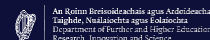




Dublin HPC and
User Days
22→24 September 2026

Dublin Ireland
22→24
September 2026



Coupling LAMMPS with LPC3D

Mesoscopic simulations of electrolytes in porous carbons:
a Python–MPI workflow for CPU and GPU supercomputers

El Hassane Lahrar, Mohamed Houssein Mohamed, Mathieu Salanne,
Guillaume Jeanmaret, Caspar van Leeuwen, Céline Merlet

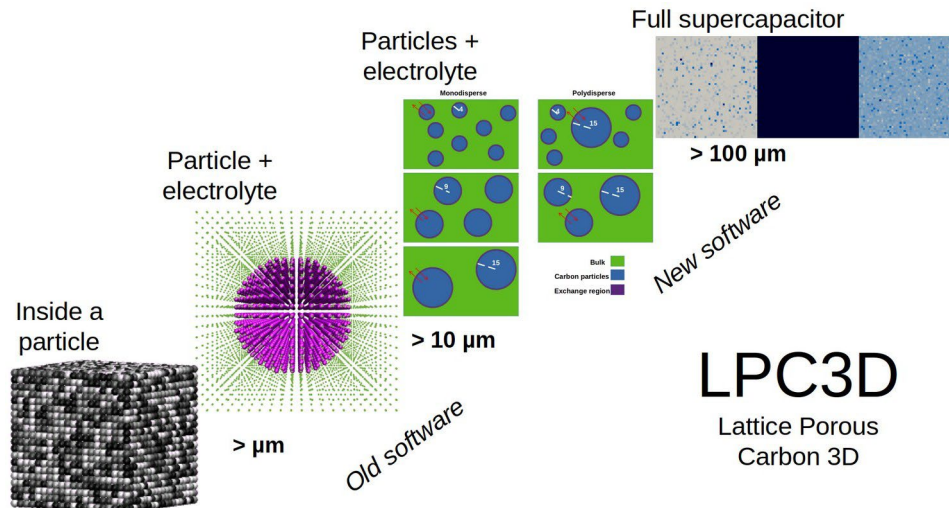
Sorbonne Université · CIRIMAT, Université de Toulouse · RS2E · SURF

EuroHPC User Days — Dublin, 22→24 September 2026

Supercapacitors: one scale is never enough

From the nanopore to the electrode, five orders of magnitude apart

- Supercapacitors store charge by voltage-driven electrosorption of ions in nanoporous carbon electrodes: high power density, very long cycle life
- Classical MD resolves the in-pore structure and desolvation, but reaches only a few nm and a handful of pores
- Real electrodes are 200–500 μm thick, with broad pore-size and particle-size distributions
- A mesoscopic level is needed to connect in-pore thermodynamics to device-scale transport



LPC3D

Lattice Porous
Carbon 3D

LPC3D spans a single particle to a full supercapacitor

LPC3D: a mesoscopic lattice-gas model

One lattice site = one pore, propagated by kinetic Monte Carlo

- Each accessible site carries a pore size, a free energy E_i and a resonant frequency ν_i
- Jumps between neighbouring sites follow a Metropolis acceptance rule, with site energies obtained by Boltzmann inversion of the densities
- Moment propagation replaces stochastic trajectories: cost scales as (number of sites $\times$ simulated time)
- Written in Python with PyStencils: symbolic update rules are compiled into optimised C++ kernels (OpenMP / MPI) and CUDA kernels
- Outputs: adsorbed quantities, diffusion coefficients, NMR spectra

10 k $\rightarrow$ 343 M

lattice sites: serial C code vs. new parallel version

1 $\rightarrow$ 1000 μm^3

single carbon particle vs. full supercapacitor

`pip install lpc3d`

open source — github.com/multixscale/LPC3D

The missing ingredient

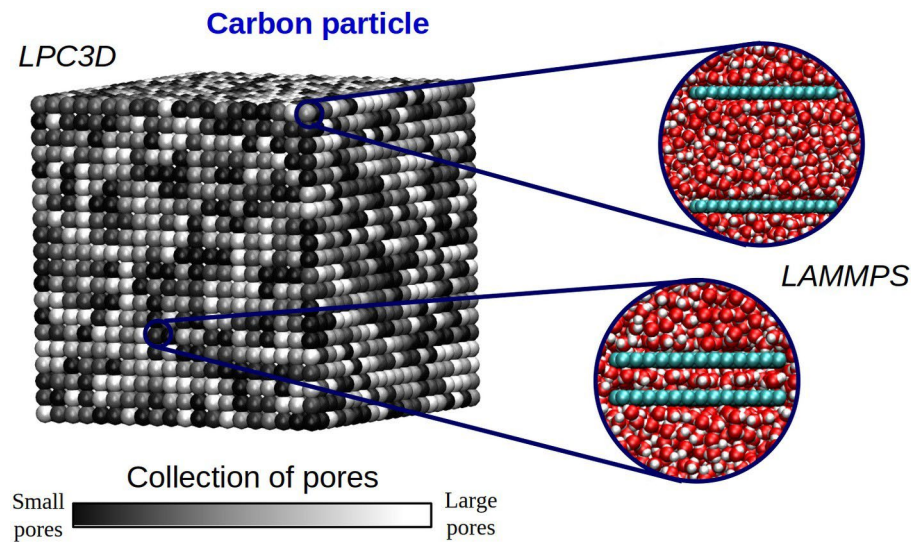
In-pore properties depend strongly on confinement — they should be computed for every pore size, not extrapolated from a single reference pore



LAMMPS × LPC3D: the idea

LPC3D drives an ensemble of independent MD simulations, one per pore size

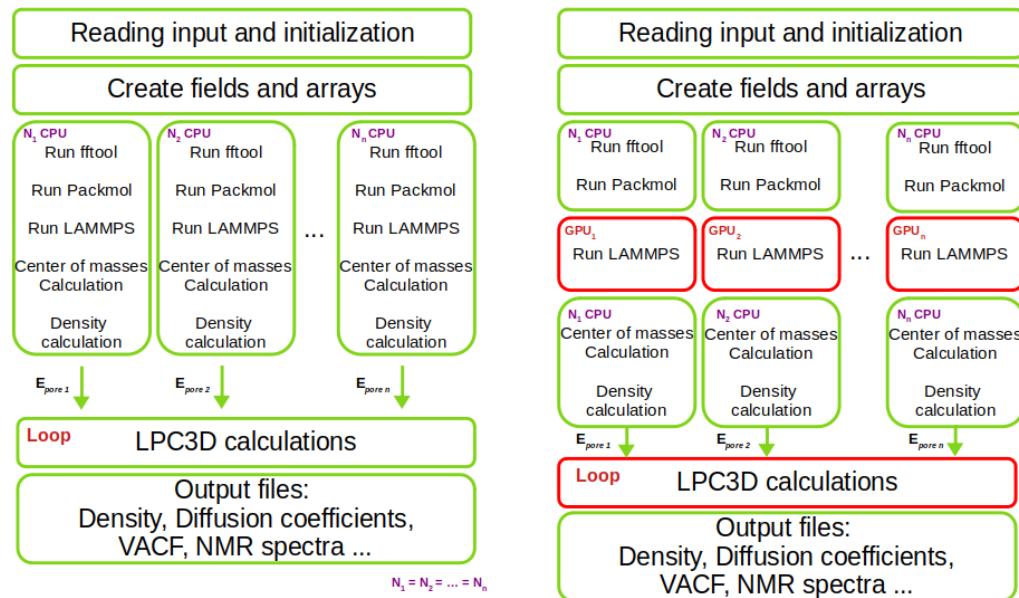
- One LAMMPS system per pore size, all executed concurrently inside a single MPI allocation
- Two quantities extracted from each trajectory: molecular centres of mass and equilibrium in-pore densities $n_{\alpha(d)}$
- α = cation, anion, solvent; d = pore size → a pore-resolved database instead of one reference value
- These densities parameterise the site energies E_i of the lattice model, so the mesoscale inherits atomistic accuracy



