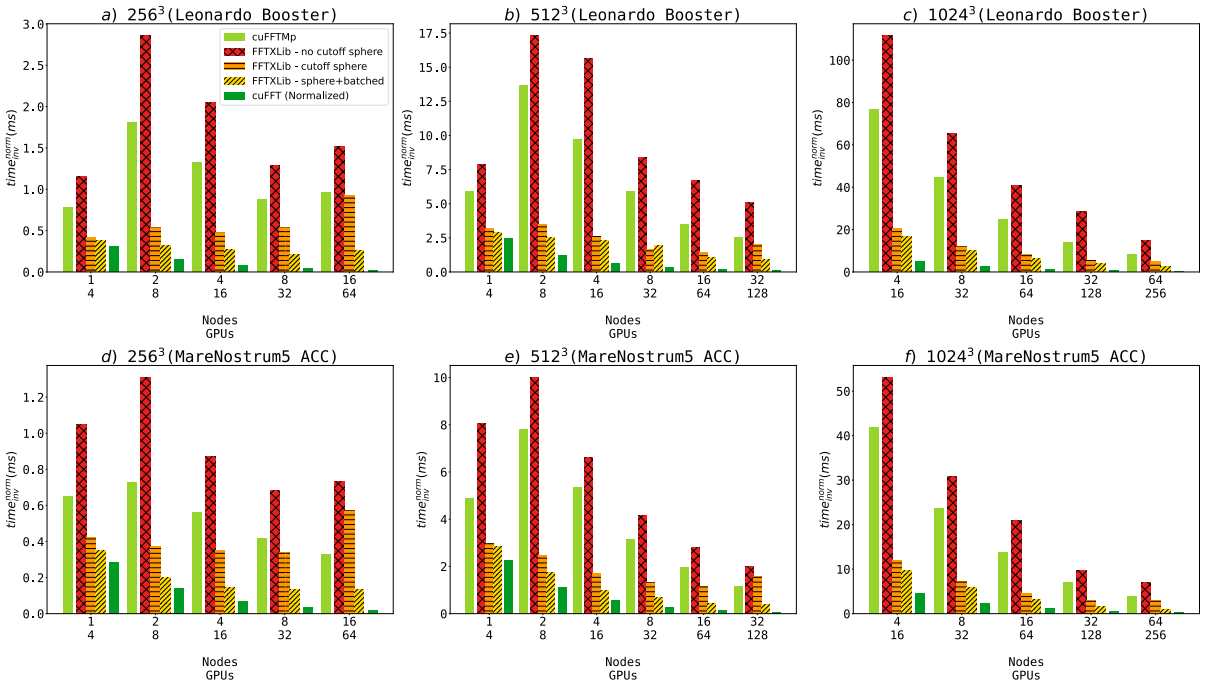


Fig. 4. Comparison of the time-to-solution of the inverse FFT w.r.t to the number of nodes for grids of size 256^3, 512^3 and 1024^3, respectively. The times are obtained with different libraries and configurations: `cuFFTMp` (plain light green), `FFTXlib` without the cutoff unbatched (criss-crossed yellow), `FFTXlib` with the cutoff unbatched (horizontally striped orange), `FFTXlib` with the cutoff batched with 4 sub-batches of size 4 each (diagonally striped red). Finally, in plain dark green, the time corresponding to ideal band scaling is obtained by normalising the serial `cuFFT` time on one GPU over the number of processes.

Three further considerations emerge from Fig. 4.

(i) On both systems, single-node executions without spherical cutoff outperform small multi-node runs because NVLink interconnect network provides higher bandwidth than InfiniBand (Table 1), particularly for smaller FFTs (256^3 and 512^3). In practical QE simulations, however, even moderate grid sizes are distributed across multiple nodes to balance memory and workload. In this regime, spherical cutoff and distributed batching significantly reduce the gap between single-node and distributed executions.

(ii) On MareNostrum 5 ACC, plane-wave distributed batched FFTs achieve performance close to the ideal pure band-parallel case, where FFTs are executed independently on each GPU, up to 2–4 nodes. This indicates that, on systems with sufficiently high interconnect bandwidth, FFT performance depends only weakly on the chosen band/plane-wave distribution. The near-equivalence between pure band parallelism and distributed plane-wave FFTs suggests that plane-wave decomposition can be introduced where needed for memory scalability and load balancing at

little additional cost, making a hybrid band/plane-wave strategy particularly attractive. Implementation of this scheme within QE is currently ongoing.

(iii) Despite the benefits of spherical cutoff and distributed batching, parallel efficiency degrades at large node counts. For fixed and relatively small FFT sizes, increasing the number of nodes reduces per-process computation, while communication becomes dominant. As communication buffers shrink, execution enters a latency-dominated regime, leading to poor scaling, particularly for smaller grids (256^3 and 512^3).

In the next section, we address this last limitation by introducing a strategy aimed at sustaining high communication bandwidth and improving strong-scaling behavior.

5. Adaptive orbital FFT batching and inclusion of NCCL

As discussed in Sect. 3, memory constraints in PW-DFT calculations typically limit FFT grid sizes to at most 1000^3 grid points. On modern GPU-based systems such FFTs could, in principle, run on a single node or GPU. In practice, however, FFT distribution follows the underlying plane-wave decomposition required to balance workload and distribute memory across the simulation. Consequently, the objective is not to maximize standalone FFT throughput, but rather to optimize performance within a realistic PW-DFT data distribution. For this reason, we consider grid sizes and decompositions representative of production QE workloads, even if atypical for general HPC FFT benchmarks.

When these FFTs are executed on modern GPU systems, increasing the number of processes leads to a regime where communication dominates computation, and the total execution time approaches that of the All-to-All communication kernel. For the slab-decomposition-based FFTs described in Sect. 3, this can be expressed as:

$$Time(\texttt{FFTXlib}) \approx Time(\text{All-to-All}) \qquad \text{for } N_{\text{ranks}} \gg 1 \tag{6}$$

In this regime, FFTs in PW-DFT become communication-bound, which is particularly detrimental for relatively small but distributed grids as used in QE. Improving `FFTXlib` performance therefore necessarily requires optimizing All-to-All communication bandwidth. This, in turn, involves two directions: (i) the use of high-performance communication backends, and (ii) the optimization of buffer sizes to fully exploit available bandwidth. The large number of orbitals typical of PW-DFT simulations provides an additional degree of freedom to address (ii).

Within this framework, we optimize `FFTXlib` throughput through two complementary strategies: an adaptive batching approach aimed at maximizing communication bandwidth, and the integration of the topology-aware `NCCL` communication backend [22], benchmarked against `HPC-X MPI`. These developments were initially implemented in the `FFTXlib` miniapp on the NVIDIA-based Leonardo Booster system and later integrated into QE.

Fig. 5 reports the All-to-All communication throughput measured with `HPC-X MPI` and `NCCL`. `FFTXlib` reaches peak collective bandwidth only above a buffer-size threshold, typically between 10 and 100 MB depending on node count.

As a consequence, FFT scalability rapidly deteriorates at large node counts for fixed grid sizes, since communication buffers become too small and execution enters a latency-dominated regime. Standard QE batching (Fig. 3), based on fixed batch and sub-batch sizes of 16 and 4 bands, partially alleviates this effect by merging communication buffers from multiple orbitals, but becomes insufficient at larger scales, as can be seen in Fig. 4. Moreover, achievable communication bandwidth depends strongly on both the hardware architecture and the communication backend, making batching parameters that are optimal on one system suboptimal on another. This motivates a dynamic strategy that adapts batching parameters to the target platform.

The large number of orbitals processed concurrently in PW-DFT simulations naturally enables such optimization. By increasing the number of bands aggregated into each communication buffer for the FFT operations in Eq. (4), communications can be kept in the bandwidth-dominated regime, extending strong scaling to larger node counts for both `FFTXlib` and QE. The main limitations are the number of available bands and memory capacity.

Since QE workloads are dominated by repeated FFT operations, optimal batching parameters can be determined efficiently through lightweight autotuning. Unlike data-driven or AI-based approaches, this method requires neither training data nor retuning across architectures and software environments.

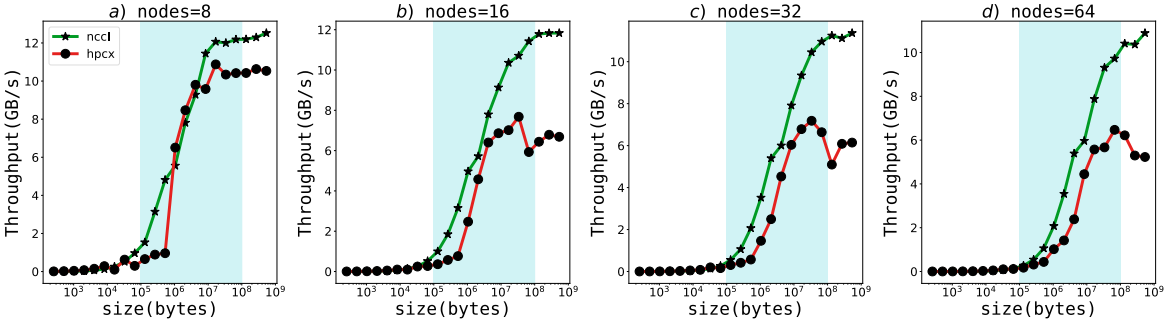


Fig. 5. Throughput of the All-to-All communication pattern on Leonardo Booster for 8, 16, 32, 64 nodes, on panels from *a*) to *f*). The stars connected by the green line refer to the NCCL implementation, while the dots connected by the red line refer to the MPI HPC-X one. The number of involved tasks is 4 times the number of nodes. On the x-axis, in log scale, the buffer size is reported, while on the y-axis the throughput is reported. The latter is defined as Throughput =(buffer size over all MPI ranks)/(time per iteration). The shaded region corresponds to the range of buffer sizes typically required by QUANTUM ESPRESSO workloads.

The autotuner is executed before the QE simulation and receives as input the maximum admissible batch size. Four sub-batches are retained to preserve overlap between communication and computation (Fig. 3). During tuning, only a single sub-batch is executed, since the runtime of the full batch can be inferred from the corresponding sub-batch time, reducing the cost of each measurement by roughly a factor of four.

Rather than exhaustively scanning all configurations, the autotuner employs a lightweight Bayesian optimization strategy based on *Thompson sampling* [28], which is robust against timing fluctuations caused by MPI communication and GPU scheduling noise. The search starts from a coarse grid of batch sizes and is progressively refined around the most promising regions. At each iteration, measured performances are used to build a Gaussian probability model that guides subsequent exploration toward the most likely optimum. Near-equivalent configurations are treated as identical to reduce redundancy and favor smaller, more memory-efficient batch sizes. In practice, a tolerance of about 10% from the best-performing configuration is used to identify performance plateaus and select the best compromise between memory footprint and communication efficiency. The autotuner typically converges in only a few iterations and introduces negligible overhead compared to production runs. The resulting batch size is then used throughout the QE simulation.

Fig. 6 compares inverse FFT time-to-solution obtained with fixed and adaptive batching using both HPC-X MPI and NCCL for three representative grid sizes relevant to QE simulations.

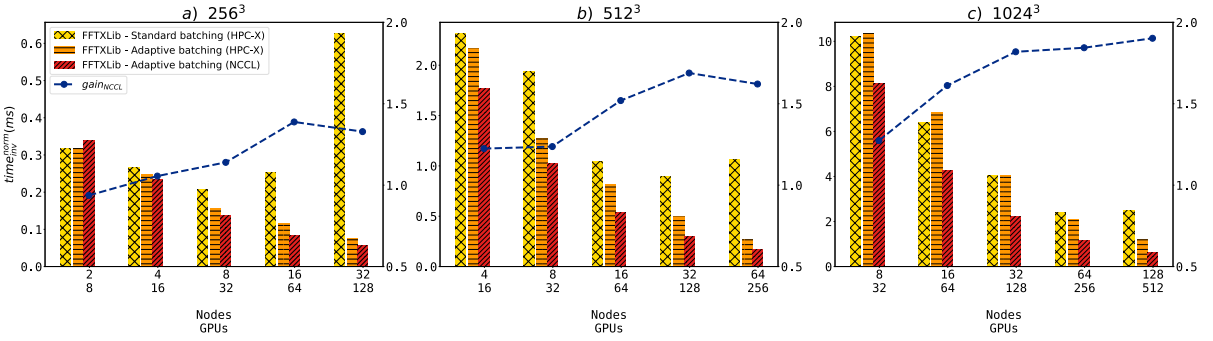


Fig. 6. Comparison between batched inverse FFT time to solution, normalized over the batch size. Different batching configurations are considered: standard batched set of 16 bands grouped into 4 sub-batches of 4 bands each (criss-crossed yellow bars) and adaptive batch/sub-batch size, both with HPC-X MPI communication library (horizontally striped orange bars) and NCCL (diagonally striped red bars). The blue dashed line (which refers to the y-axis on the right) represents the gain obtained by using NCCL over HPC-X MPI (defined as the ratio between the time to solution obtained with HPC-X MPI and the one obtained with NCCL) in the varying sub-batch size configuration. In panel *a*) to *d*) are reported the results for grids of dimension 256^3, 512^3 and 1024^3, respectively.

Beyond 8-16 nodes, fixed batching leads to a marked degradation in performance for the 256^3 and 512^3 grid sizes, whereas the adaptive strategy preserves strong scaling up to significantly larger node counts for both communication backends. `NCCL` consistently outperforms `HPC-X MPI` once the number of nodes becomes sufficiently large (typically beyond 8) and communication buffers remain above the bandwidth-dominated threshold identified in Fig. 5.

These results are consistent with the All-to-All benchmarks reported in Fig. 5, where FFT performance closely follows the ratio of the achieved communication bandwidths [Eq. (6)]. In particular, `FFTXlib` achieves speedups of approximately 1.4x for 256^3 grids at 16 nodes, 1.7x for 512^3 grids at 32 nodes, and 1.9x for 1024^3 grids at 64 and 128 nodes. Overall, the combination of adaptive batching and `NCCL` enables `FFTXlib` to effectively exploit the higher communication bandwidth available at large scale, which would otherwise remain underutilized in the standard implementation.

6. Application to Quantum ESPRESSO simulations on three EuroHPC systems

A Quantum ESPRESSO simulation exploits multiple levels of parallelism, from images and k-points down to plane waves, the latter being directly associated with Fast Fourier Transform (FFT) operations. While the interplay of these levels enables DFT simulations to scale toward exascale regimes, the present work focuses specifically on FFTs and therefore on plane-wave parallelism. Therefore we consider two representative Γ-point (single k-point) simulations with distinct electronic characteristics: a 1413-atom water system and a 1532-atom porphyrin-functionalized carbon nanotube (CNT10POR8), both using the SCAN meta-GGA exchange–correlation functional [23] as implemented in `Libxc` [29]. Meta-GGA functionals require additional FFT-intensive quantities, making them particularly suitable for assessing the impact of FFT optimizations. The water system is weakly interacting and FFT-dominated, whereas CNT10POR8 exhibits a more delocalized electronic structure, increasing the relative cost of the eigensolver. Together, these two benchmarks allow us to evaluate the benefits of the proposed optimizations in regimes where FFTs contribute differently to the overall execution time.

As discussed earlier, batching multiple orbitals together is essential to prevent communication from entering a latency-dominated regime at large node counts. Building on this observation, we extend the batching strategy to several parts of QE where FFTs were previously executed independently, including electronic density evaluation and the kinetic-energy density contribution to the Hamiltonian. In a second stage, we introduce the adaptive batching strategy of Sect. 5, which automatically selects the number of orbitals processed together so as to maintain communication in the bandwidth-dominated regime and maximize throughput. On NVIDIA-based systems, this optimization is further combined with the `NCCL` communication backend.

Benchmarks are performed on the NVIDIA-based Leonardo Booster and MareNostrum5 ACC systems, and on the AMD-based LUMI-G. Compared to earlier QE porting efforts on LUMI [30], substantial progress has been achieved, although further optimization is still required, particularly for the eigensolver, which remains CPU-based. On NVIDIA systems we use a software stack based on `NVHPC SDK v25.3` and `HPCX-MPI v2.22`, while on LUMI-G we employ Cray `CCE v20`, `ROCm v6.3`, and `MPICH v9.0.1`.

For each platform, we compare the standard QE v7.5 configuration with the two optimization stages described above. As shown in Fig. 7, both modifications provide substantial performance improvements across all systems. The gains are most pronounced for the water benchmark, where FFTs dominate the computational workload, but remain significant for CNT10POR8. MareNostrum5 ACC achieves nearly half the time-to-solution of Leonardo Booster owing to its higher InfiniBand bandwidth and newer H100 GPUs. On LUMI-G, optimized scaling approaches that of Leonardo for the water benchmark, whereas the gap remains larger for CNT10POR8 because of the increased weight of the eigensolver. Overall, these results provide a comprehensive comparison of QE simulations across NVIDIA- and AMD-based GPU systems and show that the performance gap is steadily narrowing.

7. Conclusions

The results presented in this work show that, although `FFTXlib` is less hardware-specialized than state-of-the-art distributed FFT libraries, it can outperform them in the range of workloads relevant to PW-DFT simulations by exploiting two application-specific features: the spherical cutoff in reciprocal space and distributed band batching. Together, these reduce both computational workload and communication overhead, allowing `FFTXlib` to operate

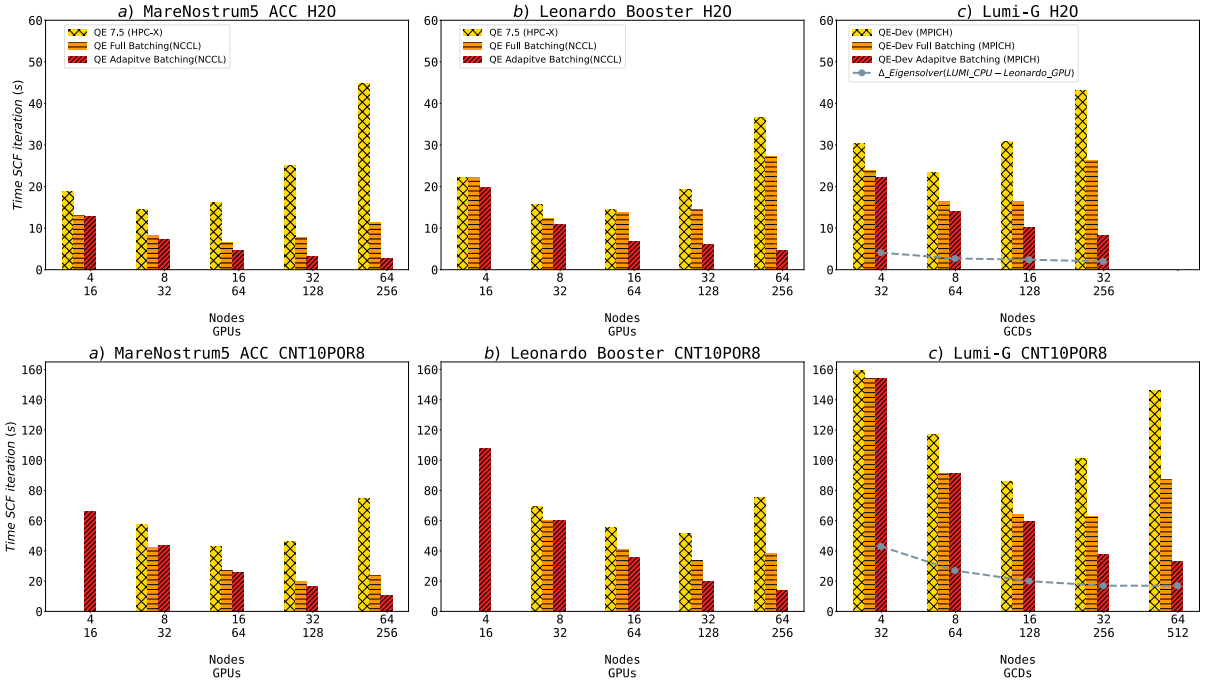


Fig. 7. Strong-scaling benchmark on three EuroHPC systems (MareNostrum 5 ACC, Leonardo Booster, and LUMI-G) for two Γ-point SCAN meta-GGA simulations: a 1413-atom water system (1884 bands, 320^3 grid points) and a 1532-atom porphyrin-functionalized carbon nanotube (CNT10POR8, 3139 bands, 512^3 grid points). Reported times are normalized by the number of self-consistent-field iterations. Yellow bars correspond to the reference QE version (QE 7.5 on NVIDIA systems and QE-Dev on AMD), orange bars to the first optimization stage (extension of fixed FFT batching), and red bars to the second optimization stage (adaptive FFT batching), where batching parameters are dynamically tuned to maximize communication throughput. The 64-node water run on LUMI-G is omitted because the number of plane-wave slabs exceeds the number of available MPI ranks. The 4-node runs on MareNostrum 5 and Leonardo with fixed batching are also omitted due to GPU memory limitations. Finally, the light grey plot in the LUMI graphs represents the difference in time due to the CPU-based eigensolver with respect to Leonardo GPU execution (as a reference).

efficiently across computation- and communication-dominated regimes and to trade FFT throughput in grid points for higher throughput in processed orbitals.