Each lattice site of LPC3D maps onto its own LAMMPS simulation

Workflow of the coupling

CPU-only (left) and GPU-accelerated (right) execution paths



1 fftool and Packmol (ref!!!!) build the atomistic system of each pore

2 LAMMPS: NPT equilibration, then NVT production

3 Post-processing: centres of mass and in-pore densities

4 E_{pore} transferred to LPC3D → densities, diffusion coefficients, NMR,...

One MPI allocation, many LAMMPS

mpi4py splits the communicator; the LAMMPS Python interface launches each instance

■ CPU configuration

IntelMPI · OpenMP for LPC3D

- The global MPI communicator is split into independent rank groups
- Each group runs one pore size from start to finish
- All pore sizes run concurrently when enough ranks are available

■ GPU configuration

IntelMPI + Kokkos · NVIDIA CUDA

- LAMMPS is compiled with the Kokkos package targeting NVIDIA devices
- Each simulation of a set is bound to one specific GPU
- Pores run in successive sets; $N(\text{pores}) \geq N(\text{GPUs})$ keeps every device busy

The Python driver keeps the workflow logic high level while LAMMPS runs natively in C++ — no external job orchestration is needed.

Deployment on HPC systems

Installed from PyPI, configured through standard environment variables

- Distributed as a Python package on PyPI, installed in one command
- Software stack provided by environment modules: Intel toolchain, CUDA, Python, CMake
- OMP_NUM_THREADS controls the OpenMP threading of the LPC3D part
- GPUs handled by the scheduler through CUDA_VISIBLE_DEVICES; LPC3D_NUM_GPUS = \$SLURM_GPUS selects the GPU execution mode

```
pip install lammmps-lpc3d
export OMP_NUM_THREADS=<N_THREADS>
export LPC3D_NUM_GPUS=$N_GPUS
srun python3 LAMMPS-LPC3D.py \
    -i lattice_gas.inpt
```

Benchmarked on three EuroHPC-related systems

 **Olympe** CALMIP — 36 cores / node

 **VEGA** 4 GPUs / node

 **MeluXina** 4 GPUs / node

Scaling results

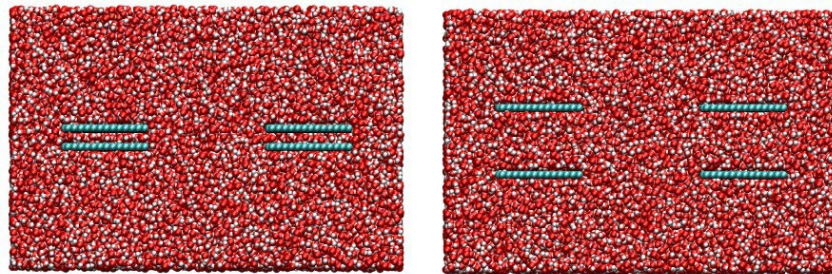
8 pore sizes · 50 000 water molecules per LAMMPS system · 125 000 lattice sites in LPC3D



Benchmark system

Liquid water in contact with slit pores of eight different sizes

- Two slit pores immersed in 50 000 SPC/E water molecules, for each pore size (two pores allow a potential difference to be applied)
- Hybrid Lennard-Jones / Coulomb, 12 Å real-space cutoff, PPPM at 10^{-4} , SHAKE constraints, 1 fs timestep
- NPT equilibration 10 ps at 298 K and 1 atm, then NVT production 100 ps
- LPC3D: $50 \times 50 \times 50 = 125\,000$ sites, 1 000 timesteps



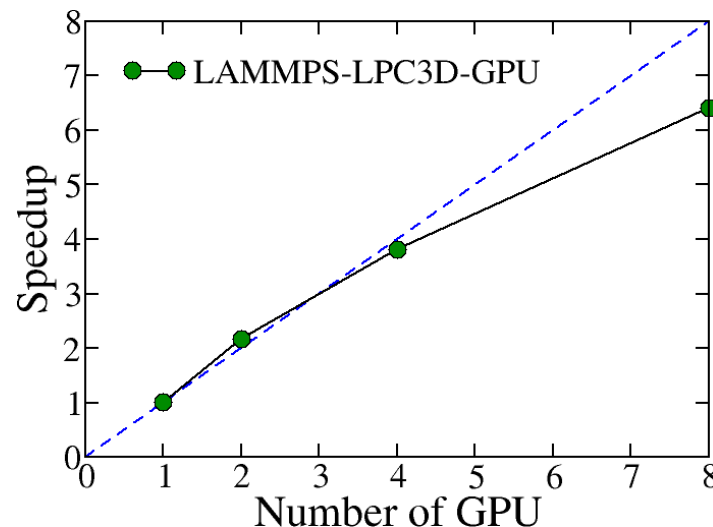
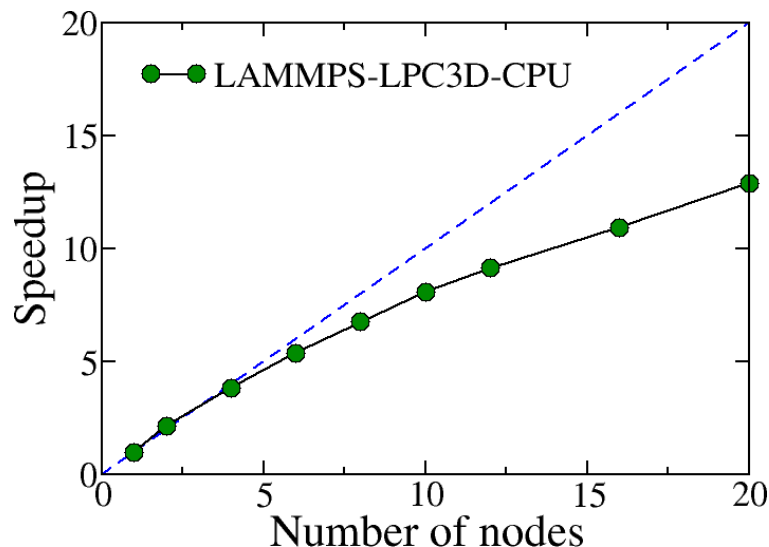
Slit pores of 0.76 nm (left) and 2.6 nm (right) immersed in water

8 pore sizes · 0.76 → 2.6 nm

chosen to match realistic carbide-derived carbons

Strong scaling

Fixed problem size of 8 pore sizes; speedup relative to a single node / a single GPU



Near-ideal up to 8 nodes (288 cores) and up to 4 GPUs, i.e. within one node. Beyond that, inter-node communication dominates.