On modern GPU-based systems, this leads to FFT performance that depends weakly on the underlying band/plane-wave distribution at small node counts. Furthermore, the adaptive batching strategy introduced in this work dynamically matches communication buffer sizes to the characteristics of the communication backend, sustaining high bandwidth utilization and significantly improving strong scaling of plane-wave distributed executions.

The benefits are particularly evident for FFT-intensive workloads, such as meta-GGA and hybrid-functional calculations (the latter still under optimization), where orbital transforms account for a larger fraction of the overall cost. As these methods become increasingly important in large-scale materials simulations, the relevance of the proposed optimizations is expected to grow.

An additional advantage of the proposed approach is that adaptive batching relies on lightweight online autotuning rather than extensive offline training. This makes it complementary to machine-learning approaches for predicting QE execution parameters from simulation inputs. Because batching performance depends strongly on hardware and communication libraries, runtime tuning can adapt naturally to evolving HPC architectures without retraining.

Finally, the optimizations introduced in this work have also accelerated the porting of QE to AMD GPU systems. Results on LUMI-G show that performance is approaching that achieved on comparable NVIDIA-based platforms, despite the current use of CPU-based eigensolvers. These results highlight the increasing portability of QE across heterogeneous architectures and the effectiveness of FFT-centered optimizations as a key component of future exascale-ready PW-DFT simulations.

Acknowledgements & Reproducibility

This work was funded by the European Union through the MaX "MAterials design at the eXascale" Centre of Excellence for Supercomputing applications (Grant agreement No. 101093374, co-funded by the European High Performance Computing Joint Undertaking (JU) and participating countries). The authors acknowledge EuroHPC JU Development Access Call for granting computational resources on MareNostrum 5 ACC cluster (EHPC-BEN-2025B03-050 project) and LUMI-G cluster (EHPC-DEV-2024D08-033 and EHPC-DEV-2025D10-044 projects).
The author FFR acknowledges Pierre Carrier (HPE) and Alfio Lazzaro (HPE) for support while enabling compatibility of QE with Cray compiler.

The QE branches used in this work are available here: https://gitlab.com/QEF/q-e and here: https://gitlab.com/QEF/q-e-omp-repository.

References

[1] L. Fiedler et al. Deep dive into machine learning density functional theory for materials science and chemistry. *Phys. Rev. Materials*, 6:040301, 2022.
[2] J. Damewood et al. Representations of materials for machine learning. *Annu. Rev. Mater. Res.*, 53:399–426, 2023.
[3] P. Giannozzi et. al. Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys. Condens. Matter*, 21(39):1–19, sep 2009. 395502.
[4] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54:11169–11186, 1996.
[5] X. Gonze et al. Recent developments in the abinit software package. *Comput. Phys. Comm.*, 205:106–131, 2016.
[6] M.C. Payne et al. Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097, 1992.
[7] K. Sundararaman et al. Jdftx: Software for joint density-functional theory. *SoftwareX*, 6:278–284, 2017.
[8] P. Giannozzi et al. Advanced capabilities for materials modelling with Quantum ESPRESSO. *J. Phys. Condens. Matter*, 29(46):1–30, 2017. 465901.
[9] https://max-centre.eu/ (last access March 30 2026).
[10] https://admire-eurohpc.eu/ (last access March 30 2026).
[11] Yi Ju et al. In-situ techniques on gpu-accelerated data-intensive applications. In *2023 IEEE 19th International Conference on e-Science (e-Science)*, pages 1–10, 2023.
[12] https://dare-riscv.eu/ (last access March 30 2026).
[13] G. Agosta et al. Towards RISC-V-based HPC: the Italian pathfinding activities in the DARE-SGA1 project. pages 360–367, 09 2025.
[14] P.K. Yeung et al. GPU-enabled extreme-scale turbulence simulations: Fourier pseudo-spectral algorithms at the exascale using OpenMP offloading. *Computer Physics Communications*, 306:109364, 2025.
[15] J. A. Turner et al. Exaam: Metal additive manufacturing simulation at the fidelity of the microstructure. *The International Journal of High Performance Computing Applications*, 36(1):13–39, 2022.
[16] NVIDIA cuFFTMp documentation. https://docs.nvidia.com/hpc-sdk/cufftmp/index.html (last access Nov, 22nd 2024).
[17] heFFTe library. https://icl.utk.edu/fft/ (last access Dec, 6th, 2024).
[18] M. Wagner et al. Performance analysis and optimization of the FFTXlib on the Intel Knights landing architecture. In *2017 46TH International Conference on Parallel Processing Workshops (ICPPW)*, pages 243–250, 2017.
[19] J. Romero et al. A performance study of Quantum ESPRESSO's' PWscf code on multi-core and GPU systems. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pages 67–87, 2018.
[20] P. Giannozzi et al. Quantum ESPRESSO toward the exascale. *J. Chem. Phys.*, 152(15):1–11, 04 2020. 154105.
[21] I. Carnimeo et al. Quantum ESPRESSO: One further step toward the exascale. *Journal of Chemical Theory and Computation*, 19(20):6992–7006, 2023. PMID: 37523670.
[22] NVIDIA. NCCL. https://developer.nvidia.com/nccl (last access March 30 2026).
[23] Jianwei Sun et al. Strongly constrained and appropriately normed semilocal density functional. *Phys. Rev. Lett.*, 115:036402, 2015.
[24] W. Kohn and L.J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133, 1965.
[25] Yuwen Zhao et al. MFFT: A GPU accelerated highly efficient mixed-precision large-scale FFT framework. *ACM Trans. Archit. Code Optim.*, 20(3), July 2023.
[26] NVIDIA. NVSHMEM. https://docs.nvidia.com/nvshmem/release-notes-install-guide/index.html (last visited 30 March 2026).
[27] NVIDIA. HPC-X. https://developer.nvidia.com/networking/hpc-x (last visited 30 March 2026).
[28] D. J. Russo et al. A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11:1–96, 2018.
[29] S. Lehtola et al. Recent developments in libxc – a comprehensive library of functionals for density functional theory. *Software X*, 7:1–5, 2018.
[30] F. Ferrari Ruffino et. al. Quantum ESPRESSO towards performance portability: GPU offload with OpenMP. *Procedia Computer Science*, 240:52–60, 2024.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 147–154

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

Optimization of the PRECISION workflow for Computational Drug Design

Akash Deep Biswas[a,*], Andrew Emerson[b], Maria Gabriella Chiariello[c], Carmen Gratteri[c], Ingrid Guarnetti Prandi[d], Parisa Aletayeb[e], Alessandro Pedretti[e], Andrea Rosario Beccari[a], Carmine Talarico[a], Giorgia Frumenzio[b,**]

[a] *EXSCALATE, Dompé Farmaceutici S.p.A., Napoli, 80131, Italy*
[b] *CINECA, HPC Dept, Casalecchio di Reno, Bologna, 40033, Italy*
[c] *LIGHT S.c.a.r.l., Via Branze 45, Brescia, 25123, Italy*
[d] *Independent Researcher, Largo Martin Luther King, 21, Ciampino, Rome, 00043, Italy*
[e] *Dipartimento di Scienze Farmaceutiche, Università degli Studi di Milano, Milano 20133, Italy*

Abstract

The PRECISION (**PR**edictive **E**xploration of **C**orrelation **I**n **S**pecific **I**nteractions and **O**ptimization **N**etworks) project of Dompe' Farmaceutici (an Italian pharmaceutical company) aims to support the large-scale analysis of protein–ligand binding free energies through molecular dynamics (MD) simulations and high-performance computing (HPC) workflows. Building on the experience gained within the LIGATE project, in which more than 4000 protein–ligand complexes were simulated, this work focuses on overcoming the computational bottlenecks associated with the analysis of large MD datasets. Traditional methods like Molecular Mechanics/Poisson-Boltzmann Surface Area (MM-PBSA) and Molecular Mechanics/Generalized Born Surface Area (MM-GBSA) are often computationally demanding and difficult to scale efficiently for large screening campaigns. To address these limitations, here we present an optimized Binding Free Energy Exscalate (BFEx) workflow for HPC infrastructures by introducing an improved MPI-based parallelization strategy and refactoring the original Bash-based implementation into a modular Python framework. The optimized workflow exploits the computational resources of the Leonardo supercomputer to significantly reduce execution times, decreasing the analysis time from more than 24 hours to approximately 10 minutes for the tested systems. The proposed implementation improves the scalability, maintainability, and portability of the workflow, while also simplifying future developments such as GPU acceleration and the efficient processing of large MD simulation datasets for future applications, including machine learning-based analyses. In addition, the modular and portable design of the optimized workflow facilitates its deployment on different HPC infrastructures beyond the Leonardo supercomputer.

Keywords: Binding Free Energy; Protein-Ligand simulation; Drug Discovery; MPI; Python; HPC

* Corresponding author. Tel.: +39-345-444-7464.
E-mail address: akashdeep.biswas@dompe.com
** Corresponding author. Tel.: +39-051-6171446.
E-mail address: g.frumenzio@cineca.it

1877-0509
Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.026

Nomenclature

A	MM-PBSA: Molecular Mechanics/Poisson-Boltzmann Surface Area
B	MM-GBSA: Molecular Mechanics/Generalized Born Surface Area
C	BFEx: Binding Free Energy Exscalate

1. Introduction

In Computer Aided Drug Design (CADD), a potential drug molecule (or ligand) for a disease or condition can be identified by how well it binds to a protein involved in that condition. The extent of binding to the protein is called its "binding affinity" and when calculated for a collection of ligands can be used to find the best drug candidates in the list. One way to calculate protein-ligand binding affinities is by determining the free energy of binding which gives a measure of the stability of the protein-ligand complex when compared to the unbound state. Computational methods to do this include Molecular Mechanics/Generalized Born Surface Area (MM-GBSA) and Molecular Mechanics/Poisson–Boltzmann Surface Area (MM-PBSA) which provide a computationally framework to estimate the binding free energies and decompose them into residue-level contributions [1, 2]. Despite their widespread use, conventional MM-GBSA/MM-PBSA approaches often rely on ensemble-averaged quantities, which may hide temporary interactions that nonetheless play critical functional roles or overemphasize noise from weak contacts. To address this limitation, several studies have integrated additional descriptors, such as hydrogen bond occupancy, interatomic distances and solvent-accessible surface area (SASA) [3], to better characterize interaction stability and persistence [4, 5]. In particular, interaction lifetime (or occupancy) has been widely used as a proxy for stability, where high occupancy contacts are indicative of persistent and potentially functionally important interactions. Recent advances have emphasized the importance of integrating multiple descriptors to identify key binding determinants. For example, combining per-residue energy decomposition with geometric filtering (e.g., distance thresholds) and interaction persistence metrics has been shown to improve the identification of hotspot residues and reduce false positives arising from temporary contacts [6, 7]. Despite these advances, automated and scalable frameworks capable of integrating these analyses into high-throughput MD workflows remain limited.

In this study, we present an optimized computational pipeline for the automated high-throughput analysis of MD trajectories on HPC infrastructures. The original pipeline (described later in the Workflow description section) was originally automated using Bash scripts and the MPI implementations of MM-GBSA and MM-PBSA on one frame at a time from the molecular dynamics trajectory. The molecular dynamics data for the calculations come from a database obtained from the LIGATE project in which over 4038 protein-ligand complexes, with four replicas per complex, were simulated with GROMACS 2023.2 [8] to give a total dataset of over 16,000 MD trajectories (each 250ns in length). The large size of the dataset (approximately 40 TB) revealed the limited computational efficiency of the original workflow, highlighting the need for a more scalable and accelerated solution. The workflow's scientific reliability and numerical consistency were validated in a study from our group [9] on a dataset of 63 protein-ligand complexes, showing high correlation with experimental binding affinities and comparing favorably against standard methods. Consequently, this study leverages that established framework to specifically address performance and architectural scalability.

2. Workflow description

The original version of BFEx (i.e. before EPICURE support with EuroHPC resources) executed Bash codes integrating scripts from AmberTools [10] . The BFEx workflow starts with the following steps:

- integration of trajectory preprocessing and alignment
- MM-GBSA and MM-PBSA binding energy calculations
- per-residue decomposition

- interaction filtering

We now describe the workflow in detail (see Figure 1).

2.1. System Preparation and Trajectory Processing

All MD trajectories were processed using a combination of GROMACS, Amber [10] (cpptraj [11], MM-PBSA.py [2]), and in-house scripts. The initial topology and trajectory files were prepared from GROMACS simulations and converted into Amber-compatible formats for downstream analysis. To ensure consistency, trajectories were reprocessed using gmx_mpi grompp and trjconv, generating a filtered trajectory (pb_filtered.xtc) containing only the complex of interest. The trajectory was centered and periodic boundary conditions were removed prior to analysis. This preprocessing step ensured structural continuity and eliminated artifacts arising from periodic imaging. Residue indexing for protein and ligand components were extracted and subsequently used throughout the analysis pipeline to define interaction pairs and analysis regions.

2.2. Binding free energy calculations

Binding free energy calculations were performed using the MM-GBSA and MM-PBSA methods implemented in MM-PBSA.py.MPI from the Amber package. Calculations were conducted over segmented trajectory windows to capture temporal variations in interaction energetics. For each trajectory segment, binding free energies were computed using both the Generalized Born (GB) model with igb=5 and the Poisson–Boltzmann (PB) model with ionic strength of 0.1 M. The GB input configuration and decomposition settings were defined to enable per-residue energy decomposition, allowing identification of key interaction contributors. Similarly, PB calculations were configured with appropriate dielectric and ionic parameters.

2.3. Per-Residue Energy Decomposition and Interaction Filtering

To identify significant protein-ligand interactions, pairwise residue decomposition was performed for all protein residues against ligand residues. Only residue pairs satisfying a distance cutoff of 12 Å were considered for further analysis, reducing computational overhead and focusing on physically relevant interactions. For each residue pair, distance calculations were performed using cpptraj, averaging over selected trajectory frames. Residue pairs within the cutoff were subjected to: MM-GBSA decomposition analysis, MM-PBSA total energy calculation, SASA computation, and finally, hydrogen bond analysis. Distance filtering ensured that only energetically meaningful contacts were included in the final dataset.

2.4. Hydrogen bond and SASA analysis

Hydrogen bond interactions between protein and ligand residues were computed using cpptraj with intermolecular filtering enabled. Both total hydrogen bond counts and residue-wise interactions were calculated. Additionally, global hydrogen bond statistics were computed across the trajectory to determine interaction persistence. SASA calculations were performed separately for protein and ligand residues using a surface probe-based method. These values were normalized and used as descriptors of solvent exposure and interaction burial.

2.5. Structural Stability Analysis

To evaluate structural stability and conformational fluctuations, the following metrics were computed using cpptraj, Root Mean Square Deviation (RMSD) relative to the first frame, Root Mean Square Fluctuation (RMSF) per residue, and Radius of Gyration (RoG). These calculations were performed on backbone atoms (C, CA, N) to capture global and local structural dynamics.

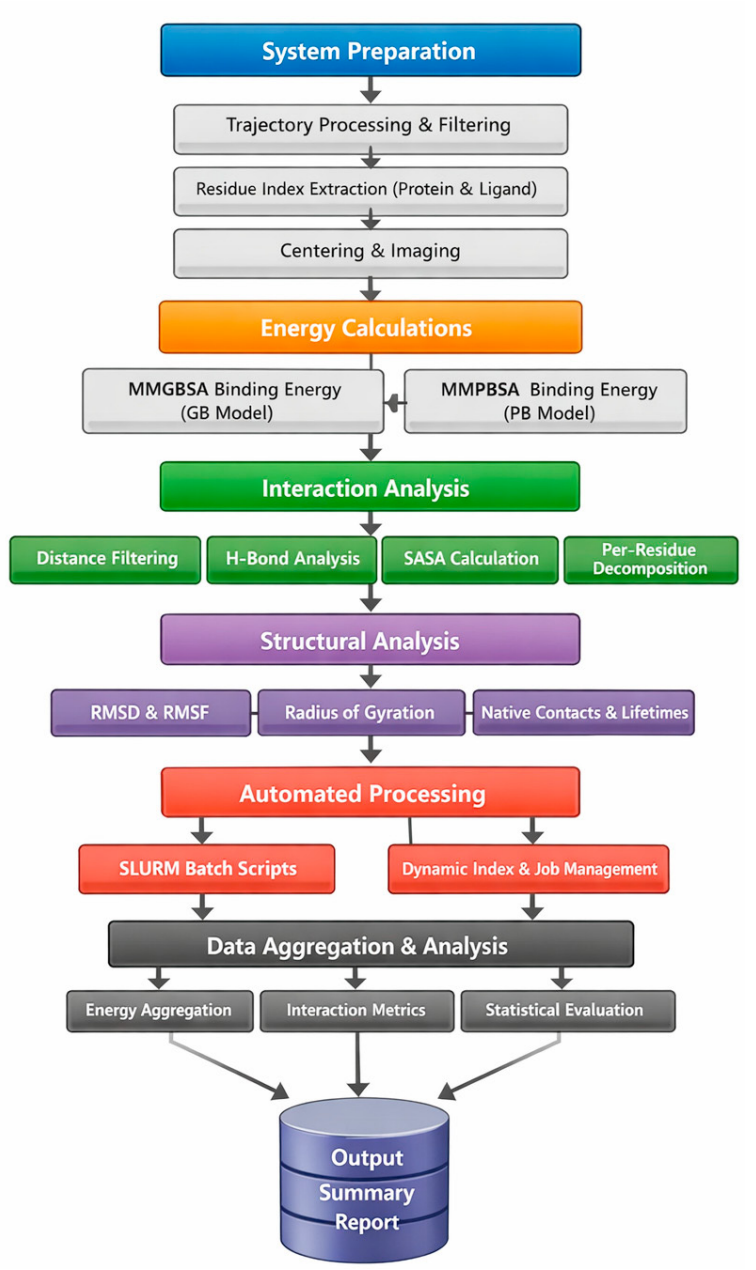


Fig. 1: The workflow of Binding Free Energy Exscalate (BFEx) version 1.0.

2.6. *Native contact and interaction persistence analysis*

Native contacts between protein and ligand residues were identified using a distance cutoff of 3.0 Å. Contact maps, residue-wise interactions, and time-series data were generated to quantify interaction persistence across the trajectory. Contact lifetimes were computed to assess the stability of individual interactions, providing insight into long-lived binding determinants.

2.7. Automated workflow execution

The entire workflow was automated using SLURM-based batch scripts to enable high-throughput processing on the Leonardo CPU-partition of CINECA. The job scripts performed sequential execution of preprocessing, energy calculations, and analysis, and the parallel execution of the MM-PBSA calculations. Automation scripts dynamically inserted residue indices and managed dependencies between steps, ensuring reproducibility and scalability.

2.8. Data aggregation and post-processing

All intermediate results were consolidated using custom Python scripts. Frame-wise interaction data were aggregated into structured datasets, which were then processed to compute, average decomposition energies, interaction lifetimes, distance and SASA averages and hydrogen bond frequencies. In addition, residue–ligand interaction pairs were grouped and statistical metrics calculated. These include, mean interaction energy, energy contributions filtered by interaction lifetime (>70%), and hydrogen bond-dependent energy contributions. This processing pipeline generate a final dataset summarizing all interaction metrics.

2.9. Statistical analysis

Post-processed data were further analyzed to extract global trends. Metrics that could be included are average and cumulative decomposition energies, lifetime-weighted interaction energies, hydrogen bond-dependent energy contributions, mean MM-GBSA and MM-PBSA binding energies, Filtering criteria (e.g., lifetime >70% of maximum) were applied to identify dominant interactions contributing to binding affinity. This 70% threshold follows the PANTHER framework [12] to strictly isolate the most persistent and dominant contacts throughout the trajectory. It is considered that any interaction which persist >30–40% is considered as a moderate interaction, whereas >50% as stable interaction and >70–80% as highly persistent/ strong interaction. Thus, by filtering interactions with lifetimes above 70%, transient thermal fluctuations are filtered out.

3. Workflow optimization overview

Here we present the refactoring and parallelization of the BFEx pipeline. The original implementation, based on a combination of mainly Bash but also some Python scripts organized in nested sequential loops, was functional but limited in efficiency and scalability: trajectory segments were processed serially, cpptraj input files were rewritten at each iteration, and no parallelism was exploited. In addition, the presence of large amounts of Bash code leads to portability issues.

These limitations made the application of the pipeline to large-scale datasets prohibitively time-consuming on modern HPC systems. To overcome these bottlenecks, the pipeline was completely re-implemented in Python and parallelized using the MPI programming model through the mpi4py library. The new design adopts a task-level parallel strategy over independent trajectory segments, enabling embarrassingly parallel execution across the available CPU cores of HPC nodes. The refactored pipeline was deployed and benchmarked on the Leonardo supercomputer at CINECA, demonstrating significant reductions in wall time and improved scalability with respect to both dataset size and system complexity.

4. Methods

Binding free energy calculations and interaction fingerprint extraction were performed using the BFEx pipeline, a refactored and fully parallelized Python workflow designed for large-scale analysis of molecular dynamics (MD) trajectories of protein–ligand complexes. For each trajectory, the pipeline computes MM-PBSA and MM-GBSA binding free energies, per-residue MM-GBSA energy decomposition, solvent accessible surface area (SASA), inter-atomic distances, and hydrogen bond occupancies. All trajectory analyses were carried out using the cpptraj program from the Amber simulation suite. The pipeline was rewritten in Python and parallelized using the MPI programming model via the mpi4py library, replacing the original serial implementation based on nested Bash/Python loops with sequential

cpptraj input rewriting. Parallelism is achieved through task-level decomposition over independent trajectory segments. By default, each trajectory is partitioned into various batches where 1 batch contains 10 frames, represented as a NumPy array of frame index sub-arrays. A default skip value of 40 frames is applied between consecutive batches. These sampling parameters are adopted from the PANTHER protocol [12] to reduce temporal autocorrelation inherent in consecutive MD simulation frames. Skipping 40 frames ensures statistical independence between sampled windows, while the 10-frame blocks capture accurate, local time-averaged contact behaviors. This approach balances computational efficiency and statistical relevant, and can also work as a noise cancellation strategy, although the parameters can be tuned according to user needs. An MPI communicator distributes segments across worker processes: each MPI rank receives the full trajectory and an assigned sub-array of frame indices, extracts the corresponding reduced trajectory, and independently performs the complete analysis suite, writing results to disk without inter-process communication. This embarrassingly parallel design ensures linear scalability with the number of available CPU cores. The optimized pipeline was applied to a test dataset of 4 protein-ligand complexes of different sizes, to evaluate scalability with respect to system size. All calculations were executed on the Leonardo supercomputer in CINECA, using the DCGP partition, whose nodes are each equipped with two Intel Xeon Platinum 8480+ processors (56 cores each, 112 cores per node total), providing the CPU-based parallelism required by the workflow. The software environment was managed with mamba and requires NumPy, Pandas, MDAnalysis, pytraj, and mpi4py Python packages. The Amber toolsuite (ambertools22) was installed within the same procedure to reduce the dependency on external system modules. A dedicated setup script is used to automate environment creation, package installation, and runtime configuration, ensuring full reproducibility across HPC systems and local development environments.

5. Results

To validate the new implementation, a series of benchmark tests were performed on four molecular dynamics (MD) simulations carried out using GROMACS (version 2023). The corresponding PDB codes of the selected systems are 6FHU, 5DA3, 5D0C, and 4KP8. Each trajectory, spanning 250 ns, was partitioned into shorter sub-trajectories. The benchmark systems were selected from a larger dataset of approximately 16,000 protein–ligand MD simulations. Since system size and trajectory length are the primary factors affecting computational performance, we chose four systems which span the range of system-sizes simulated. These systems, after removing solvent, contained 836, 4319, 6354, and 8172 atoms (see Table 1). For each trajectory, the following analyses were performed sequentially: (i) MM-PBSA calculation on the full complex; (ii) distance-based residue filtering; (iii) MM-GBSA calculation on both the full complex and the binding pocket residues; (iv) hydrogen bond analysis; and (v) solvent-accessible surface area (SASA) analysis. The MM-PBSA calculation was carried out considering all protein and ligand atoms, providing an estimate of the binding free energy of the entire complex. However, including all atoms becomes computationally demanding, particularly for per-residue MM-GBSA decomposition. To address this limitation, a distance-based filtering step was introduced as a preprocessing stage, restricting the per-residue analysis to binding pocket residues, typically defined as those within 10 Åof the ligand. The performance results are reported in Figure 2 and show a consistent improvement with increasing computational resources, as expected for an embarrassingly parallel strategy. For the smallest system (836 atoms), the execution time decreases from 15 minutes using 112 cores (corresponding to a full single node of the DCGP partition of Leonardo) to 4 minutes using 448 cores (four full nodes). For the larger system comprising 4319 atoms, execution times range from 25 to 8 minutes under the same scaling conditions. Similarly, for the analysis of the 5D0C system, the execution time is reduced from 43 to 8 minutes when increasing the number of cores from 112 to 448. Finally, for the largest system, 4KP8, the execution time decreases from approximately 58 to 12 minutes when scaling from one to four nodes.

To further quantify the benefits of parallel implementation, additional tests were conducted using a serial execution scheme, representative of the typical analysis workflows employed by widely used MD packages such as Amber and GROMACS. Although these packages are highly optimized for the simulation phase, their analysis tools are often largely serial, resulting in significantly long runtimes. In fact, it was found that the serial benchmarks yielded execution times of 7 hours for the smallest system and more than 24 hours for the larger systems (table1), corresponding to speedups of 28×, 72×, 74× and for 6FHU, 5DA3, 5D0C, 4KP8 respectively, achieved by using just a full single node of the Leonardo DCGP partition as shown in Figure 3.

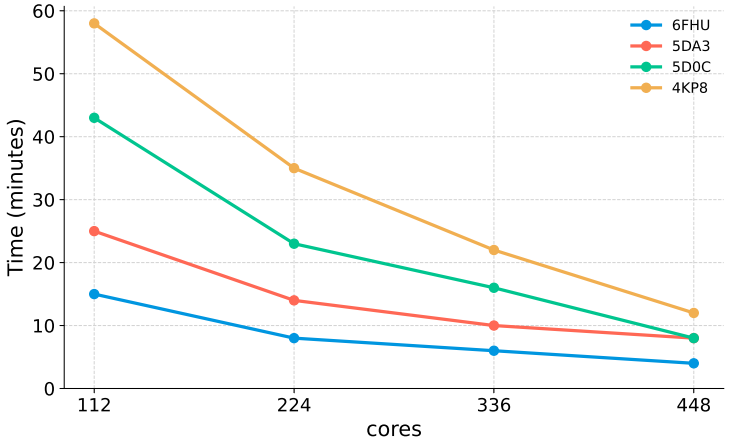


Fig. 2: Benchmark performance of four molecular systems (6FHU, 5DA3, 5D0C, 4KP8) as a function of the number of cores (112, 224, 336, 448). The y-axis reports the wall-clock time in minutes.

PDB ID	Number of atoms	Serial time (hours)
6FHU	836	7
5DA3	4319	30
5D0C	6364	53
4KP8	8172	72

Table 1: Baseline serial CPU simulation times for the selected PDB structures

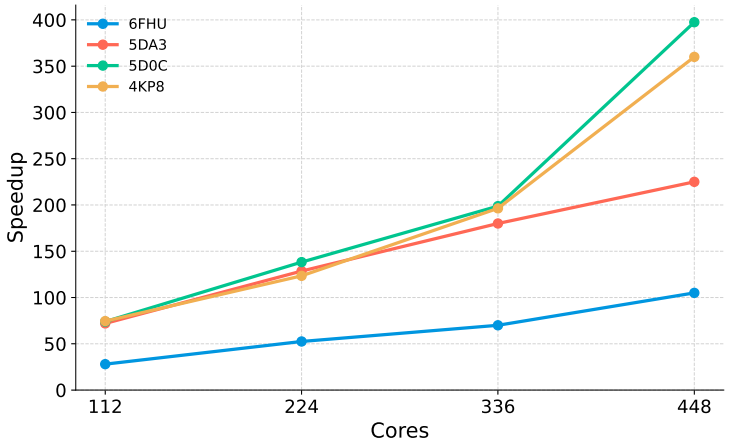


Fig. 3: Speedup of the MPI parallel implementation for four PDB structures as a function of the number of cores (112–448). Speedup is computed as the ratio between the serial CPU execution time and the parallel MPI execution time.

Overall, this parallel strategy has been show to accelerate the analysis of large-scale MD datasets, thereby enabling the efficient processing of large simulation datasets for computational applications in drug discovery.

6. Conclusions

The calculation of binding affinities is essential for understanding protein-ligand stability in computer aided drug design. However, the current workflows for calculating these properties are computationally inefficient for large datasets. But thanks to the resources provided by EuroHPC and the support from EPICURE it has been possible to significantly reduce the time required to run the workflows. In addition, the code has been restructured in a more professional and portable way, facilitating further enhancements and optimization activities.

Acknowledgements

We acknowledge the EuroHPC Joint Undertaking for awarding this project access to the EuroHPC supercomputer LEONARDO, hosted by CINECA (Italy) and the LEONARDO consortium, through the EuroHPC Development Access project EHPC-DEV-2024D11-037. In addition, we acknowledge EPICURE, a EuroHPC Joint Undertaking initiative, for supporting this project on LEONARDO through the above call.

References

[1] Kollman, P.A., Massova, I, Reyes, C, Kuhn, B, Huo, S, Chong, L, Lee, M, Lee, T, Duan, Y, Wang, W, and Donini, O. (2000) "Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models." *Accounts of chemical research* **19; (12) 33**: 889–97.

[2] Miller III, B. R., McGee Jr, T. D., Swails, J. M., Homeyer, N., Gohlke, H., and Roitberg, A. E. (2012) "MM-PBSA. py: an efficient program for end-state free energy calculations." *Journal of chemical theory and computation* **8** (9): 3314–3321.

[3] Richards, F. M. (1974) "The interpretation of protein structures: Total volume, group volume distributions and packing density." *Journal of Molecular Biology* **82** (1): 1–14.

[4] Kumari, R., Kumar, R., Open Source Drug Discovery Consortium, and Lynn, A.(2014) "g_mmpbsa-A GROMACS tool for high-throughput MM-PBSA calculations." *Journal of chemical information and modeling* **54** (7): 1951–1962.

[5] Hou, T., Wang, J., Li, Y., and Wang, W. (2011) "Assessing the performance of the MM/PBSA and MM/GBSA methods. 1. The accuracy of binding free energy calculations based on molecular dynamics simulations." *Journal of chemical information and modeling* **51** (1): 69-8-2.

[6] Wang, E., Sun, H., Wang, J., Wang, Z., Liu, H., Zhang, J. Z., and Hou, T. (2019) "End-point binding free energy calculation with MM/PBSA and MM/GBSA: strategies and applications in drug design." *Chemical reviews* **119** (16): 9478–9508.

[7] Sun, H., Li, Y., Tian, S., Xu, L., and Hou, T. (2014) "Assessing the performance of MM/PBSA and MM/GBSA methods. 4. Accuracies of MM/PBSA and MM/GBSA methodologies evaluated by various simulation protocols using PDBbind data set." *Physical Chemistry Chemical Physics* **16** (31): 16719–16729.

[8] Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., & Lindahl, E. (2015) "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers." *SoftwareX* **1** (): 19–25.

[9] Wingbermuhle, S., Biswas, A.D., Bonanni, D., Shugaeva, T., Gadioli, D., Beránek, J., Frumenzio, G., Querciagrossa, L., Piserchia, A., Accordi, G., Lunghini, F., et. al. (2026) "Molecular Dynamics Workflows to Compute Large-Scale Sets of Absolute Binding Free Energies Aiding Drug Candidate and Binding Pose Selection." *Journal of Chemical Theory and Computation* **22** (10): 1549–9618.

[10] Case, D. A., Aktulga, H.M., Belfon, K, Cerutti, D.S, Cisneros, G.A., Cruzeiro, V.W., Forouzesh, N., Giese, T.J., Gotz, A.W., Gohlke, H., and Izadi, S. (2023) "AmberTools." *Journal of Chemical Information and Modeling* **8;63** (20): 6183-91.

[11] Roe, D. R., & Cheatham, T. E., III (2013) "PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data." *Journal of Chemical Theory and Computation* **9** (7): 3084–3095.

[12] Aletayeb, P., Biswas, A.D., Rocca, S., Talarico, C., Vistoli, G., Pedretti, A. (2026) "PANTHER Score: Protein-Affinity for Nucleic Target-binding, Hybridization, and Energy Regression." *RNA* **1;32** (2): 131–49.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 155–164

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

AI4Land: A Scalable Deep Learning Framework on MareNostrum5 for Global High-Resolution Land Use Reconstruction

Amirpasha Mozaffari[a,*], Marina Castaño[a], Stefano Materia[a], Etienne Tourigny[a], Oscar Molina-Sedano[b], Jordi Varela-Agrelo[b], Dario Garcia-Gasulla[b], Miguel Castrillo Melguizo[a], Mario Acosta[a], Amanda Duarte[a]

[a]*Earth Science Department, Barcelona Supercomputing Center, Barcelona, 08034, Spain*
[b]*AI Institute, Barcelona Supercomputing Center, Barcelona, 08034, Spain*

Abstract

Uncertainty in the terrestrial carbon cycle remains a major constraint in climate projections, partly driven by the uncertainties affecting the land surface representation and variability in Earth system models. To address this limitation, we present a data-driven framework —AI4Land— for generating high-resolution historical reconstructions and future projections of key land surface variables. The framework follows a two-phase approach using a U-Net architecture. In the first phase, which is the focus of this work, it reconstructs annual land use and land cover by integrating coarse-resolution scenario data with static geophysical features. In a planned second phase, the resulting high-resolution maps will be used to predict dynamic biophysical variables, particularly leaf area index, at finer temporal scales. Trained on Earth observation data, the models learn to reproduce spatially explicit and physically consistent land surface patterns, extending temporal coverage to periods lacking direct observations. AI4Land was developed and trained on MareNostrum5, demonstrating how GPU-accelerated HPC infrastructure enables global-scale climate AI pipelines. The final product is a suite of open-source emulators designed for real-time coupling with digital twin platforms, such as those developed under the Destination Earth initiative. By delivering realistic and evolving land surface conditions on demand, this work aims to reduce critical uncertainties and improve the predictive power of next-generation climate simulations.

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026

Keywords: land use downscaling; deep learning; earth system models; semantic segmentation; digital twin

1. Introduction

The terrestrial carbon cycle remains a major source of uncertainty in climate projections [1], partly because land surface processes are not fully resolved. Land-use (LU) and land-cover (LC) changes are significant sources of uncertainty in estimating carbon fluxes [2]. Relying on coarse resolution and non-dynamic land boundary data can further lead to misrepresentation of land–atmosphere exchanges. Studies have shown that coarse land cover spatial resolutions

* Corresponding author.
E-mail address: amirpasha.mozaffari@bsc.es

Peer-review under responsibility of the scientific committee of the Proceedings of the fourth EuroHPC User Days 2026
10.1016/j.procs.2026.08.027

can introduce substantial biases in simulated terrestrial carbon sequestration, affecting its magnitude, interannual variability, and spatial distribution [3]. By contrast, providing high-resolution (≈1 km) land surface information, such as detailed land cover maps, leaf area index (LAI), or satellite vegetation indices like Normalized Difference Vegetation Index (NDVI), enhanced vegetation index (EVI), and soil-adjusted vegetation index (SAVI), allows climate models to capture fine-scale heterogeneity in vegetation and soil characteristics. This improved detail enhances the realism of carbon, water, and energy fluxes between the land and atmosphere. For example, using a new 1 km-resolution land parameter dataset in a land model produced pronounced spatial variability in soil moisture and surface fluxes (latent heat and radiation), whereas aggregating those inputs to ∼12 km led to a loss of about 31–54 % of the spatial information [4]. High-resolution vegetation data are particularly crucial, since changes in LAI or greenness directly affect processes like photosynthesis (carbon uptake), evapotranspiration, and surface energy balance. For instance, a drop in LAI reduces canopy shading, increases ground net radiation, and dries out soil moisture [5, 6]. Accurate representation of high-resolution land boundary conditions in climate models is essential for reducing uncertainties in the terrestrial carbon cycle and for improving simulations of coupled carbon-water-energy fluxes [2]. Complementing this, advances in remote sensing of vegetation provide critical observational constraints that enhance the monitoring, understanding, and modelling of these land-atmosphere interactions [7]. Several datasets offer valuable insights into land surface boundaries, but each falls short in terms of spatial resolution, temporal (historical or future) and global coverage, or continuity [8, 9, 10, 11, 12, 13]. Bridging this gap at a global scale requires not only advanced deep learning methodology but also GPU-accelerated HPC infrastructure to make training and inference computationally tractable. Building on these insights and datasets, our work aims to reduce uncertainties in terrestrial carbon cycle representation by developing high-resolution land surface boundary datasets that span historical, contemporary, and future periods. By integrating satellite-era observations with reconstructions for pre-observation times and projections for observation-limited futures, we provide temporally continuous, spatially detailed datasets for use in climate and weather models. These datasets were produced using EuroHPC compute allocations on MareNostrum5 at BSC and are designed to improve the representation of land surface heterogeneity, enabling more accurate simulations of carbon, water, and energy fluxes over time.

2. Methods

To generate temporally continuous, spatially detailed land surface boundaries, we focus in this work on reconstructing the slow-varying land surface boundary condition LU/LC at high spatial resolution; a planned second phase will extend the framework to dynamic biophysical variables such as LAI.

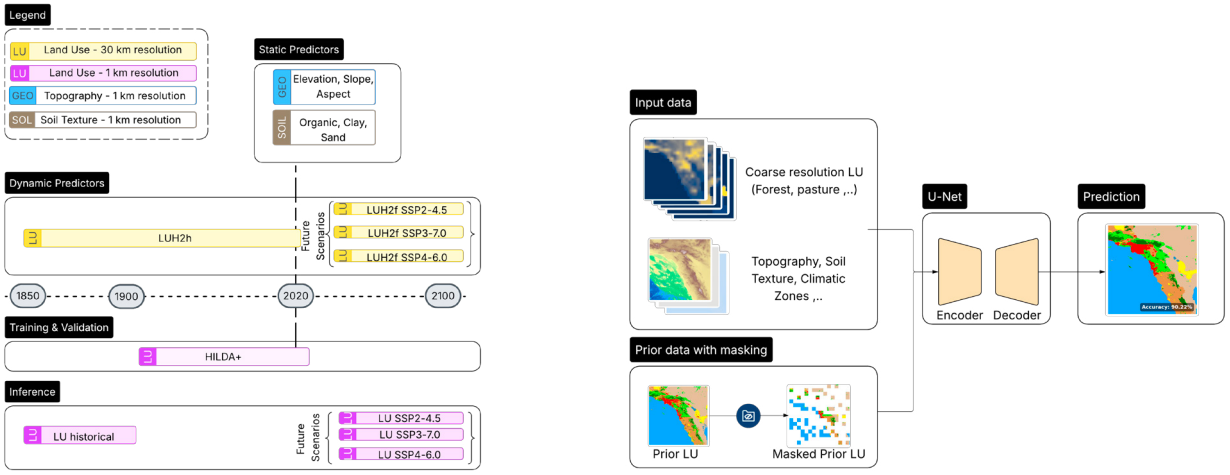
We have generated historical and future annual LU at high spatial resolution (1 km), derived from coarse-resolution inputs (0.25°, ∼28 km). The goal is to produce high-resolution LU maps for a target year t (H_t), with each pixel classified into predefined categories (e.g., Forest, Cropland, Urban). Ground-truth labels are sourced from the high-resolution HILDA+ LU/LC dataset [10]. While the original HILDA+ classification scheme contains 13 classes, we consolidated these into 8 distinct LU categories to reduce class imbalance and streamline the learning objectives. The input tensor for year t is a multichannel stack comprising Land-Use Harmonization 2 (LUH2) data for year t [8] providing 12 fractional LU variables and two land surface parameters standardized to unit variance on a $0.25° \times 0.25°$ (∼31 km) grid, static topographic features (elevation, slope, and aspect) [14], high-resolution soil characteristics including clay, sand, and organic content from PNNL global datasets [4], and an autoregressive prior from year $t - 1$ or $t + 1$ to enable bidirectional prediction.