Weak scaling

Problem size and allocated resources grow together — runtime should stay flat

CPU — one node per pore size

Pore sizes	Nodes	Time (h)
8	8	2.88
16	16	2.75
20	20	2.73

GPU — one GPU per pore size

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

Less than 6 % variation on CPU (2.73–2.88 h) and about 7 % on GPU (5.47–5.87 h).

Adding pore sizes costs almost nothing as long as resources scale with them — exactly what a heterogeneous electrode with a broad pore-size distribution requires.

Conclusions and outlook

A production-ready multi-scale workflow for porous carbon electrodes

■ What is done

- LPC3D drives concurrent LAMMPS runs inside a single MPI allocation, on CPU and on GPU
- Pore-resolved MD densities replace single-pore extrapolation
- Near-ideal strong scaling to 8 nodes / 4 GPUs; weak scaling within 6–7 %

■ What comes next

- Pore-size dependent dynamics, not only equilibrium densities
- Coupling LPC3D with molecular MDFT

```
pip install lammps-lpc3d
```

Thank you

Questions are welcome

el-hassane.lahrar@utoulouse.fr

celine.merlet@utoulouse.fr



This project has received funding from the European Union, the EuroHPC Joint Undertaking and the countries participating in the MultiXscale project under grant agreement No 101093169.

HPC resources: VEGA (EHPC-BEN-2025B11-078), CALMIP (P21014), MeluXina (p201090).



EuroHPC
User Days

Dublin Ireland
22→24
September 2026



EuroHPC
Joint Undertaking



ÉIRE2026.EU
IRELAND2026.EU

An FFT-oriented performance analysis and optimization of QUANTUM ESPRESSO on top-tier EuroHPC systems

Fabrizio Ferrari Ruffino, CNR-IOM



ISTITUTO OFFICINA
DEI MATERIALI



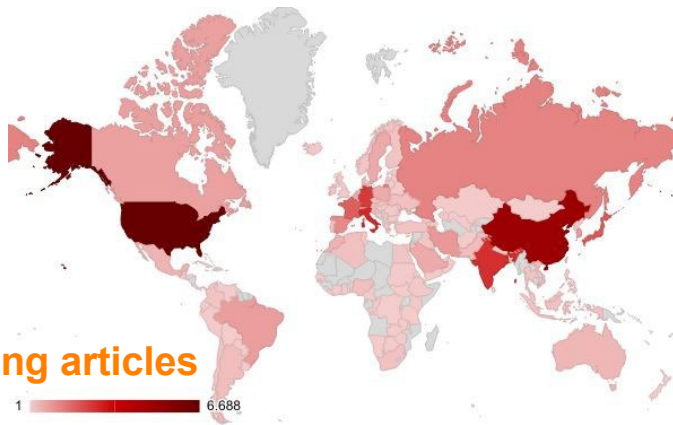
Outline

1. The QUANTUM ESPRESSO (QE) project;
2. Use cases;
3. The QE suite;
4. FFTs in QE;
5. Adaptive batching optimizations;
6. Results;
7. Summary & Outlook.



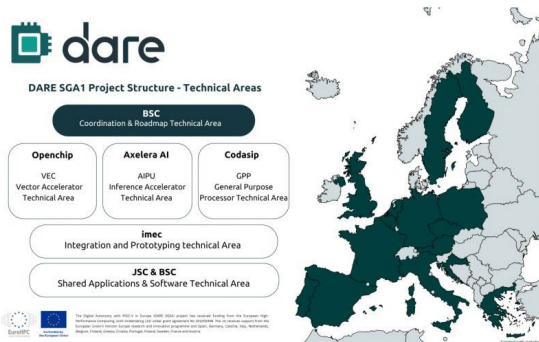
The QUANTUM ESPRESSO project

QUANTUM ESPRESSO (QE) is an integrated suite of **Open-Source** computer codes for electronic-structure calculations and materials modeling at the nanoscale.



28k+ citing articles

3 large projects:

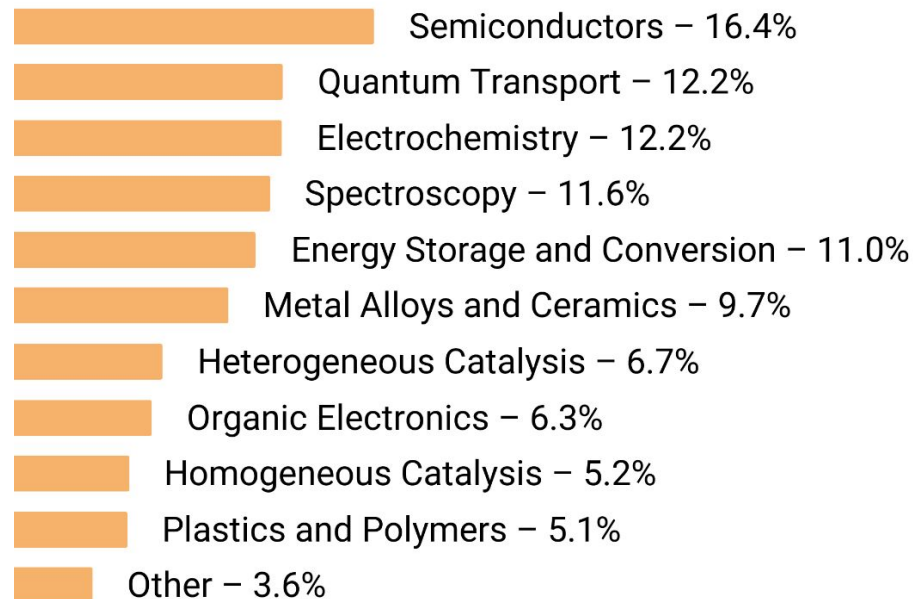
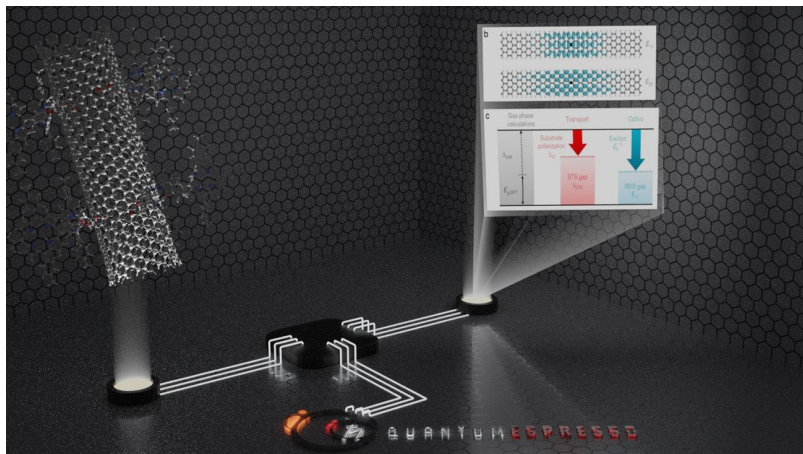


ICSC
Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

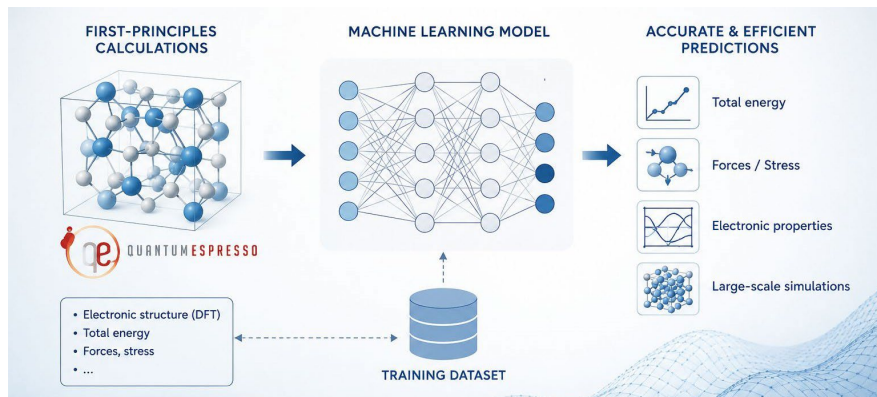


Use cases of QE I

QUANTUM ESPRESSO is routinely used to simulate **large and complex atomistic systems**. Clusters hosted at HPC centers play a crucial role to enhance accuracy and use predictive methods.