2.1. Preprocessing

To ensure consistency across diverse data sources, we established a standardized preprocessing pipeline relying on the Climate Data Operators (CDO) software [15]. All inputs were remapped using nearest-neighbour interpolation to align with the HILDA+ WGS84 grid at 1 km resolution. For soil data, a non-linear depth-weighted average ($w = [0.40, 0.25, 0.15, 0.10, 0.05, 0.05]$) was applied to prioritize root-zone interactions. The final processed tensors were stored in Analysis-Ready Cloud Optimized (ARCO) Zarr format [16], chunked in 512×512-pixel blocks for efficient parallel I/O during distributed training.

2.2. Model Architecture

We implemented the LU reconstruction and projection using a U-Net architecture [17] for 512×512-pixel samples. Since all inputs are aligned to a WGS84 longitude–latitude grid at 1 km resolution, each patch covers approximately 512×512 km at the equator, and per-pixel classification is performed independently via dense semantic segmentation. The model employed a standard U-Net with 35 base channels, accepting inputs from 14 LUH2 variables across 2 time steps, 6 static features, and 1 HILDA prior. Its encoder consisted of four max-pooling and double-convolution blocks, mirrored by a symmetric decoder with upsampling and skip connections, terminating in a 1×1 convolution for final segmentation. To encode the categorical target, we utilized entity embeddings [18]. During training, 60% of the autoregressive prior is randomly masked to prevent the model from trivially copying the high-resolution prior and force it to learn the mapping from the coarse LUH2 inputs and static features, which is essential for inference in periods without ground truth. As we produce both historical and future projections, we have trained two separate models: one for historical data (using LUH2h) and one for future projections (using LUH2f).



(a) Spatio-Temporal Data Scope

(b) Reconstruction Pipeline

Fig. 1: The AI4Land Framework. (a) Data Availability: The timeline illustrates the critical resolution gap: while coarse dynamic forcing (LUH2) covers the full centennial scope (1850–2100), high-resolution ground truth (HILDA+) is limited to the satellite era. We fuse these dynamic inputs with high-resolution static priors to reconstruct the missing historical and future boundaries. (b) Reconstruction Pipeline: The U-Net backbone fuses heterogeneous inputs: coarse dynamic variables are upsampled and concatenated with fine-scale static features. To enforce temporal consistency, a randomly masked land use map from the adjacent time-step ($t \pm 1$) is injected as an autoregressive prior.

2.3. Distributed Training on MareNostrum5

We trained the model on MareNostrum5 using Distributed Data Parallelism (DDP) via the Hugging Face Accelerate library [19], scaling from a single node (4 NVIDIA H100 GPUs) up to 8 nodes (32 H100 GPUs). In each epoch, we trained on 10,000 batches of 8 samples and validated on 2,000 batches of 8 samples, with the training dataset comprising 800,000 carefully selected samples. Each epoch on a single node required approximately 2 hours and 40 minutes, including training, validation, and I/O overhead.

To quantify the scalability of the pipeline, we conducted a weak scaling analysis in which the per-node workload was kept constant while the number of nodes was increased from 1 to 8. As shown in Figure 2, the DDP training pipeline achieves near-linear scaling throughout this range, retaining 98.5%, 97.4%, and 97.7% of ideal throughput at 2, 4, and 8 nodes, respectively. At maximum scale (8 nodes, 32 H100 GPUs), the system sustains approximately 300 samples per second, confirming that inter-node communication overhead on MareNostrum5 remains negligible.

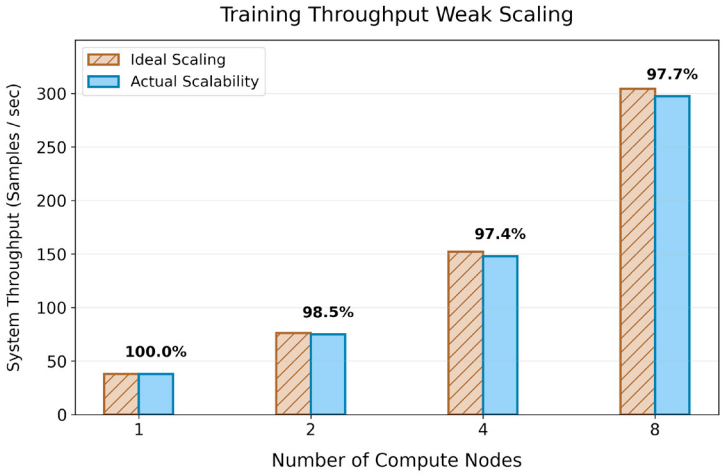


Fig. 2: Weak scaling throughput of the AI4Land DDP training pipeline on MareNostrum5. System throughput (samples/sec) is shown for 1, 2, 4, and 8 compute nodes alongside ideal linear scaling. Parallel efficiency remains above 97% up to 8 nodes (32 H100 GPUs), demonstrating near-linear scalability of the Hugging Face Accelerate-based distributed training.

2.4. Evaluation

We employed a two-phase evaluation strategy combining Grid-Based Partitioning and Farthest Point Sampling (FPS) to eliminate spatial autocorrelation leakage between training and testing sets. The spatial domain is divided into a coarse grid of 30×30 bins, with entire grid cells assigned exclusively to training, validation, or testing sets. Within these regions, FPS selects maximally distant points to ensure uniform spatial coverage. From approximately 214 million land pixels, we subsampled a final dataset of 448,000 points, stratified into 320,000 training, 64,000 validation, and 64,000 testing samples. To prevent data leakage, we applied a temporal split: years 1960–2000 were used for training and 2001–2015 for testing.

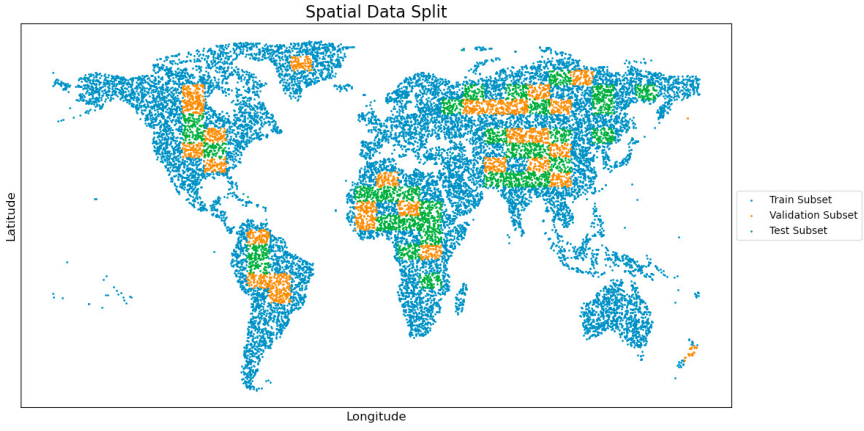


Fig. 3: Spatial distribution of the training, validation, and test sets. Grid-based partitioning assigns entire macro-grid cells exclusively to each split, ensuring strict physical separation and eliminating spatial autocorrelation leakage between training and testing sets.

2.5. Inference

To generate spatially consistent global land use maps, the system employs a distributed inference pipeline processing the globe in overlapping 512×512 patches. To generate seamless global maps, we employ a sliding window approach with a Gaussian weighting function W to merge overlapping predictions. The final probability $\bar{\mathbf{P}}(x, y)$ is computed by normalizing the weighted sum of overlapping patches $\Omega_{(x,y)}$:

$$\bar{\mathbf{P}}(x, y) = \frac{\sum_{k \in \Omega} \mathbf{p}^{(k)}(x, y) \cdot W(u_k, v_k)}{N(x, y)} \tag{1}$$

The final class is derived via $\arg\max_c$, ensuring smooth transitions across patch boundaries without blocky or sharp border artifacts.

3. Results

3.1. Model Performance

The U-Net architecture, a cornerstone of our LU classification methodology, was rigorously trained on a dataset comprising 800,000 carefully selected samples. As we have to produce both historical and future projections, we have tackled this effort with two models: one for predicting historical data and one for predicting future data. Since the performance metrics for the historical projection model and the future projection model are nearly identical, we use the average performance metrics for both models. Following this extensive training phase, the model's performance was assessed, yielding a mean Intersection over Union (mIoU)—the average across classes of the intersection-over-union between predicted and ground-truth pixel sets—of 0.805. This metric corresponds to an overall classification accuracy of 94.67%, achieved under a specific configuration that involved a 60% masking strategy during the evaluation.

Figure 4 shows training and validation loss across epochs for all node configurations (1, 2, 4, and 8 nodes). In all cases the training loss remains substantially higher than the validation loss, which is explained by the use of input masking during training whereas validation is performed on unmasked inputs. Notably, increasing the node count improves convergence per epoch: after 10 epochs the 8-node configuration achieves a training loss of approximately 0.29, compared to 0.40 for the single-node baseline, with validation loss similarly improving from $\approx$0.12 to below 0.10. This behaviour is consistent with the larger effective batch size under multi-node DDP, which provides a better gradient estimate per update step. Combined with the near-linear throughput scaling shown in Figure 2, this confirms that multi-node training on MareNostrum5 delivers both faster and better-converged solutions.

There is a performance disparity that was dramatically amplified for those LU types that were underrepresented in the training dataset. For these minority classes, the model's ability to generalize and accurately predict decreased substantially, resulting in a notable drop in the IoU, in the urban class, to 0.46. This outcome underscores the model's sensitivity to class imbalance and highlights the need for further strategies to improve generalization across all LU types, particularly those with fewer training examples. Table 1 shows the accuracy (%), F1 score, and IoU across the classes.

Table 1: **Global Evaluation Results.** Per-class segmentation performance of the AI4Land U-Net model. Average loss: 0.1434.

Class	Accuracy (%)	F1	IoU
Urban	48.68	0.632	0.463
Cropland	90.62	0.897	0.815
Pasture	89.52	0.897	0.814
Forest	94.93	0.941	0.889
Grass/shrubland	81.47	0.833	0.715
Other land	97.05	0.971	0.944
Water	99.24	0.995	0.991

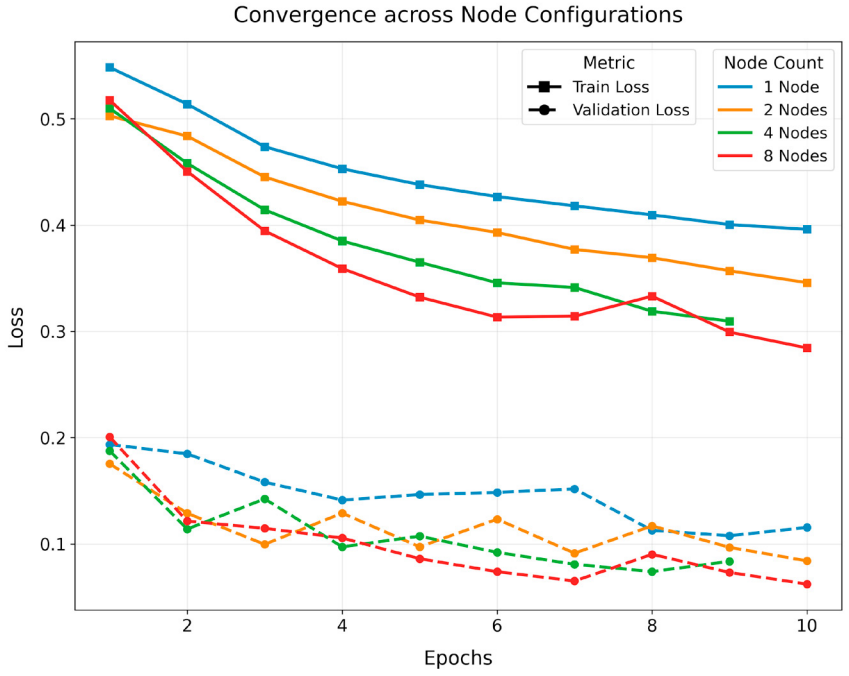


Fig. 4: Convergence across node configurations on MareNostrum5. Solid lines with square markers denote training loss; dashed lines with circle markers denote validation loss, for 1, 2, 4, and 8 nodes. Configurations with more nodes consistently reach lower loss at equivalent epoch counts, with the 8-node run (32 H100 GPUs) achieving a final training loss of ≈0.29 versus ≈0.40 for the single-node baseline.

To provide a concrete illustration of the model's output and the complexities of the classification task, Figure 5 presents a qualitative example. The figure illustrates the multi-band input channels provided to the U-Net, the ground truth masked prior used for training and evaluation, and the final resulting model predictions. Importantly, these visual examples are provided for two consecutive years, 2004 and 2005, which helps to qualitatively assess the model's consistency and ability to handle temporal variations within the classification task.

3.2. Global Inference

The execution of this pipeline produced a comprehensive high-resolution dataset spanning the period 1850–2100, stored in both ARCO Zarr and NetCDF formats. The primary historical output integrates model predictions for the pre-observation era (1850–1899) with consolidated HILDA+ ground truth for the observational record (1899–2020). Furthermore, the system successfully generated three distinct future projection cubes corresponding to the SSP2-4.5, SSP3-7.0, and SSP4-6.0 climate scenarios. Validation of the inference stage, specifically for the year 2014 as shown in Figure 6, demonstrated high fidelity to the ground truth, achieving a global accuracy of 94.88% and a mIoU of 0.8569, confirming the framework's ability to accurately reconstruct complex land-use patterns on a global scale.

4. Discussion and Future Work

Existing global land-use datasets trade off spatial resolution against temporal coverage: HILDA+ [10] offers high resolution but only spans the satellite era, LUH2 [8] covers 850–2100 but at coarse resolution, and Chen et al. [13] provide global 0.050.05°projections limited to 2015–2100. ML-based downscaling has been demonstrated for specific regions or single countries [20, 11, 21], but to our knowledge no model has yet been created that provides a LU dataset bridging this gap at global scale. AI4Land addresses this scaling mismatch by learning the statistical mapping from coarse LUH2 forcing to high-resolution HILDA+ patterns, producing a consistent, high-resolution (1 km) LU dataset covering the historical period (1850–2020) and three future scenarios (up to 2100).

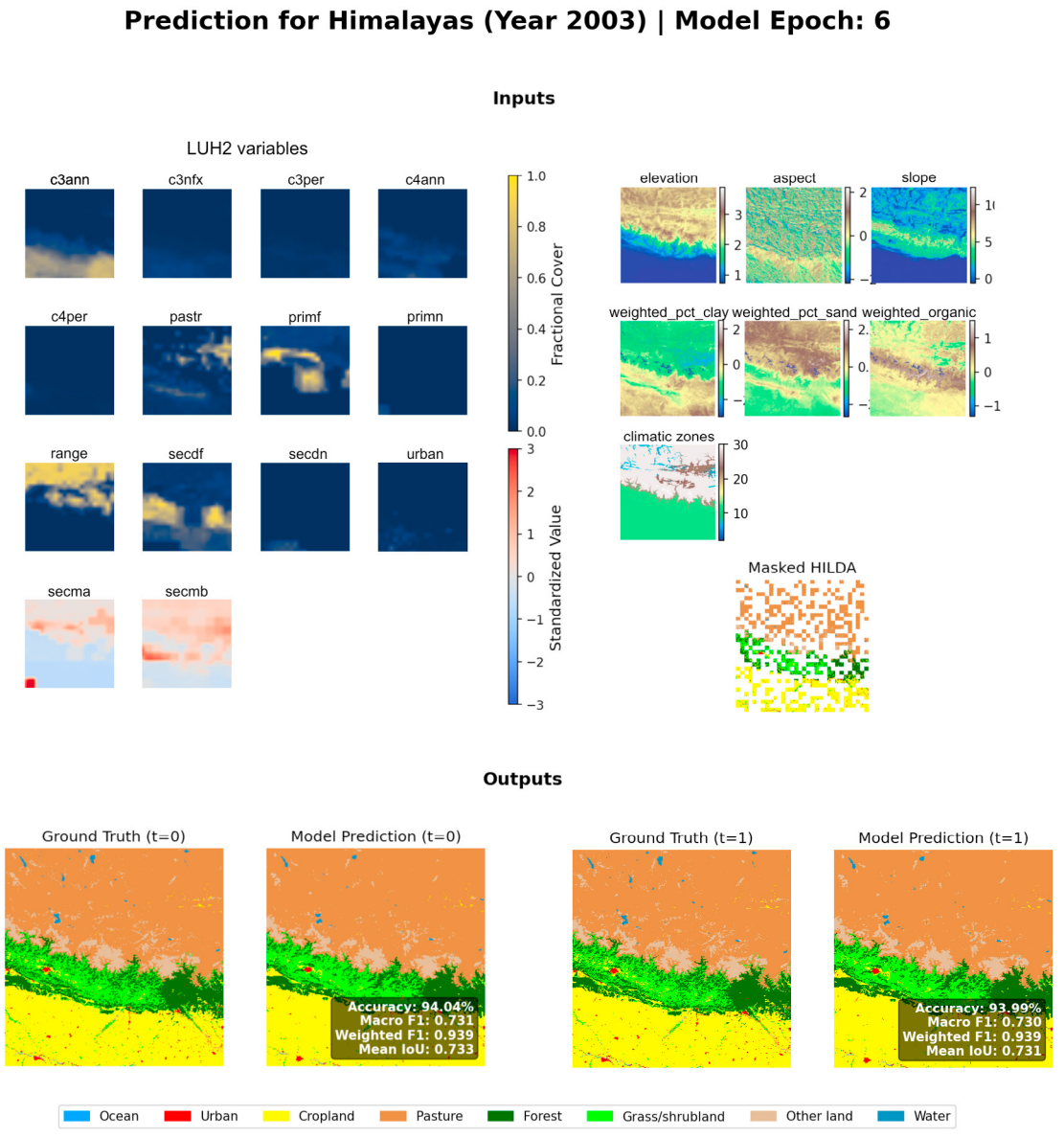


Fig. 5: Qualitative results for the Himalayas region (Year 2003). The figure displays the full stack of input channels, including 12 coarse fractional LU inputs (top-left), 6 high-resolution static variables (middle-left), and the masked HILDA prior (bottom-left). The model's multi-step predictions (right) for two time steps ($t = 0$, $t = 1$) show strong agreement with the ground truth, achieving ~94% accuracy and a Mean IoU of ~0.73.