Use cases of QE II



QE is used as scientific application to drive and benchmark multiple **hardware/software co-design** projects (DARE, MaX,...).

⇒ For example the EuroHPC project **DARE** has selected QE as strategic application for porting and benchmarking new **RISC V processors and vector accelerators**, currently under development.

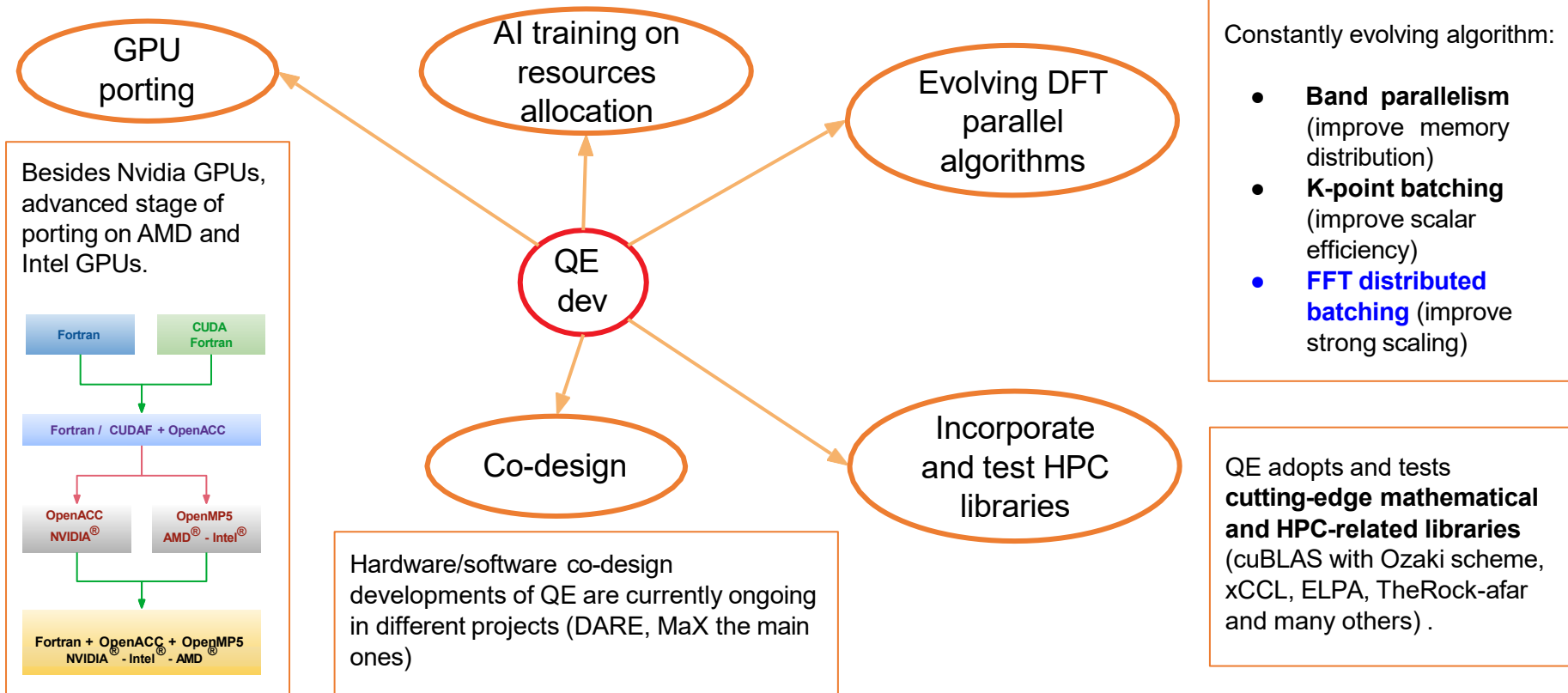
A lot of activity in **atomistic simulations to create machine learning models** for dynamics, spectroscopy, magnetism...

⇒ QE can generate **accurate state-of-the-art** data to **train the models**.

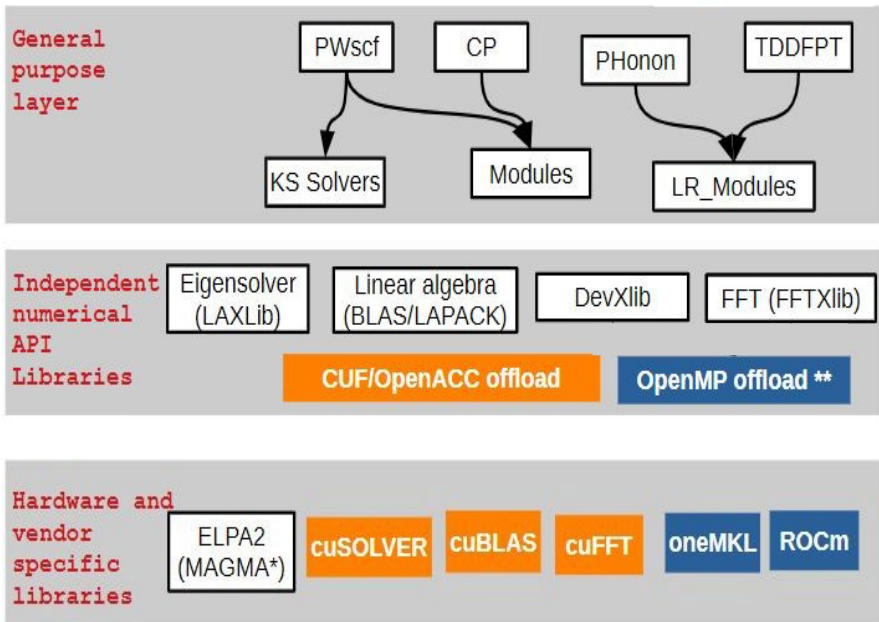
Main requirements: more features; reliability and performance; easy to plug into training workflows (data extraction, kernel interfaces...);