While the underlying U-Net with distributed data-parallel training builds on well-established components, the primary contribution of this work is the demonstration that such a workflow can be deployed on EuroHPC infrastructure to produce a scientifically useful dataset at global scale. The distributed training pipeline achieved parallel efficiency

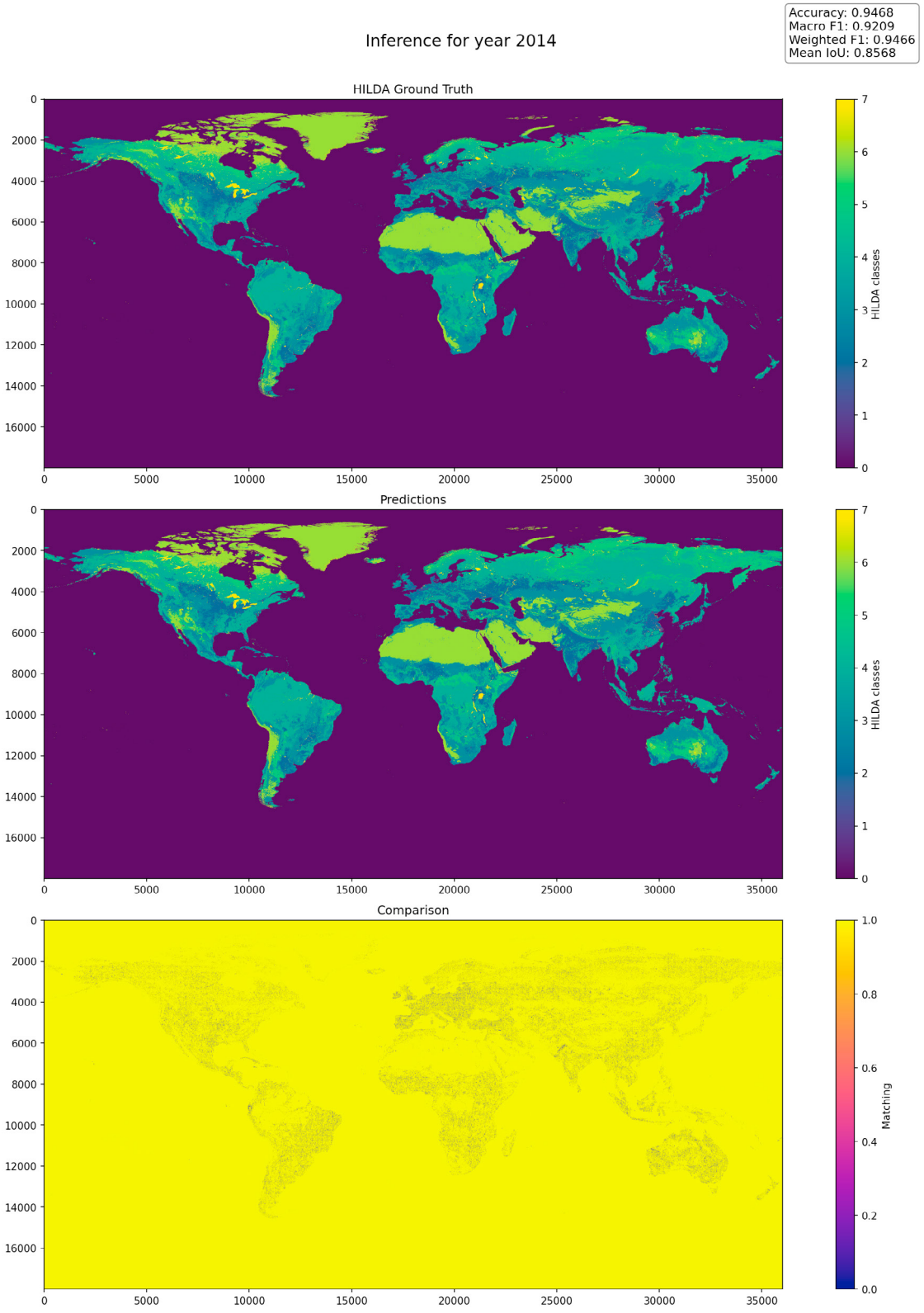


Fig. 6: Worldwide inference results (Year 2014). The figure shows the ground truth LU data (top), the predictions with our U-Net model (middle), and a heat map of the similarities between these two (bottom).

above 97% up to 8 nodes (32 H100 GPUs) on MareNostrum5; while this represents only a modest fraction of the full system, it indicates that the workflow scales efficiently and provides a foundation for larger-scale runs in future iterations. Qualitative analysis confirms the model's ability to maintain temporal consistency and smooth spatial transitions without edge artifacts. We acknowledge that the LU dataset is inherently impacted by the quality and biases of the source data, specifically the uncertainty in HILDA+ (ground truth) and the forcing data from LUH2. Consequently, our model reflects these inherent biases. While global accuracy is high (94.88%), performance dropped for underrepresented classes, specifically urban areas.

Several directions are planned to address these limitations and extend the AI4Land framework. First, to improve urban and minority class classification, we are incorporating dynamic annual Köppen-Geiger climatic zones (1 km) [22] and ISIMIP2b population density data [23] as additional predictors. These additions are expected to significantly improve the urban class classification by providing richer anthropogenic and climatic context.

Second, to better enforce temporal consistency across centennial-scale projections, we are integrating a Recurrent U-Net that extends the standard U-Net into a recurrent framework, unrolling the model over multiple time steps. The prediction at step t serves as the "historic prior" input for step $t + 1$. To address exposure bias during long rollouts, we employ scheduled sampling: training starts with ground-truth priors for stability, then linearly decays to rely on self-predictions, ensuring robustness against accumulated errors over the 1850–2100 projection period. Third, to capture long-range spatial dependencies that CNNs may miss, we plan to investigate generative approaches, specifically Flow Matching U-Nets with Vision Transformer (ViT) backbones. These architectures will enable probabilistic realizations of future land use, quantifying uncertainty and capturing long-range dependencies more effectively than deterministic baselines, providing a robust foundation for Digital Twin initiatives. Building on the results of phase one, the second phase of the framework extends to predict continuous biophysical variables characterizing vegetation state. Specifically, we aim to generate high-resolution maps of LAI_t for a given time step t (monthly or sub-monthly). Finally, we are exploring the possibility of leveraging foundation models — through adapters and fine-tuning — as a replacement for our from-scratch trained models, expecting them to outperform the latter by inheriting capabilities already encoded in the foundation model.

In this study, we introduce the first components of the AI4Land framework for high-resolution, temporally consistent estimation of historical and future land surface boundary conditions essential for weather and climate prediction. We present LU/LC models that generate high-resolution, temporally consistent data with expanded temporal coverage spanning 1850–2100. Future work will focus on incorporating additional predictors to improve performance on under-represented classes. To ensure transparency, reproducibility, and broad accessibility, all datasets, models, code, and pretrained weights will be released as open-source, supporting continued innovation in Earth system modeling.

Acknowledgements

This work received funding from the European Union's Horizon Europe Framework Programme through the project CONCERTO (Grant Agreement 101185000), TerraDT (Grant Agreement 101187992) and ELLIOT (Grant Agreement 101214398). AM acknowledges Grant JDC2023-051208-I, funded by MICIU/AEI/10.13039/501100011033. AD and SM acknowledge their AI4S fellowship within the "Generación D" initiative by Red.es. This research was supported by the EuroCC National Competence Centre. We acknowledge EuroHPC Joint Undertaking for awarding project ID EHPC-DEV-2025D03-112 and EHPC-AIF-2025SC01-004 access to MareNostrum5 at BSC, Spain. The authors thank the organizers of the Open Hackathon EuroCC AI Hackathon, especially Milos Maric and Simeon Harrison.

Appendix A. Additional Experiments

References

[1] B. B. Booth, C. D. Jones, M. Collins, I. J. Totterdell, P. M. Cox, S. Sitch, C. Huntingford, R. A. Betts, G. R. Harris, J. Lloyd, High sensitivity of future global warming to land carbon cycle processes, Environmental Research Letters 7 (2) (2012) 024002.

[2] B. K. Gier, M. Schlund, P. Friedlingstein, C. D. Jones, C. Jones, S. Zaehle, V. Eyring, Representation of the terrestrial carbon cycle in cmip6, Biogeosciences 21 (22) (2024) 5321–5360.

[3] S. Zhao, S. Liu, Z. Li, T. L. Sohl, A spatial resolution threshold of land cover in estimating terrestrial carbon sequestration in four counties in georgia and alabama, usa, Biogeosciences 7 (1) (2010) 71–80.
[4] L. Li, G. Bisht, D. Hao, L. R. Leung, Global 1 km land surface parameters for kilometer-scale earth system modeling, Earth System Science Data 16 (4) (2024) 2007–2032.
[5] S. Boussetta, G. Balsamo, A. Beljaars, T. Kral, L. Jarlan, Impact of a satellite-derived leaf area index monthly climatology in a global numerical weather prediction model, International journal of remote sensing 34 (9-10) (2013) 3520–3542.
[6] H. Fang, F. Baret, S. Plummer, G. Schaepman-Strub, An overview of global leaf area index (lai): Methods, products, validation, and applications, Reviews of Geophysics 57 (3) (2019) 739–799.
[7] I. C. Prentice, M. Balzarolo, K. J. Bloomfield, J. M. Chen, B. Dechant, D. Ghent, I. A. Janssens, X. Luo, C. Morfopoulos, Y. Ryu, et al., Principles for satellite monitoring of vegetation carbon uptake, Nature Reviews Earth & Environment 5 (11) (2024) 818–832.
[8] G. C. Hurtt, L. Chini, R. Sahajpal, S. Frolking, B. L. Bodirsky, K. Calvin, J. C. Doelman, J. Fisk, S. Fujimori, K. K. Goldewijk, et al., Harmonization of global land-use change and management for the period 850–2100 (luh2) for cmip6, Geoscientific Model Development Discussions 2020 (2020) 1–65.
[9] Copernicus Global Land Service / EEA, Leaf Area Index 1999–2020 (raster 1 km), global, 10-daily – version 2, temporal coverage: 1999–2020; spatial resolution: raster 1 km; dekadal (every 10 days) (2020). doi:10.2909/d5fdc595-2e03-4cbe-a39e-5f006f9cef07.
URL https://land.copernicus.eu/en/products/vegetation/leaf-area-index-v2-0-1km
[10] K. Winkler, R. Fuchs, M. Rounsevell, M. Herold, Hilda+ global land use change between 1960 and 2019 [dataset]. pangaea (2020).
[11] M. Luo, G. Hu, G. Chen, X. Liu, H. Hou, X. Li, 1 km land use/land cover change of china under comprehensive socioeconomic and climate scenarios for 2020–2100, Scientific data 9 (1) (2022) 110.
[12] J. Lv, Y. Gao, C. Song, L. Chen, S. Ye, P. Gao, Land system changes of terrestrial tipping elements on earth under global climate pledges: 2000–2100, Scientific Data 12 (1) (2025) 163.
[13] M. Chen, C. R. Vernon, N. T. Graham, M. Hejazi, M. Huang, Y. Cheng, K. Calvin, Global land use for 2015–2100 at 0.05 resolution under diverse socioeconomic and climate scenarios, Scientific Data 7 (1) (2020) 320.
[14] G. C. Group, Gebco 2024 grid, Distributed by GEBCO, British Oceanographic Data Centre, accessed on 27 July 2025 (2024). doi:10.5285/1c44ce99-0a0d-5f4f-e063-7086abc0ea0f.
URL https://www.gebco.net/data-products/gridded-bathymetry-data
[15] U. Schulzweida, Cdo user guide (2019). doi:10.5281/zenodo.3539275.
[16] D. J. Newman, Zarr storage specification version 2: Cloud-optimized persistence using zarr, Technical report, NASA Earth Science Data and Information System Standards Coordination Office (2024). doi:10.5067/DOC/ESCO/ESDS-RFC-048v1.
URL https://doi.org/10.5067/DOC/ESCO/ESDS-RFC-048v1
[17] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, 2015, pp. 234–241.
[18] C. Guo, F. Berkhahn, Entity embeddings of categorical variables (2016). arXiv:1604.06737.
URL https://arxiv.org/abs/1604.06737
[19] S. Gugger, L. Debut, T. Wolf, P. Schmid, Z. Mueller, S. Mangrulkar, M. Sun, B. Bossan, Accelerate: Training and inference at scale made simple, efficient and adaptable., https://github.com/huggingface/accelerate (2022).
[20] J. Yang, B. Tao, H. Shi, Y. Ouyang, S. Pan, W. Ren, C. Lu, Integration of remote sensing, county-level census, and machine learning for century-long regional cropland distribution data reconstruction, International Journal of Applied Earth Observation and Geoinformation 91 (2020) 102151.
[21] V. T. Truong, S. Hirayama, D. C. Phan, T. T. Hoang, T. Tadono, K. N. Nasahara, Jaxa's new high-resolution land use land cover map for vietnam using a time-feature convolutional neural network, Scientific Reports 14 (1) (2024) 3926.
[22] H. E. Beck, T. R. McVicar, N. Vergopolan, A. Berg, N. J. Lutsko, A. Dufour, Z. Zeng, X. Jiang, A. I. van Dijk, D. G. Miralles, High-resolution (1 km) köppen-geiger maps for 1901–2099 based on constrained cmip6 projections, Scientific data 10 (1) (2023) 724.
[23] K. Frieler, S. Lange, F. Piontek, C. P. Reyer, J. Schewe, L. Warszawski, F. Zhao, L. Chini, S. Denvil, K. Emanuel, et al., Assessing the impacts of 1.5 c global warming–simulation protocol of the inter-sectoral impact model intercomparison project (isimip2b), Geoscientific Model Development 10 (12) (2017) 4321–4345.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 286 (2026) 165–185

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the fourth EuroHPC User Days 2026

The European High Performance Computing Joint Undertaking Infrastructure and Access Calls State-of-Play

Krishnakshi Bhuyan[a], Laura Martínez-Calvo[a], Annika Kossack[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 the European High Performance Computing Joint Undertaking (EuroHPC JU) infrastructure and its evolving access ecosystem supporting High-Performance Computing (HPC), Quantum Computing and Artificial Intelligence (AI) across Europe. It provides an overview of the EuroHPC supercomputing ecosystem, highlighting recent deployments, upgrades and the introduction of Europe's first operational exascale and hybrid quantum-HPC systems. The paper further reviews the EuroHPC JU access framework, including traditional HPC Access Calls, AI Factory Access Calls and the newly launched Quantum Access Pilot Call, outlining their objectives, evaluation procedures and target user communities. Based on proposal and allocation data collected between 2021 and May 2026, the paper analyses proposal submissions, success rates, awarded computational resources, research domains distribution, Principal Investigator demographics and geographical participation across the different access calls. The results illustrate the sustained growth in demand for EuroHPC resources, particularly for AI-driven applications, and demonstrate how the introduction of AI Factory access mechanisms and the Quantum Access Pilot Call are expanding support for scientific research and industrial innovation, supporting academia, start-ups, SMEs, commercial companies and the public sector. The presented analysis provides a comprehensive overview of the evolution of the EuroHPC JU access framework and its role in enabling European scientific excellence, technological innovation and digital sovereignty through a world-class computing infrastructure.

Keywords: High Performance Computing (HPC); Infrastructure; Supercomputers; Artificial Intelligence (AI); AI Factories; Quantum Computers; Access calls; HPC resources; node hours; research domains

Corresponding authors
E-mail addresses: Krishnakshi.BHUYAN@eurohpc-ju.europa.eu (Krishnakshi Bhuyan), Klara.MESTROVIC@eurohpc-ju.europa.eu (Klara Meštrović).

10.1016/j.procs.2026.08.028

1. The European High Performance Computing Joint Undertaking (EuroHPC JU)

The 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 38 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 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 JU 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; two (2) mid-range systems – Arrhenius in Sweden and DAEDALUS in Greece; three (3) pre-exascale systems – LUMI in Finland, Leonardo in Italy, and MareNostrum 5 in Spain; and two (2) exascale systems – JUPITER in Germany and Alice Recoque in France (currently under deployment). In 2025, access to the JUPITER exascale system was made available to users, expanding the list of operational EuroHPC JU supercomputers. Additionally, Discoverer was upgraded to Discoverer+ with enhanced performance and AI capabilities, while Leonardo was upgraded to the Leonardo Improved Supercomputing Architecture (LISA), introducing an AI-optimized partition to support large language models and multimodal generative AI. Other EuroHPC systems also received incremental upgrades to software environments, storage, and AI frameworks as part of ongoing infrastructure evolution across the EuroHPC ecosystem.

The latest editions of the TOP500 and Green500 lists, released on 23 June 2026, once again featured all operational EuroHPC supercomputers, including the two new systems, DAEDALUS and Arrhenius. Europe's first exascale supercomputer, JUPITER—the first system in Europe to surpass one quintillion calculations per second—remains among the world's five most powerful supercomputers, ranking fifth in the TOP500. LUMI and Leonardo also continue to demonstrate Europe's leadership in high performance computing, securing 11th and 12th place, respectively, and maintaining two EuroHPC systems in the global top 20. Beyond performance, EuroHPC systems continue to set the benchmark for energy efficiency. JUPITER retains its position as the world's most energy-efficient exascale supercomputer, while DAEDALUS, Arrhenius and several other EuroHPC systems achieved leading positions in the Green500 rankings, underlining the EuroHPC JU's strong commitment to sustainable high performance computing.

JUNE 2026	TOP500	Green500
JUPITER	#5	#17
LUMI	#11	#45
LEONARDO	#12	#90
MARENOSTRUM 5	#16	#58
DAEDALUS	#31	#23
ARRHENIUS	#42	#24
MELUXINA	#177	#99
KAROLINA	#250	#96
DISCOVERER	#317	#233
DEUCALION	#353	#137
VEGA	#365	#277

Figure 1: EuroHPC JU supercomputers in the June 2026 TOP500 and Green500 rankings

Apart from these developments, Artificial Intelligence is now a key pillar of the EuroHPC Joint Undertaking's mission to strengthen Europe's leadership in high-performance computing and emerging technologies. Through its AI Factories initiative, the EuroHPC JU is creating a dynamic innovation ecosystem centered on AI-optimized supercomputers, providing advanced computing resources, data, expertise and tailored support services to European start-ups, SMEs, industry and the scientific community. Alongside the deployment of new systems, the initiative also supports the enhancement of existing EuroHPC supercomputers with advanced AI capabilities, accelerating the development and uptake of trustworthy artificial intelligence across Europe. Since September 2024, the EuroHPC JU has launched successive cut-offs under its calls for proposals to support the establishment of AI Factories across Europe. To date, hosting agreements have been signed for 19 AI Factories: LUMI-AI (Finland), HammerHAI and JAIF (Germany), PHAROS (Greece), IT4LIA (Italy), L-AIF (Luxembourg), BSC AIF and 1HealthAI (Spain), MIMER (Sweden), AI:AT (Austria), BRAIN++ (Bulgaria), AI2F (France), PIAST AI (Poland), SLAIF (Slovenia), CZAI (Czechia), LitAI (Lithuania), NLAIF (the Netherlands) and RO AI (Romania). Together, these facilities form a pan-European network that will strengthen Europe's AI ecosystem and reinforce its position as a global leader in artificial intelligence, advanced computing and innovation. Most sites will deploy new AI-optimized supercomputers, while others will build on or upgrade existing EuroHPC systems, including JUPITER, Alice Recoque, MareNostrum 5 and DAEDALUS.

Further, EuroHPC JU completed a call for proposals for the establishment of 13 AI Factory Antennas linked to selected AI Factories across Europe. The AI Factory Antennas provide participating countries with remote access to AI-optimized supercomputing resources, expertise, training, and support services hosted by their linked AI Factory. They also strengthen national AI ecosystem by supporting academia, industry, and public sector in developing and deploying trustworthy AI applications. Where appropriate, AI Factory Antennas may also provide small-scale AI computing resources for the fine-tuning, testing, and validation of AI applications. The initiative enables EuroHPC JU Participating States to benefit from AI Factory resources and services without investing in a full-fledged AI Factory, thereby promoting a more inclusive and geographically balanced European AI infrastructure.

In addition to this, the EuroHPC JU has invested in its quantum computing fleet comprised of 10 systems offering 6 different qubit modalities, deployed across mainland Europe. The current fleet entails the trapped-ion system 'Piast-Q' operated by PCSS in Poznan (Poland), the superconducting qubit system with a dedicated star-shaped topology 'VLQ' operated by IT4I in Ostrava (Czech Republic), the superconducting qubit system 'Euro-Q-Exa' operated by LRZ in Munich (Germany), the photonic qubit system 'Lucy' operated by CEA/GENCI in Paris (France), the quantum annealer 'EuroQCS-Spain' operated by BSC in Barcelona (Spain) and the neutral atom system 'SOL' operated by CINECA in Bologna (Italy). Besides, European end-users will also be granted access to the two quantum simulators 'RUBY' and 'JADE', which were procured within the pilot project 'HPCQS'. All eight systems have already been inaugurated and their technical details can be found on the EuroHPC dedicated website - Our Quantum Computers - The European High Performance Computing Joint Undertaking (EuroHPC JU). The EuroHPC JU is actively working on expanding its fleet to incorporate novel spin qubit systems, such as 'EuroSSQ-HPC' that will be operated by SURF (The Netherlands) and 'MeluXina-Q', which will be operated by LuxProvide (Luxemburg).

To connect its expanding infrastructure, the EuroHPC Joint Undertaking developed the EuroHPC Federation Platform (EFP), a unified, secure platform that provides seamless access to its supercomputers through a single user interface and authentication system. It integrates the computational, data analysis, software, and storage capabilities of EuroHPC Hosting Entities, providing a common environment for accessing distributed resources while supporting integrated workflows, resource management, and data handling across the EuroHPC ecosystem.

3. Access to the EuroHPC JU Infrastructure

The EuroHPC JU offers a variety of access opportunities within nine (9) access calls that are designed to address different user needs. 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 that was open for one year. With the announcement of the AI Factories in 2025, the JU restructured its existing calls into Calls for Traditional HPC Applications and additionally introduced Calls for AI Applications (AI Factories Access calls). Calls for AI Applications include 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. Across all mentioned access calls, the EuroHPC JU has awarded 3,792 proposals and 187,924,938 node hours in the period of December 2021 – May 2026.

In June 2026, EuroHPC JU launched its first call for accessing quantum infrastructure, the Quantum Access Pilot Call, marking the first time that European researchers, public institutions and industry can request free access to EuroHPC quantum computers and simulators. The pilot call supports testing, algorithm and workflow development, performance evaluation, and quantum computing training, enabling users to experiment with multiple quantum technologies integrated with Europe's supercomputing infrastructure as part of the EuroHPC hybrid quantum-HPC strategy.

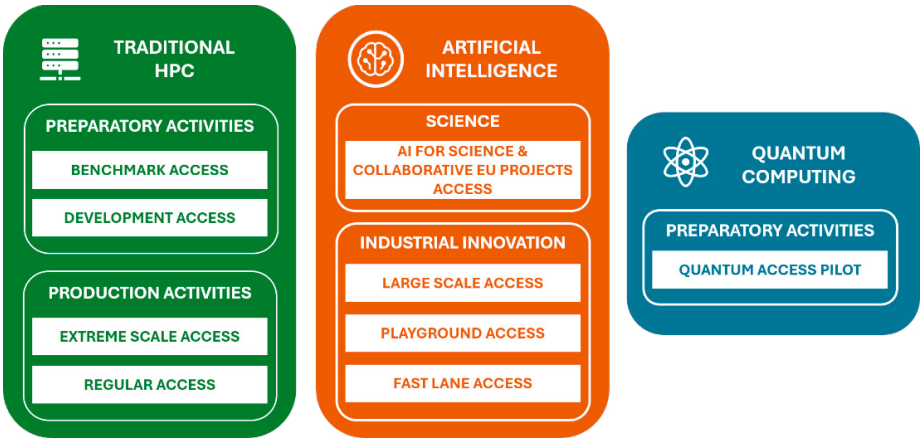


Figure 2: EuroHPC JU Access Calls available in 2026

3.1. Access Calls for Traditional HPC Applications

The EuroHPC JU Access calls for Traditional HPC Applications define the different modalities through which EuroHPC JU computing resources are made available to support traditional HPC workflows across a broad range of scientific domains and fields of application, based mainly on classical HPC usage. These access calls 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 computing resources of its operational systems to support traditional HPC applications across four access calls: Extreme Scale Access, Regular Access, Development Access, and Benchmark Access.

Across all cut-offs from December 2021 to May 2026, these four access calls received a total of 3,639 proposals, resulting in 3,074 awarded projects and the allocation of more than 167 million node hours to successful applicants. The distribution of proposals and awarded node hours across the different access calls is presented in the table below.

Access call	Proposals awarded	Proposals awarded %	Node hours awarded	Node hours awarded %
Extreme Scale Access (Dec 2022-Oct 2025)	162	5.3%	100,856,071	60.2%
Regular Access (Dec 2021-Mar 2026)	419	13.6%	51,715,942	30.8%
Development Access (Jan 2022-May 2026)	1,624	52.8%	10,981,663	6.6%
Benchmark Access (Jan 2022-May 2026)	869	28.3%	4,055,367	2.4%
TOTAL	**3,074**	**100%**	**167,609,043**	**100%**

Table 1:EuroHPC JU awarded projects and resources throughout Traditional HPC Access Calls in the period December 2021 – May 2026

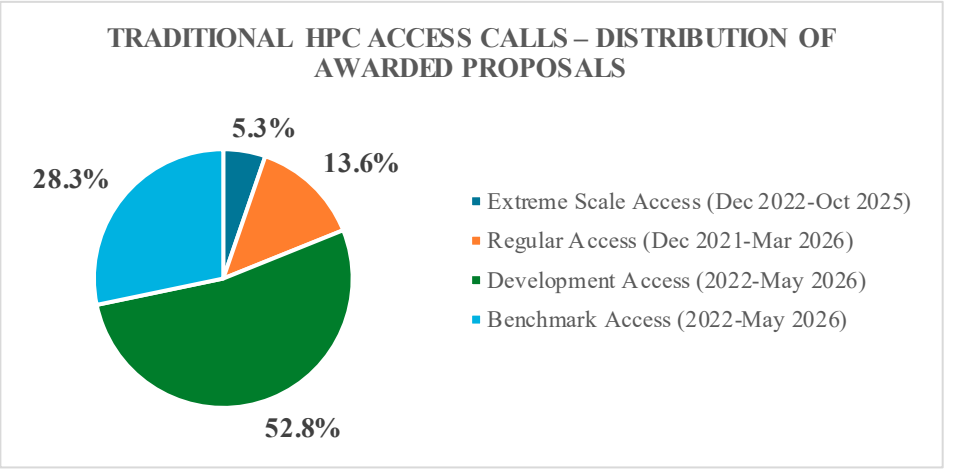


Figure 3: Share of awarded proposals across the different Traditional HPC Access Calls

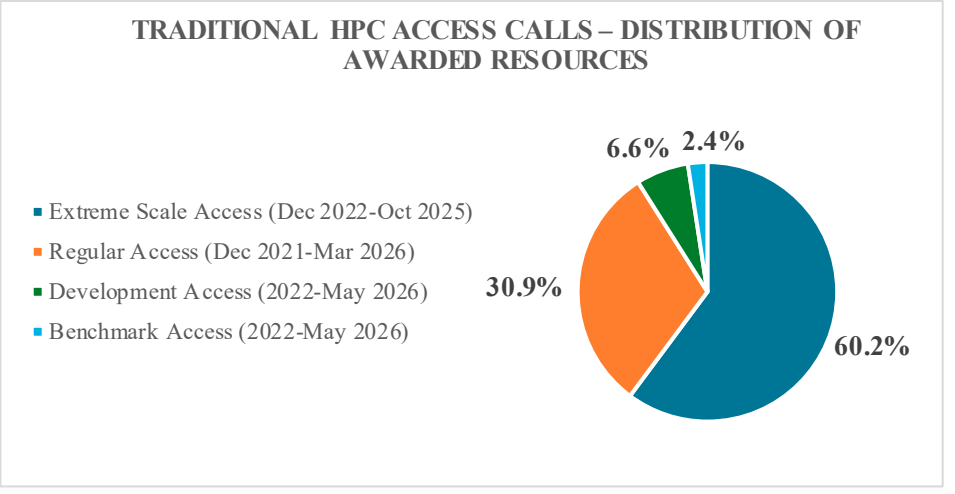


Figure 4: Share of awarded resources (percentage of node hours) across the different Traditional HPC Access Calls

Partition	Extreme Scale Access	Regular Access	Development Access	Benchmark Access
Vega CPU	-	6,267,147	396,000	295,000
Vega GPU	-	421,322	152,100	24,900
MeluXina CPU	-	4,282,032	673,713	275,500
MeluXina GPU	-	1,809,264	374,400	58,600
Karolina CPU	-	3,131,881	989,000	229,367
Karolina GPU	-	511,750	466,400	65,500
Discoverer CPU	-	6,006,173	364,500	237,000
Discoverer GPU	-	-	25,950	1,800
LUMI-C	18,260,260	10,726,672	1,482,000	731,500
LUMI-G	31,630,100	3,069,383	1,505,000	883,500
Leonardo Booster	24,488,847	3,930,150	2,290,500	675,900
Leonardo DCGP	3,558,580	2,219,524	357,000	76,000
MareNostrum 5 GPP	11,915,962	5,334,982	418,500	162,500
MareNostrum 5 HBM	-	-	56,000	14,000
MareNostrum 5 ACC	4,596,888	886,542	1,039,500	269,500
Deucalion ARM	-	-	176,000	18,000
Deucalion x86	-	615,214	96,000	30,000
Deucalion GPU	-	110,080	36,000	2,800
JUPITER Booster	6,405,434	2,193,877	66,500	4,000
Arrhenius CPU	-	30,000	-	-
Arrhenius GPU	-	169,950	-	-
TOTAL NODE HOURS	**100,856,071**	**51,715,942**	**10,981,663**	**4,055,367**

Table 2: EuroHPC JU Awarded resources (node hours) in different partitions throughout Traditional HPC Access Calls in the period December 2021 – May 2026

Among the 3,074 awarded projects within Traditional HPC Access Calls, 1,698 projects (55.53%) are within the Engineering, Mathematics and Computer Sciences domain. This is followed by Chemical Sciences and Materials, Solid State Physics with 461 projects (15.00%), Computational Physics: Universe Sciences, Fundamental Constituents of Matter with 429 projects (13.96%), Biochemistry, Bioinformatics, Life Sciences, Physiology and Medicine with 257 projects (8.36%), and Earth System Sciences & Environmental Studies with 186 projects (6.05%), and the Socio-Economic Sciences and Humanities: Economics, Finance and Management, Linguistics, Cognition and Culture account for 34 projects (1.11%).

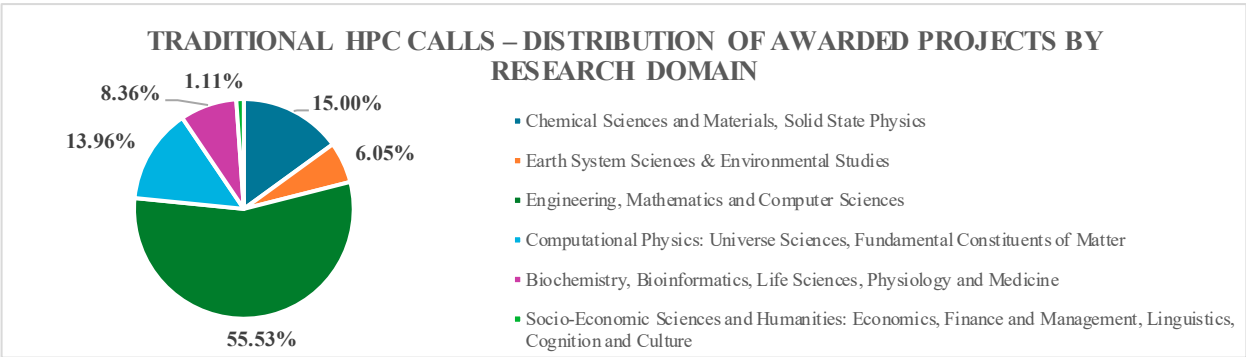
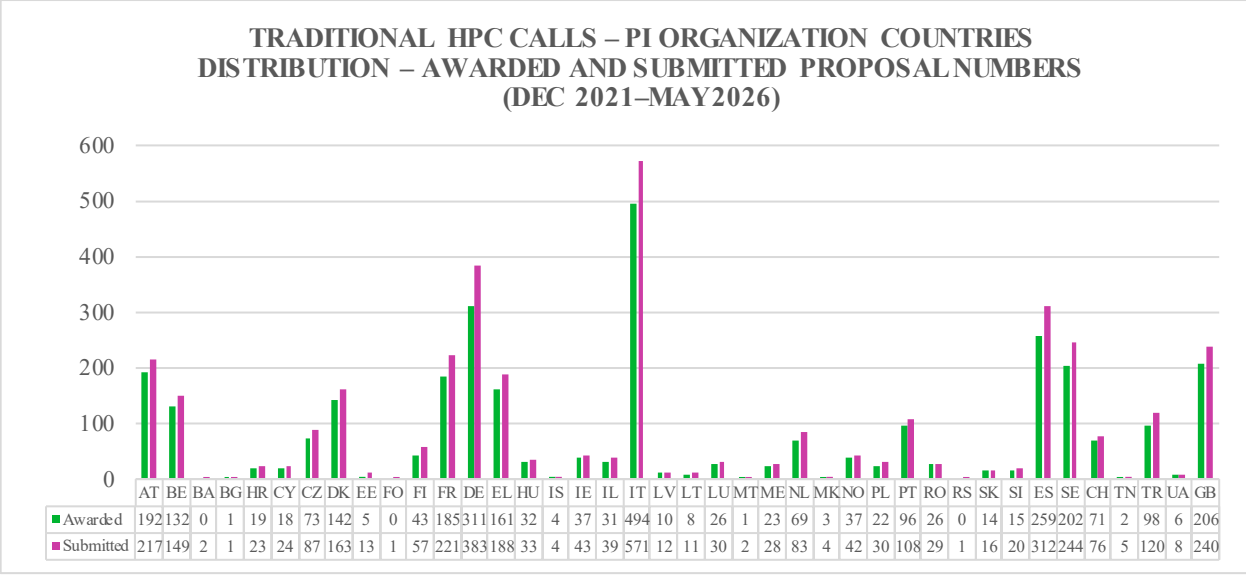


Figure 5: Research domains distribution of awarded projects throughout the different Traditional HPC Access Calls in the period December 2021 – May 2026

The Principal Investigator (PI) organization country distribution within the 3,639 submitted and 3,074 awarded projects of Traditional Access Calls shows that the awarded projects are led by PIs affiliated in Italy (16.06%), Germany (10.11%), Spain (8.36%), Sweden (6.57%), Austria (6.24%), and France (6.01%). Together, these six countries account for 53.35% of all awarded projects. The remaining 46.65% of awarded projects are associated with PI affiliations in Belgium, Croatia, Cyprus, Czechia, Denmark, Estonia, Finland, Greece, Hungary, Iceland, Ireland, Israel, Latvia, Lithuania, Luxembourg, Malta, Montenegro, Netherlands, North Macedonia, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Switzerland, Tunisia, Türkiye, Ukraine, and the United Kingdom.



	AT	BE	BA	BG	HR	CY	CZ	DK	EE	FO	FI	FR	DE	EL	HU	IS	IE	IL	IT	LV	LT	LU	MT	ME	NL	MK	NO	PL	PT	RO	RS	SK	SI	ES	SE	CH	TN	TR	UA	GB
Awarded	192	132	0	1	19	18	73	142	5	0	43	185	311	161	32	4	37	31	494	10	8	26	1	23	69	3	37	22	96	26	0	14	15	259	202	71	2	98	6	206
Submitted	217	149	2	1	23	24	87	163	13	1	57	221	383	188	33	4	43	39	571	12	11	30	2	28	83	4	42	30	108	29	1	16	20	312	244	76	5	120	8	240

Figure 6: PI organization country distribution – awarded and submitted proposals throughout Traditional HPC Access Calls in the period December 2021 – May 2026

When analyzing the PI gender of awarded projects, 84.6% were from male PIs, while 14.5% were from female PIs. The remaining 0.9% PIs preferred not to specify their gender.

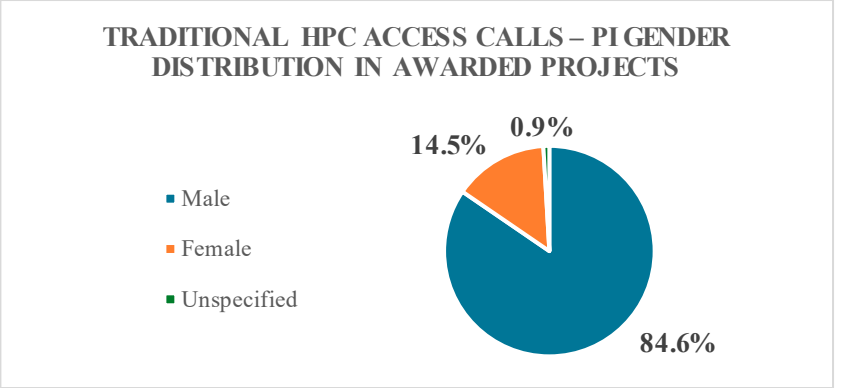


Figure 7: PI gender distribution – awarded proposals throughout Traditional HPC Access Calls in the period December 2021 – May 2026

3.1.1. The Extreme Scale Access call

This access call is intended for applications with high-impact, high-gain innovative research, 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 call. The call is open to all fields of science, industry and public sector and the 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 Scale Access call allocates 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.

During the period December 2022 to May 2026, a total of 253 proposals were submitted, of which 227 proposals were administratively accepted and 162 proposals were awarded (proposals submitted in the May 2026 cut-off are under evaluation – only submitted and administratively accepted proposal numbers are taken into consideration). The number of submitted proposals started from 40 proposals in December 2022, decreased to 19 proposals in May 2023, and then gradually increased to reach 60 proposals in May 2026, which is the highest value in the period. The number of administratively accepted proposals can be directly observed in a span of 3 years, taking the May 2023 cut-off with 17 administratively accepted proposals with the current, May 2026 cut-off with 53 administratively accepted proposals shows a natural increase over the years. The number of awarded proposals varied between different cut-offs with a gradual increase in awarded outputs in the later calls with an average success rate of 75% comparing the administratively accepted and awarded proposals. Across all Extreme Scale Access cut-offs, a total of 100,856,071 node hours were allocated to awarded projects.

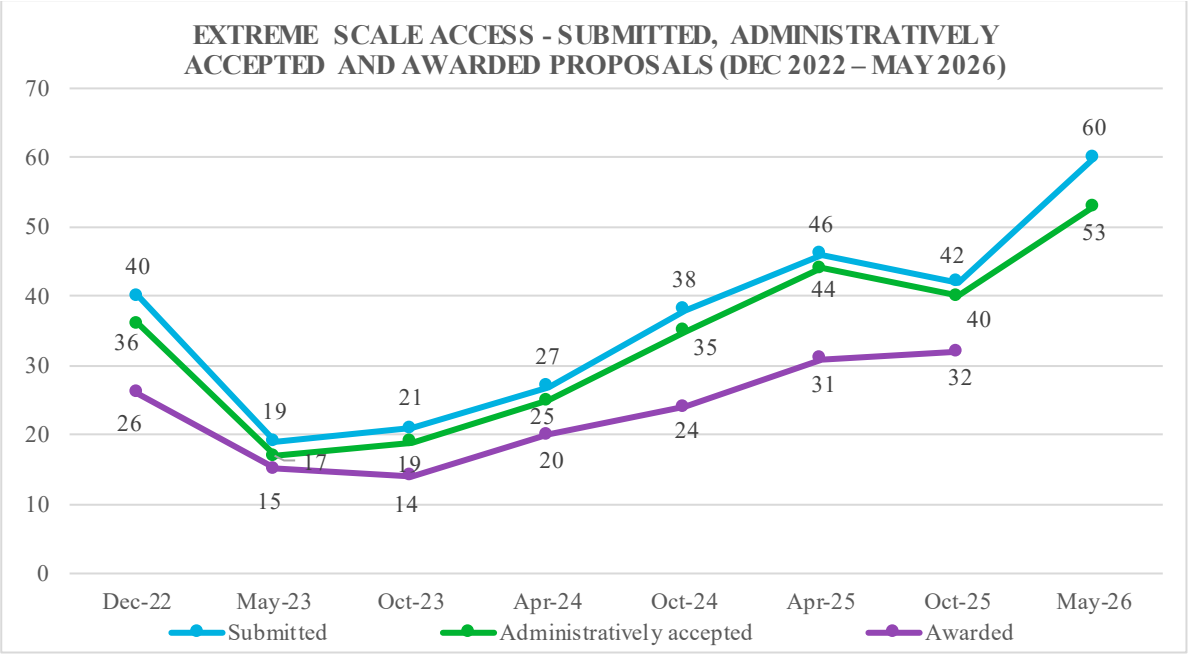


Figure 8: Extreme Scale Access - Submitted, administratively accepted and awarded proposals (Dec 2022 – May 2026)

The projects awarded within the Extreme Scale Access call belong to different research domains and the distribution of these domains can be observed in the chart below.

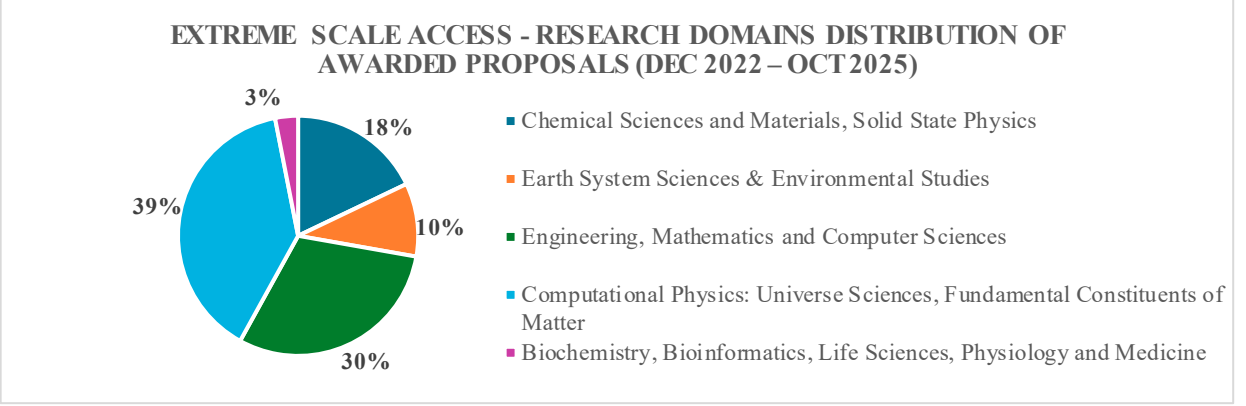


Figure 9: Extreme Scale Access – research domains distribution of awarded proposals (Dec 2022 – Oct 2025)

3.1.2. 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 allocates 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 to March 2026, within the Regular Access call, a total of 625 proposals were submitted, of which 576 proposals were administratively accepted and 419 proposals were awarded. The number of submitted proposals increased from 20 proposals in the December 2021 cut-off to 103 proposals in the March 2026 cut-off. Likewise, the number of administratively accepted proposals increased from 19 to 100, while the number of awarded proposals increased from 15 to 67 comparing the same cut-off periods. The highest number of submitted proposals (103) and administratively accepted proposals (100) was recorded in the March 2026 cut-off, whereas the highest number of awarded proposals (67) was achieved in both the March 2025 and March 2026 cut-offs. The continuous growth in the number of applications and awarded projects highlights the increasing demand for Regular Access as a stable and established access mode for production-oriented traditional HPC activities with an average success rate of 60%.