Ongoing HPC developments

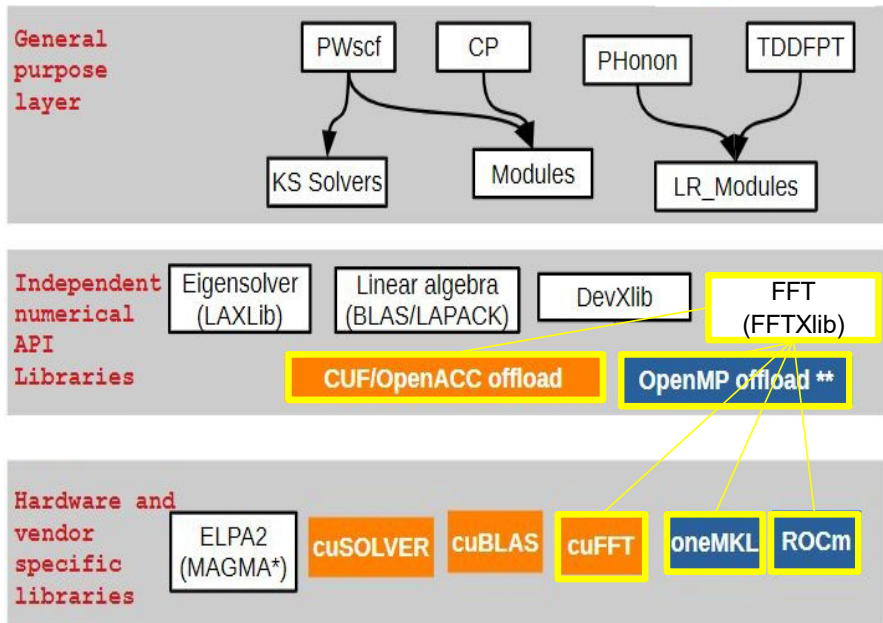


The QE suite

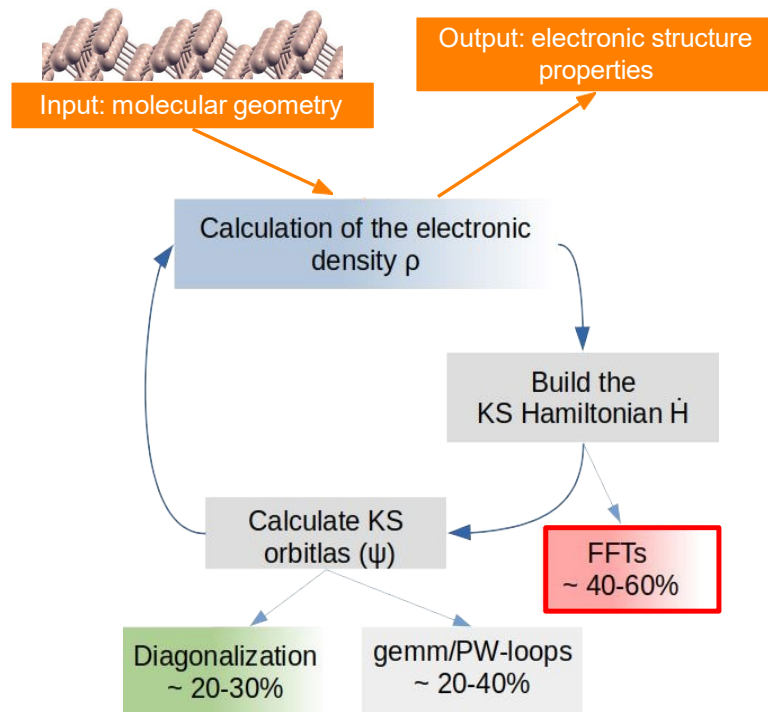


- **PWscf:** main code. SCF calculations;
- **PHonon:** vibrational frequencies, dielectric response, anharmonic terms and many more with linear response;
- **HP** - Computes Hubbard parameters with linear response;
- **TDDFPT:** Optical spectra and collective excitations;
- **QEHeat:** energy flux;
- **PP:** post-processing utilities;
- **EPW:** electron phonon with Wannier Function;
- **KCW:** Koopmans-Compliant band structure calculations with Wannier Functions;
- Interoperability with many other external packages ...

The QE suite and PWscf



PWscf flow



Main parallel levels (PWscf)

Distributing the **Kohn-Sham** equations for materials:

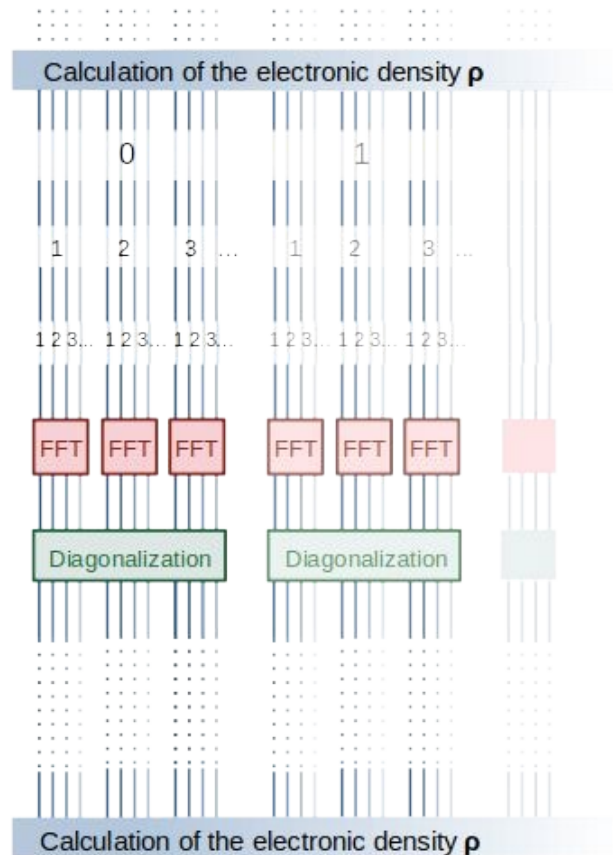
$$\hat{H}^{KS} \psi_{ik}(\mathbf{r}) = \varepsilon_{ik} \psi_{ik}(\mathbf{r})$$

$$\psi_{ik}(\mathbf{r}) = \sum_{\mathbf{G}}^{N_{PW}} C_{\mathbf{G},ik} \frac{e^{i((\mathbf{G}+\mathbf{k}) \cdot \mathbf{r})}}{\sqrt{\Omega}}$$

Pools:

Bands:

R&G:



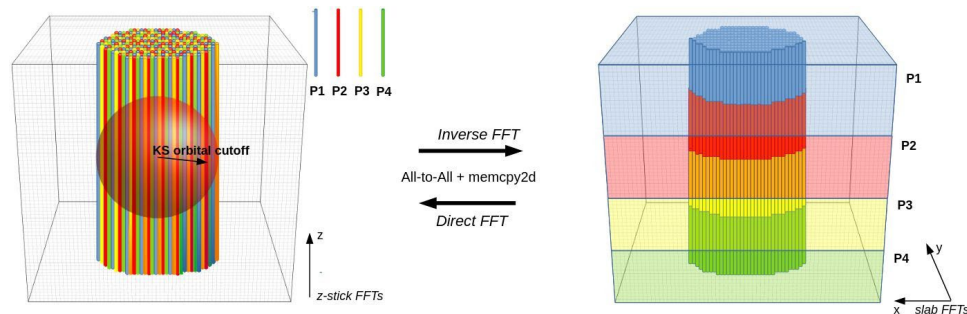
Average backend problems size

- Memory bound $\propto$ [grid_size x n_bands];
- **FFT grid size** in QE: $< 1.000^3$ [largest so far in HPC: $\sim 30.000^3$ *];
- **Projected eigenproblem size** in QE (n_bands): < 15.000 [largest so far in HPC: $\sim 2.000.000$ **];

→ **many medium-small FFTs** and a **few small diagonalizations** (fitting a single GPU).

FFTXlib: FFTs for materials simulations

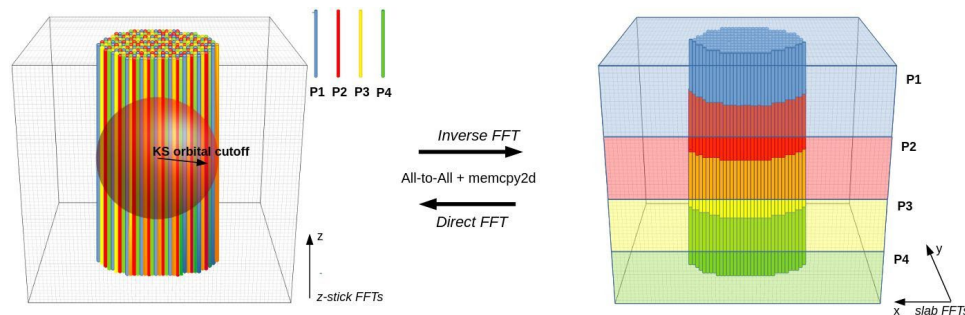
Spherical cutoff in reciprocal space:



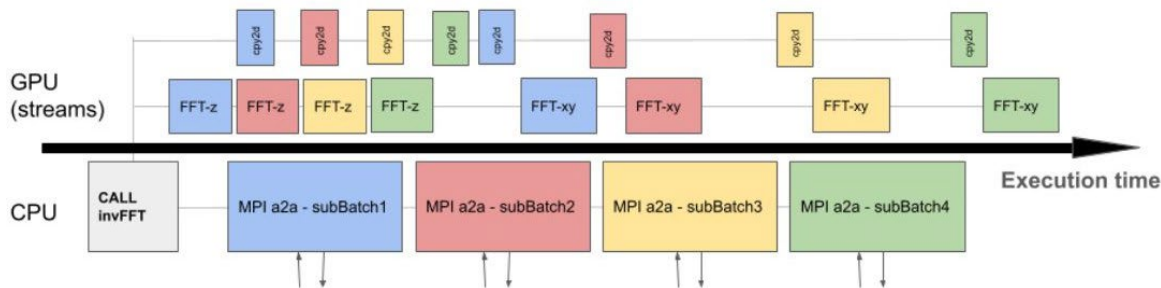
- To prevent aliasing electronic orbitals have smaller cutoff:
→ exclude large portion of reciprocal space (less computation and smaller communication).

FFTXlib: FFTs for materials simulations

Spherical cutoff in reciprocal space:



Many functions (**orbitals**) to process **concurrently**



- To prevent aliasing electronic orbitals have smaller cutoff:

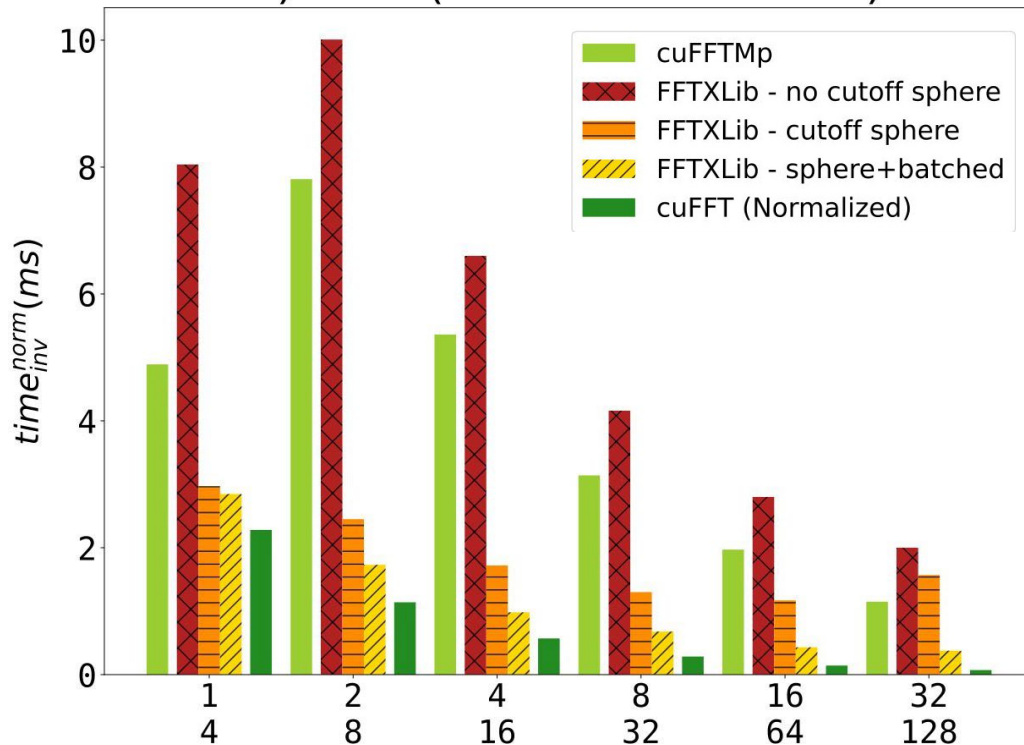
→ exclude large portion of reciprocal space (less computation and smaller communication).

- To process many distributed FFT grids (relatively small):

→ orbital batching to exploit communication/computation asynchronicity and optimize resources usage.

FFTXlib: current performance I

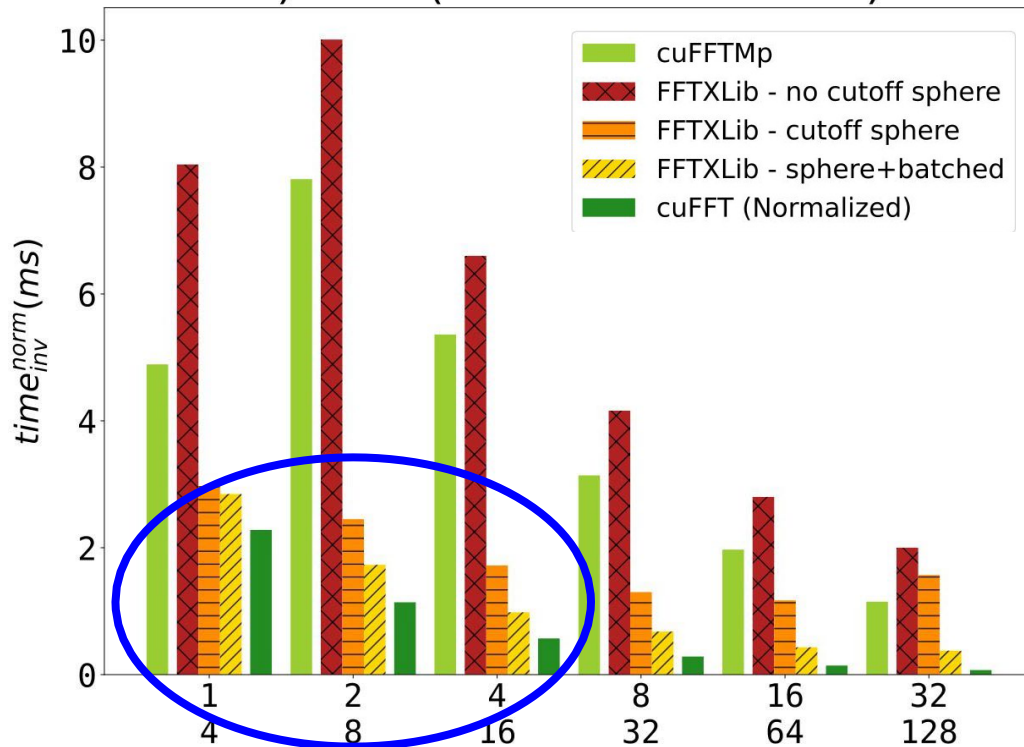
e) 512^3 (MareNostrum5 ACC)