Across all Regular Access cut-offs, a total of 51,715,942 node hours were allocated to awarded projects. In the most recent March 2026 cut-off, a total of 7,591,549 node hours were awarded to successful proposals.

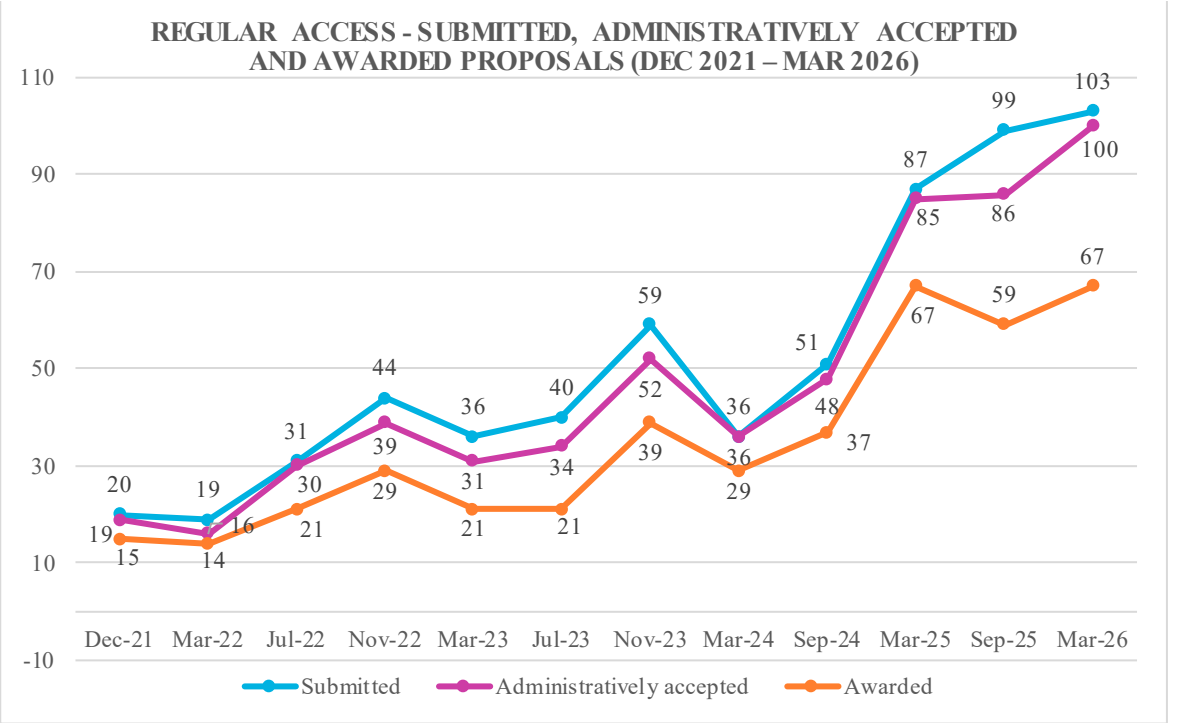


Figure 10: Regular Access – submitted, administratively accepted and awarded proposals (Dec 2021 – Mar 2026)

The projects awarded within the Regular Access call belong to different research domains and the distribution can be observed in the chart below.

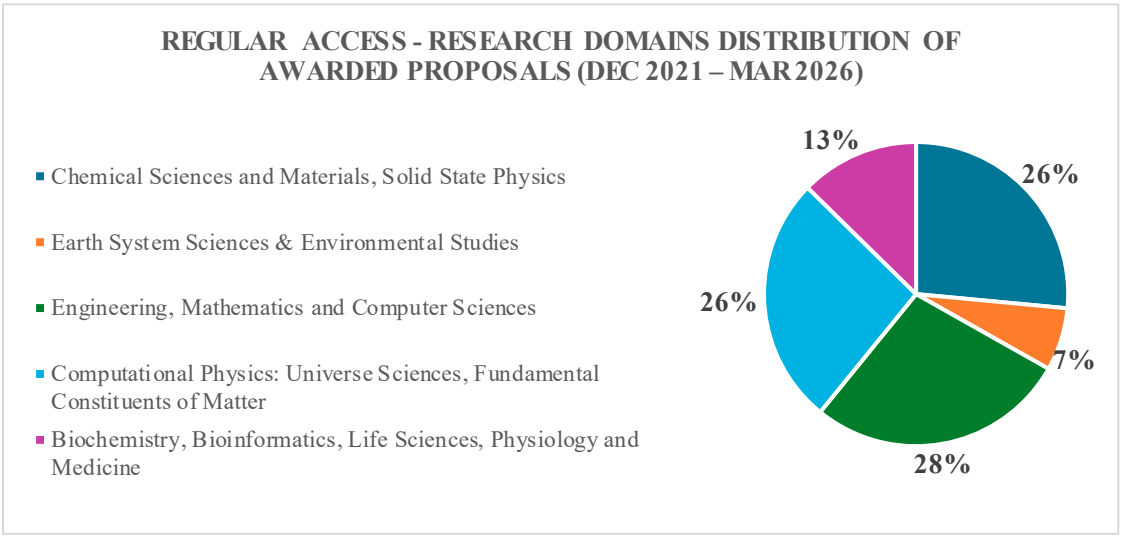


Figure 11: Regular Access – research domains distribution of awarded proposals (Dec 2021 – Mar 2026)

3.1.3. The Development Access call

The Development Access call 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 the Extreme Scale Access or Regular Access call. Access is provided through a continuously open call with monthly cut-offs (12 cut-off dates per year). 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.

From January 2022 to May 2026, within the Development Access call, a total of 1,814 proposals were submitted, of which 1,624 proposals were awarded, showcasing an overall success rate of 90%. The number of submitted proposals increased from 91 proposals in 2022 to 578 proposals in 2025. In 2026, we can observe a record submission number of 497 proposals . Similarly, the number of awarded proposals increased from 85 in 2022 to 529 in 2025, with 438 proposals awarded during the first five months of 2026. The annual success rate was 93% in 2022, 95% in 2023, 84% in 2024, 92% in 2025, and 88% for the period up to May 2026.

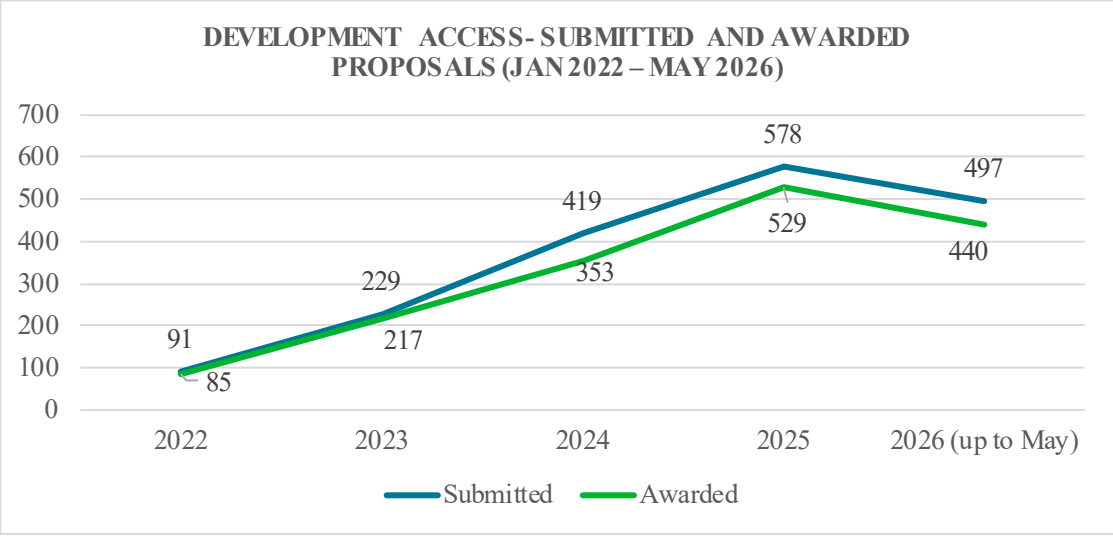


Figure 12: Development Access – distribution of submitted and awarded proposals (Jan 2022 – May 2026)

3.1.4. The Benchmark Access call

The Benchmark Access call 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 calls. 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 the Benchmark Access call from 2022 to May 2026, a total of 967 proposals were submitted, of which 869 proposals were awarded, resulting in an overall success rate of 90%. The number of submitted proposals increased from 108 in 2022 to 298 in 2025, before reaching 141 submissions by May 2026. Similarly, the number of awarded proposals increased from 102 in 2022 to 271 in 2025, with 124 proposals awarded during the first five months of

2026. The annual success rate remained consistently high throughout the period, ranging from 94% in 2022 to 88% in 2026, indicating a high proportion of submitted Benchmark Access proposals being awarded each year.

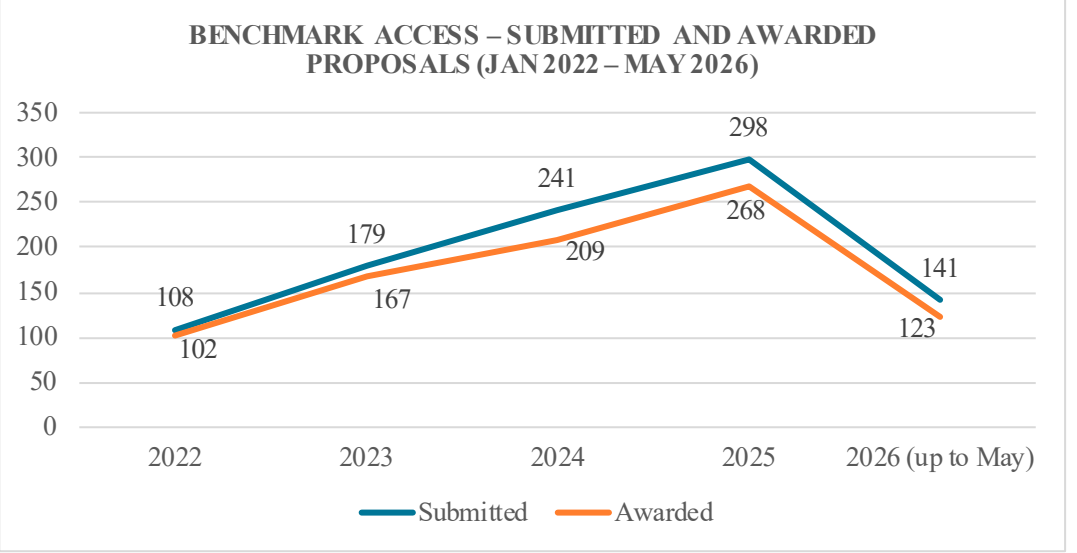


Figure 13: Benchmark Access – distribution of submitted and awarded proposals (Jan 2022 – May 2026)

3.2. Access Calls for AI Applications (AI Factory Access Calls)

During 2025, the EuroHPC JU went through 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 currently relies 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 Calls. The available partitions within these calls are the following: Vega GPU, MeluXina GPU, Discoverer GPU, LUMI-G, Leonardo Booster, MareNostrum 5 ACC, JUPITER Booster and Arrhenius GPU.

The AI Factories Access Calls, AI for Science and Collaborative EU Projects Access and AI for Industrial Innovation Access calls aim to support ethical AI, using a spectrum of different AI technologies such as machine learning and generative AI. These access calls allocate:

- AI for Science and Collaborative EU Projects Access: up to 25% of the overall EuroHPC share of access time;
- AI for Industrial Innovation Access: up to 30% of the overall EuroHPC share of access time.

Access call	Proposals awarded	Proposals awarded %	Node hours awarded	Node hours awarded %
AI and Data Intensive Applications Access (Apr 2024 – Apr 2025)	79	12.56%	3,266,600	16.1%
AI for Science and Collaborative EU Projects (June 2025-May 2026)	89	14.15%	5,524,329	27.2%
AIF Large Scale Access (Apr 2025 – May 2026)	55	8.74%	9,311,155	45.8%
AIF Fast Lane Access (Apr 2025 – May 2026)	229	36.41%	1,926,936	9.5%
AIF Playground Access (Apr 2025 – May 2026)	266	42.29%	286,875	1.4%
TOTAL	**718**	**100%**	**20,315,895**	**100%**

Table 3: AI Factory Access Calls – awarded projects and resources in the period April 2024 – May 2026

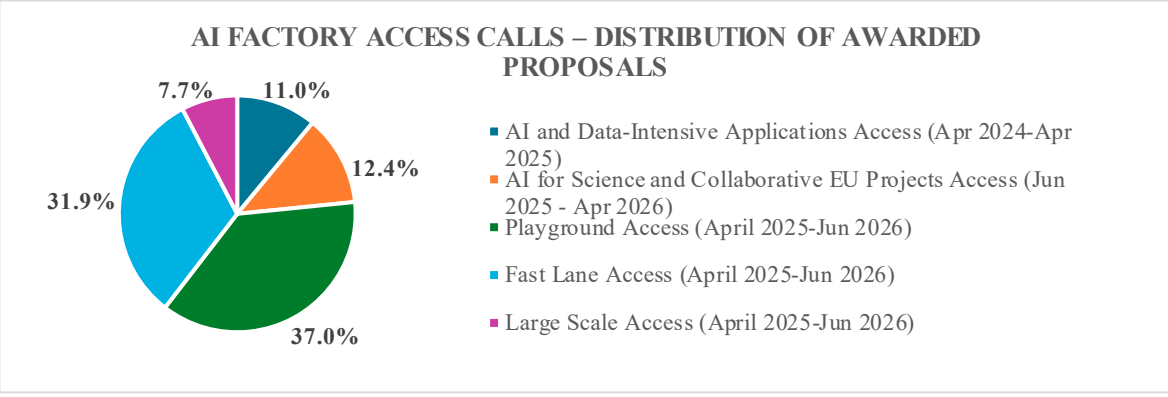


Figure 14: Share of awarded proposals across the different Traditional HPC Access Calls

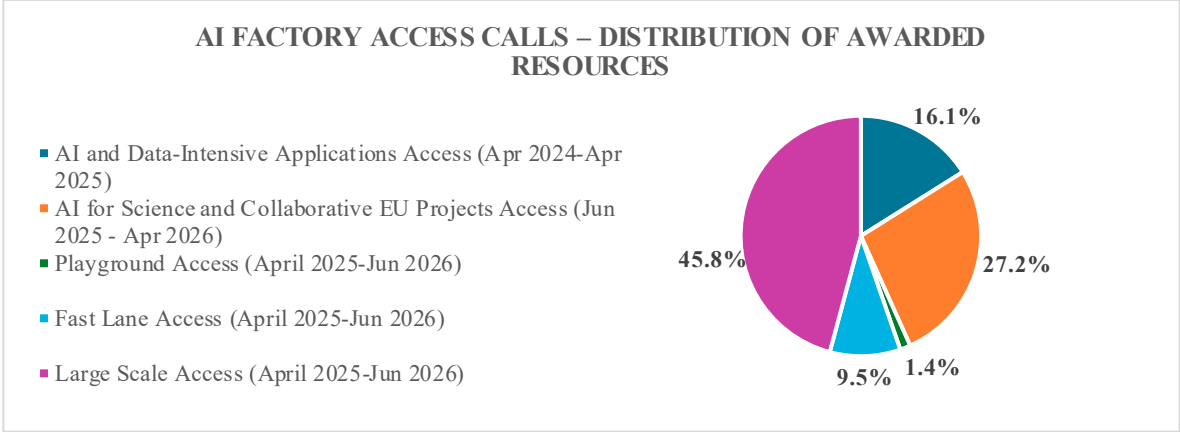


Figure 15: Share of awarded resources (percentage of node hours) across the different AI Factory Access Calls

Partition	AI and Data Intensive Applications Access	AI for Science and Collaborative EU Projects	AIF Large Scale Access	AIF Fast Lane Access	AIF Playground Access
Vega GPU	7,100	-	-	-	16,250
MeluXina GPU	100,000	230,899	-	95,113	8,750
Karolina GPU	7,500	-	-	-	-
Discoverer GPU	-	-	-	-	18,125
LUMI-G	280,000	1,039,036	2,285,000	167,250	27,500
Leonardo Booster	2,200,000	3,184,484	5,435,000	966,503	122,500
MareNostrum5 ACC	672,000	793,700	1,591,155	698,070	93,750
JUPITER Booster	-	193,710	-	-	-
Arrhenius GPU	-	82,500	-	-	-
TOTAL NODE HOURS	**3,266,600**	**5,524,329**	**9,311,155**	**1,926,936**	**286,875**

Table 4: AI Factory Access Calls – awarded resources in different partitions in the period April 2024 – May 2026

The Principal Investigator (PI) organization country distribution within the 1,042 submitted and 719 awarded projects of AI Factory Access calls shows that the awarded projects are led by PIs affiliated in Italy (14.60%), Spain (19.91%), Sweden (10.85%), Germany (8.34%), and France (6.54%) and Austria (5.56%). Together, these six countries account for 59.81% of all awarded projects. The remaining 40.19% of awarded projects are associated with PI organizations in Albania, Armenia, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Denmark, Estonia, Finland, Greece, Hungary, Iceland, Ireland, Israel, Latvia, Lithuania, Luxembourg, Moldova, Montenegro, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Switzerland, Türkiye, Ukraine, and the United Kingdom.

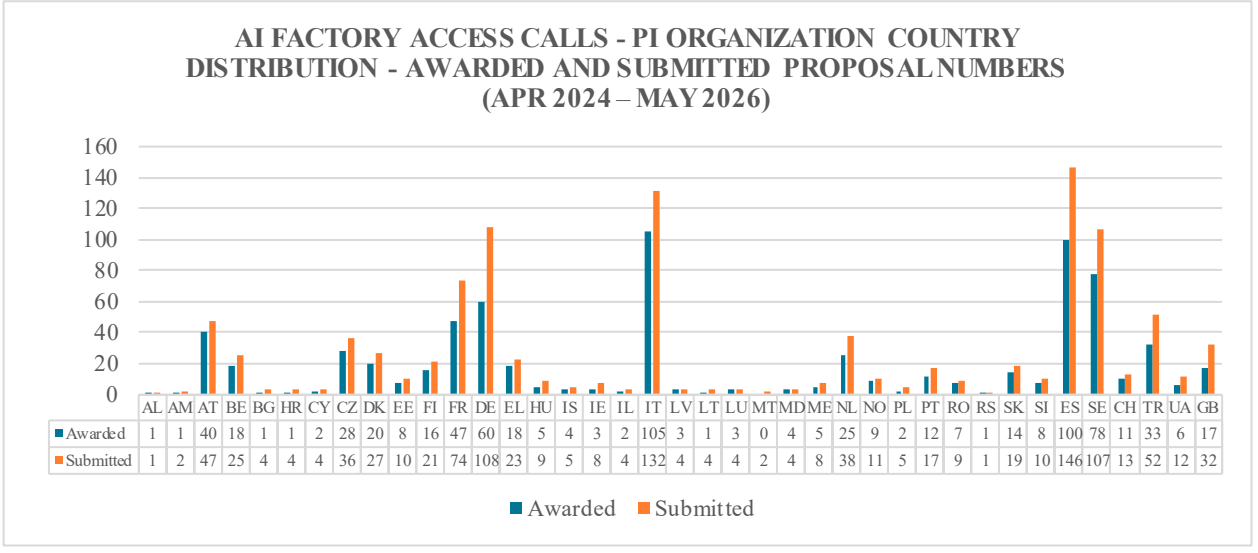


Figure 16: PI gender distribution – awarded proposals throughout AI Factory Access Calls in the period Apr 2024

When analyzing the PI gender of awarded projects, 88.8% were from male PIs, while 10.9% were from female PIs. The remaining 0.2% PIs preferred not to specify their gender.

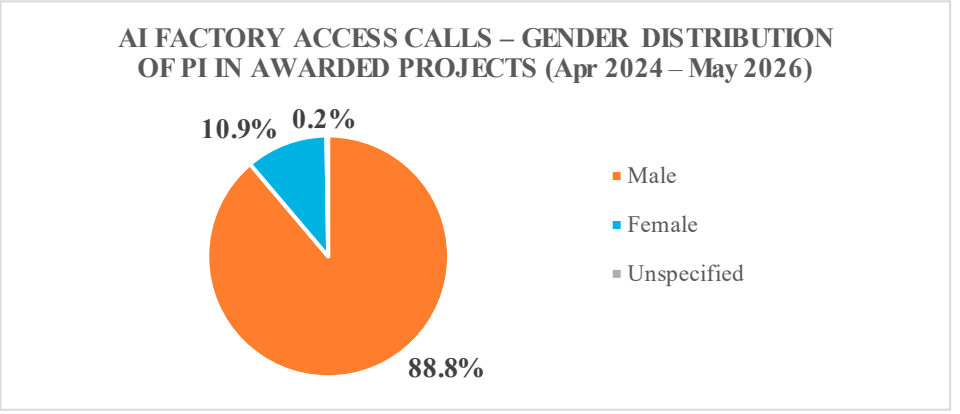


Figure 17: AI Factory Access Calls – PI gender distribution of awarded proposals in the period Apr 2024 – May 2026

3.2.1. The AI and Data-Intensive Applications Access call

The AI and Data-Intensive Applications Access call was introduced in 2024 in order to accommodate the growing number submissions and requests related to AI. The call was addressing requests coming from all sectors with an emphasis on industry, SMEs and startups.

This access call 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 steps: an administrative check, technical assessment, expert evaluation and consolidation of the results.

From the April 2024 to April 2025 cut-off periods, 122 proposals were submitted, of which 79 were awarded, resulting in the allocation of 3,266,600 node hours. Among the awarded projects, 38% of Principal Investigators (PIs) were affiliated with universities, followed by SMEs (25%), research institutes (14%), startups (11%), public entities (6%), and large enterprises (5%). In May 2025, the AI and Data-Intensive Applications Access call was succeeded by the AI for Science and Collaborative EU Projects Access call, concluding this access scheme under its original framework.

3.2.2. The AI for Science and Collaborative EU Projects Access call

Since June 2025, the AI for Science and Collaborative EU Projects Access call broadened the scope to support AI-driven scientific research, coming from academia, public sector organizations and industry. The call maintained the AI and Data-Intensive Applications Access call's accelerated evaluation process and continuous six-cut-off per year schedule. Since 2025, this access call has become part of the EuroHPC AI Factories initiative, where it constitutes the scientific access call alongside the AI Factory Access Calls for Industrial Innovation, supporting AI applications for science with a particular focus on machine learning, foundation models and generative AI.

Between June 2025 and April 2026, a total of 173 proposals were submitted under the AI for Science and Collaborative EU Projects Access call, of which 89 were awarded, corresponding to an overall success rate of 51%.

Across the six cut-off periods, the success rate ranged from 22% to 71%, with the highest success rates recorded in June 2025 and February 2026 (71%).

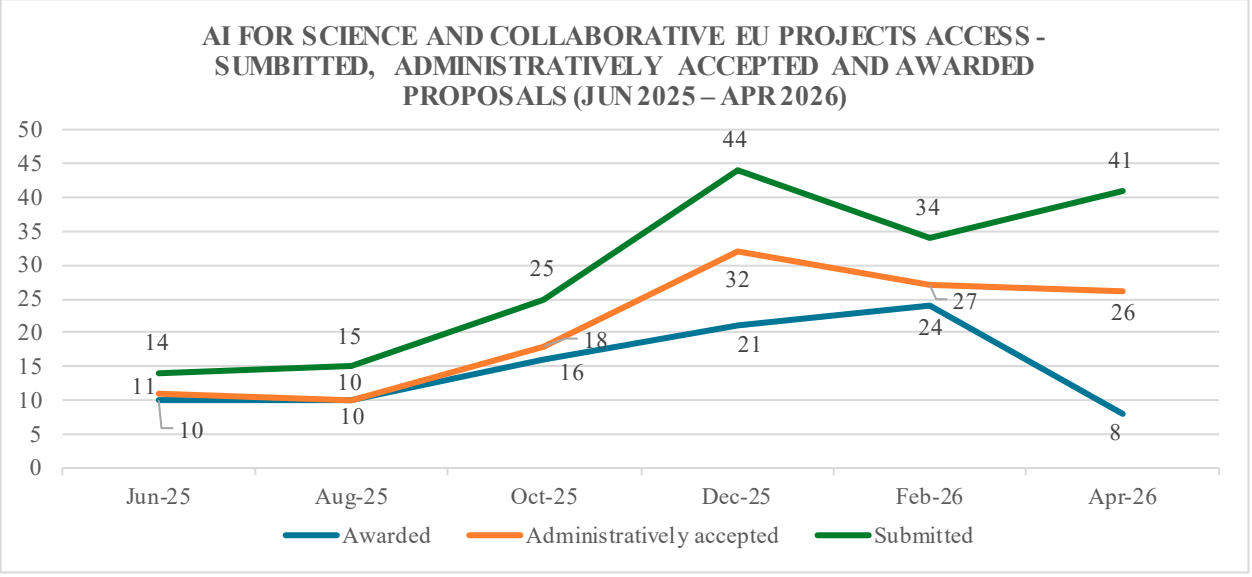


Figure 18: AI for Science and Collaborative EU Projects Access – submitted, administratively accepted and awarded proposals (Jun 2025 – Apr 2026)

The portfolio of awarded projects reflects the rapid adoption of modern AI methodologies across scientific disciplines. Deep learning represented the most frequently used AI technology, followed by computer vision (including technologies like image recognition, image generation, text recognition OCR, etc.), natural language processing, generative language models and machine learning, demonstrating the increasing demand for large-scale GPU-enabled infrastructure to support both AI models development and AI-driven scientific applications.

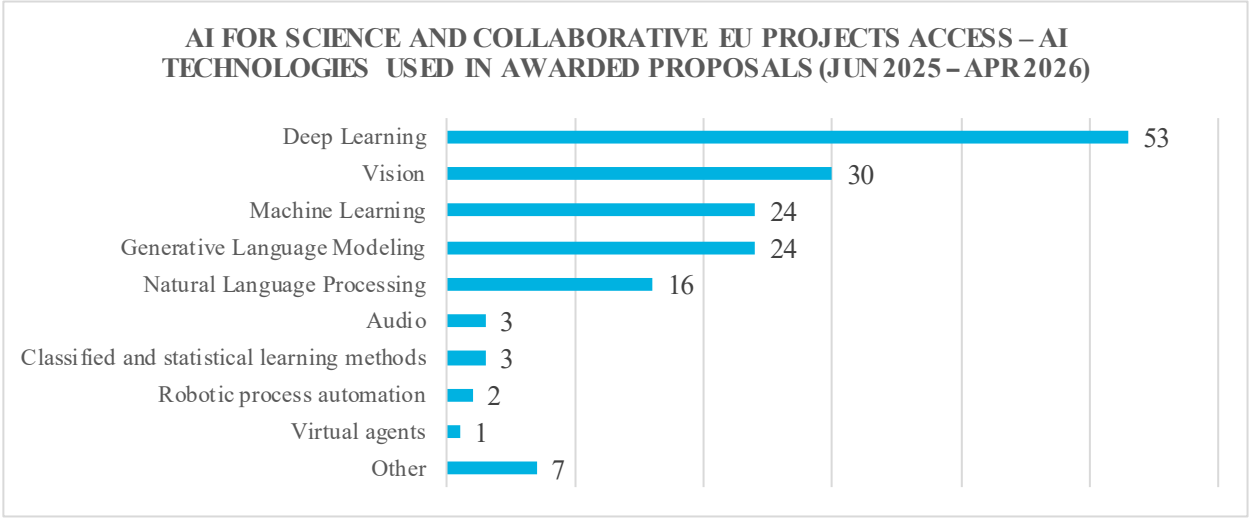


Figure 19: AI for Science and Collaborative EU Projects Access – distribution of used AI technologies in awarded proposals (Jun 2025 – Apr 2026)

3.2.3. The AI for Industrial Innovation Access calls

The AI Factories Access calls for Industrial Innovation are intended for users coming from industry, SMEs and startups that need HPC resources to perform AI related activities. These calls offer up to 30% of the EU share of the access time. The AI Factories Industrial Innovation Access calls include three access calls, 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 June 2026, a total of 746 proposals were submitted out of which 550 were awarded resources. Overall, 11,524,966 GPU hours have been allocated to successful projects (81% resources were allocated within the Large Scale Access call).

Access Call	Proposals submitted	Proposals awarded	Awarded %
Playground Access	319	266	83%
Fast Lane Access	315	229	73%
Large Scale Access	112	55	49%
TOTAL	**746**	**550**	100%

Table 5: AI for Industrial Innovation Access Calls – submitted and awarded proposals (April 2025 – May 2026)

The figure below presents the distribution of AI technologies employed across the AI Factory Access Calls between April 2025 and June 2026. Machine Learning was the most frequently reported technology, appearing in 188 proposals, followed by deep learning (148), generative language modeling (138), natural language processing (125) and vision applications (117). Other AI technologies included audio processing (37 proposals), virtual agents (32), decision management using classified and statistical learning methods (24), robotic process automation (15), and a small number of proposals (17) that have presented AI technologies outside of the provided selection. Across the three AI Factory access calls, the Playground Access mode accounted for the highest number of proposals for most AI technologies, followed by Fast Lane, while Large Scale Access was primarily used for computationally demanding workloads, particularly in generative language modeling, deep learning and machine learning applications.

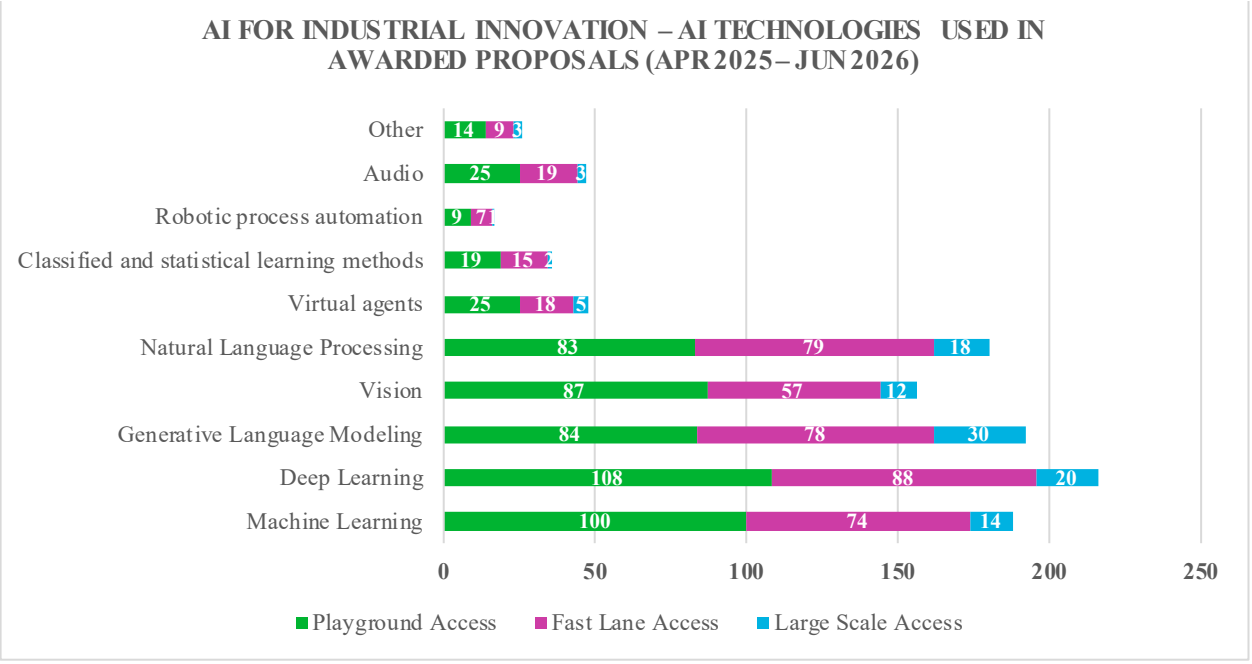


Figure 20: AI for Industrial Innovation Access Calls – distribution of AI technologies used in awarded proposals (Apr 2025 – Jun 2026)

3.3. Access Call for Quantum Computing – The Quantum Access Pilot Call

The Quantum Access Pilot call is intended for users requiring quantum resources for testing and development purposes, namely for collecting performance data or testing a method for documenting technical feasibility of their applications. Projects focusing on code and algorithm development, workflows development and quantum computing trainings are welcome to apply via this access call. The Quantum Access Pilot call provides specialized hybrid access (quantum accelerated HPC, or HPC-QC) and the users are encouraged to use hybrid workflows.

All submitted proposals undergo an eligibility check and a Technical Assessment performed by the Hosting Entity of the selected partition(s). The Technical Assessment is based on the project feasibility and the Hosting Entity may request the users to perform emulator benchmarks prior to accessing the physical hardware. The emulator access will be ensured by the Hosting Entity requesting the specific benchmarks. The call is continuously open with monthly cut-off dates (12 cut-off dates per year), with allocations granted for a period of three (3) or six (6) months.

Currently, the users can request access to Euro-Q-Exa, Lucy, PIAST-Q and VLQ systems and access to all EuroHPC quantum computers and simulators will be made available dynamically in the future as the systems become available. Resources requests are expected to be in QPU (Quantum Processing Unit) hours and the users can request a value between predefined resources ranges (either 1-10 or 10-25 QPU hours depending on the system). Considering that the call has been opened in late June 2026, the first projects are expected to be evaluated and granted access within the first cut-off period, with the submission deadline on 1 August 2026.

3.4. Strategic Access

The EuroHPC JU can allocate up to 10% of its resources share to strategic initiatives of public interest. The percentage limit is shared across all Strategic Access projects. In order to support projects of European strategic relevance, these projects do not need to undergo a peer-review evaluation.

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 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.

In 2025, the European Commission has identified a new strategic initiative, the OpenEuroLLM project that aimed at developing LLMs (Large Language Models) covering all EU languages and ensuring open and sustainable access to these models. The initiative represents a key step towards strengthening Europe's position in the global LLM landscape by fostering a sovereign, multilingual, competitive, and compliant European LLM. The OpenEuroLLM project has been granted access to four (4) EuroHPC JU systems, LUMI, Leonardo, JUPITER, and MareNostrum 5 for one (1) year starting in 2026.

During 2026, the European Commission has, together with the EuroHPC JU, launched a second Frontier AI Grand Challenge competition to support the development of sovereign, large-scale European AI models. The competition invited European actors to develop frontier AI models with a computational capacity equivalent to at least 400 billion parameters. The winning project, EUROPA, intends to develop a 500-billion-parameter AI model trained across all official EU languages. The EUROPA will be the first frontier model to deeply capture Europe's unique legal, administrative, and cultural knowledge. The project will receive access to EuroHPC supercomputers in 2026 for a period of one (1) year.

3.5. Experts involved in the peer-review process

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, almost 1,500 experts were involved in the review process till date, of which 101 have worked as Chairs, 778 as Rapporteurs and 616 as external referees (Scientific Reviewers). 50% of these experts were involved in the Extreme Scale Access call, 34% in the Regular Access call and 16% in the AI and Data-Intensive Applications Access and AI for Science and Collaborative EU Projects Access call.

3.6. Application Support Team (AST)

To facilitate the effective use of EuroHPC JU resources and support the successful implementation of awarded projects, dedicated application support services are provided through two complementary EuroHPC JU projects: EPICURE and MINERVA.

Proposals awarded through the Traditional HPC Access Calls can benefit from up to six months of support for application porting, optimisation, and scalability improvements through the EPICURE project. EPICURE provides a single point of contact through the European HPC Application Support Portal, allowing users from the public and

private sectors to retrieve information on EuroHPC JU systems, their architectures, access mechanisms, and available support services. Applicants may request assistance from the Application Support Team (AST) when submitting their proposal, while Hosting Entities may also recommend AST support during the evaluation process if additional technical assistance is considered beneficial.

For the AI Factories Access Calls, AST support is provided through the EuroHPC JU MINERVA project. The project aims to support successful applicants in the efficient development and deployment of AI applications on EuroHPC JU AI infrastructure by providing specialized technical assistance and expertise tailored to AI workflows. In addition, MINERVA serves as a European knowledge hub for large-scale AI technologies, offering guidance, training, and user support in collaboration with the AI Factories and the EuroHPC Academy.

The AST service was introduced in 2024 to support the access calls applications. Since its launch, a total of 489 proposals have received support, including 323 proposals under the Benchmark and Development Access calls, 73 under the Regular Access call, 37 under the Extreme Scale Access call, 37 under the AI and Data-Intensive Applications Access call, and 19 under the AI for Science and Collaborative EU Projects Access.

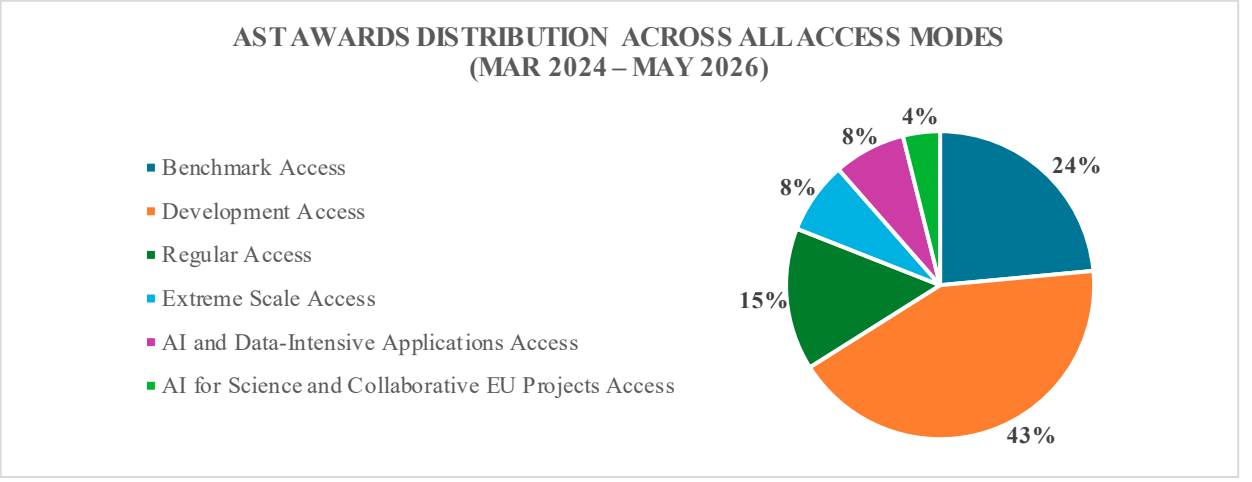


Figure 21: Application Support Team awards distribution across all access calls (Mar 2024 – May 2026)

4. Conclusion

The EuroHPC Joint Undertaking has established a comprehensive European access ecosystem that complements its world-class High Performance Computing, Artificial Intelligence and emerging Quantum Computing infrastructure. Beyond the deployment of petascale, mid-range, pre-exascale and exascale systems, the introduction of AI Factories and Quantum infrastructure has considerably expanded opportunities for users coming from academia, industry and the public sector to access computational resources through transparent and competitive access calls.

The analysis of proposals and resource allocations between December 2021 and May 2026 demonstrates a sustained increase in demand across all access calls and the growing reliance on the provided computational resources. While the established access calls continue to attract an increasing number of high-quality scientific proposals, the rapid uptake of AI Factory calls reflects the growing need for advanced GPU-based infrastructure supporting machine learning, foundation models and generative AI. At the same time, strengthened user support through the EPICURE and MINERVA Application Support Teams illustrates the continuous evolution of the EuroHPC access framework, ensuring that users can effectively exploit the offered computing infrastructure.

The core of the access ecosystem is a robust, transparent and independent peer-review process that ensures scientific excellence, technical feasibility and the fair allocation of Europe's most advanced computing resources. As demand continues to grow, the access and peer-review framework will become even more critical allocating impactful projects, fostering scientific excellence, accelerating industrial innovation and reinforcing Europe's technological sovereignty.

Acknowledgement

The authors would like to thank their colleagues from the EuroHPC JU Infrastructure Unit, especially Evangelos Floros, Head of Unit, for his continuous support and mentorship; Theo Brandmeyer and Dobromir Vasilev, Administrative and Programme Assistants, Peer-Review Sector; for their valuable contributions in the access calls management and their dedication; colleague from the Research and Innovation Unit, René Chatwell, Programme Manager, for his contribution and collaboration in the quantum access implementation.