- Setting up a **performance assessment** of FFTXlib on GPU
- Grid sizes on the boundary between GPU- scalar and distributed FFTs;
- Comparison between **cuFFTMp** and **cuFFT**

FFTXlib: current performance II

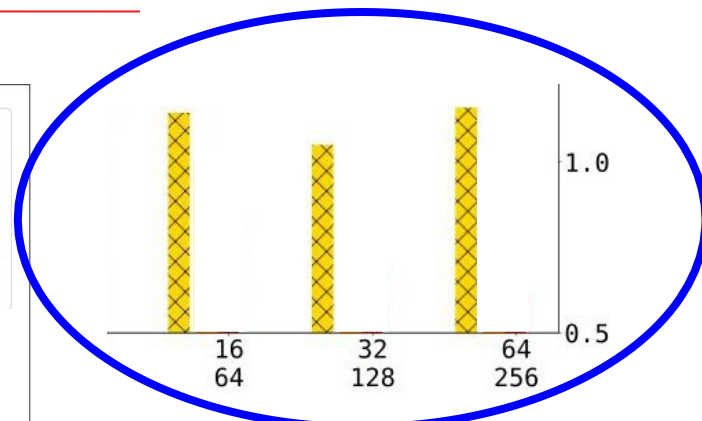
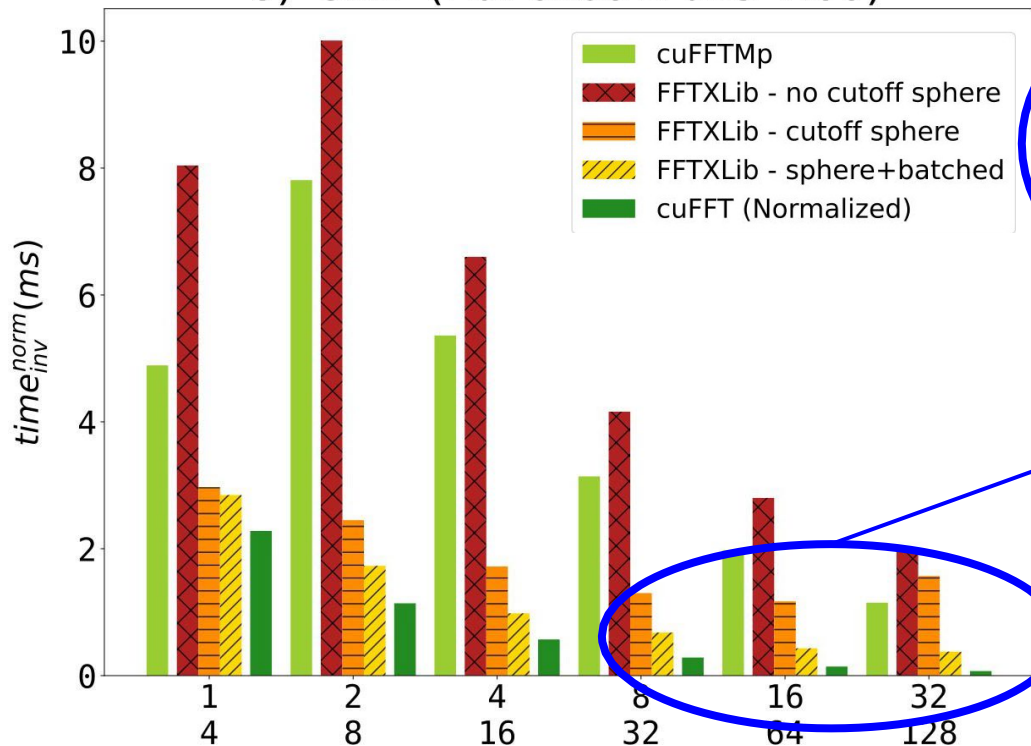
e) 512^3 (MareNostrum5 ACC)



- On modern GPU systems $T_{\text{comp}} > T_{\text{comm}}$ or can be made almost so on few nodes with batching;
- PW distributed FFTXlib $\approx$ band distributed cuFFT
- Constraint: GPU memory

FFTXlib: current performance III

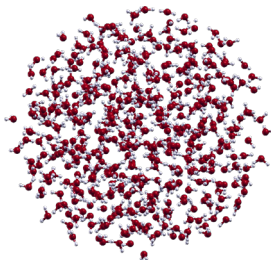
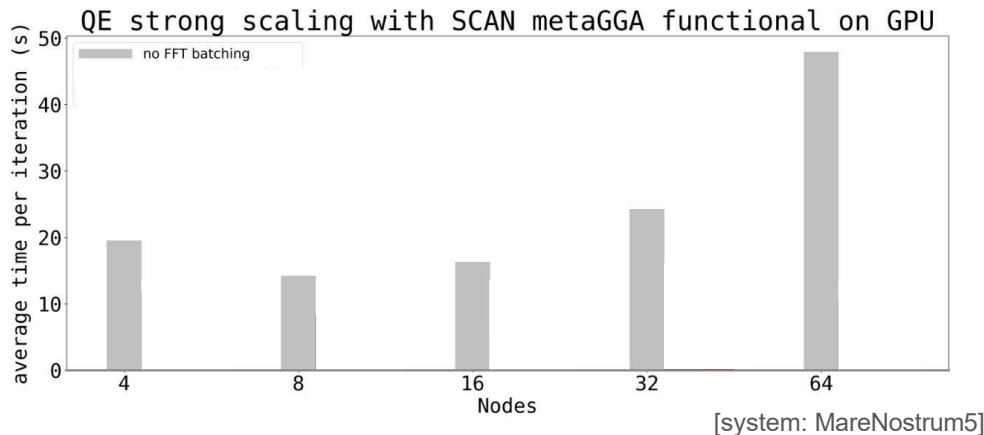
e) 512^3 (MareNostrum5 ACC)



- Buffer size becomes too small after a threshold (latency regime);
- standard batching pushes the threshold further, but the problem remains;

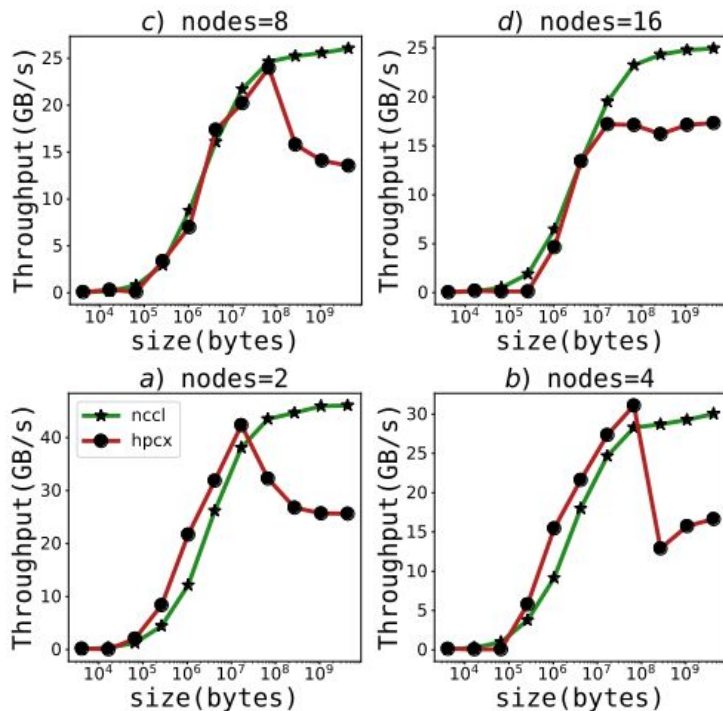
Strong scaling degradation in QE

PWscf - Water (432 molecules, MGGA SCAN XC)



- Buffer size becomes too small after a threshold (latency regime);
- standard batching pushes the threshold further, but the problem remains;
- Having a large number of bands, allows to make the batching adjustable at will (almost).

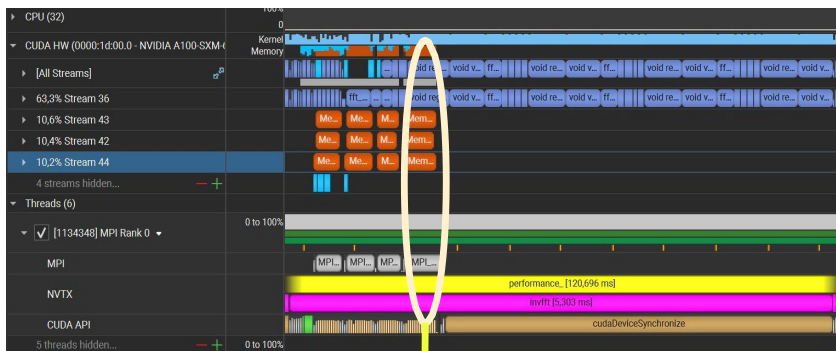
Communication backends



- Communication backends: high variability of throughput;
- xCCL: some advantages for larger buffers;
- The larger the better not always true (see graphs).

→ Tune batching parameters to get the **minimum buffer size** corresponding to the **maximum throughput**.

Adaptive FFT batching I



Minimal overlap function

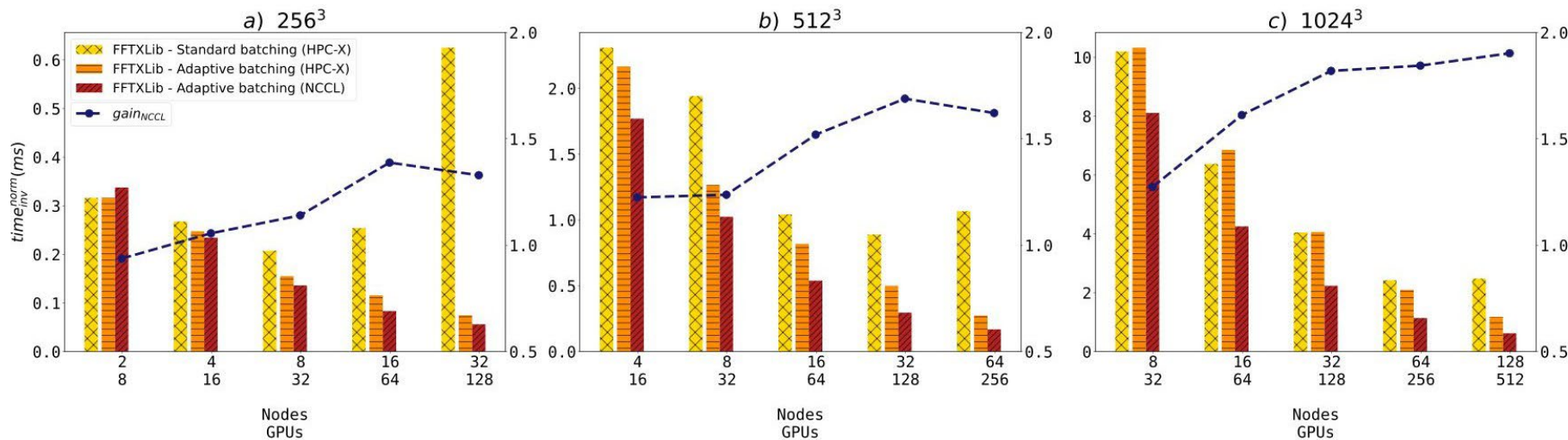
- Rather large variability between backend libraries (both computation and communication);

Aim: given a chosen hardware, software stack and input system, **find automatically the batching parameters** that gives best performance.

- Select a function to estimate the overlap in time between computation and communication;
- Optimal sampling to find the best choice in the shortest time.

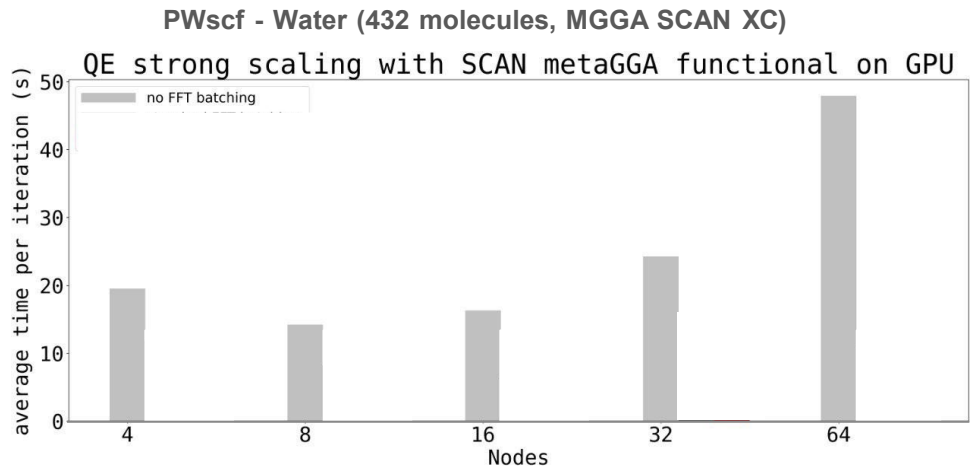
→ **Lightweight autotuning** with **Thompson sampling** over function calls. For FFTXlib: sample overlap between computation and communication for a set of batch sizes

Adaptive FFT batching II

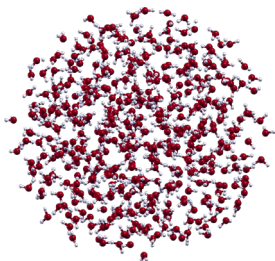


- Adaptive batching prevents large FFTs communications to fall into latency regime;
- NCCL: some gain for larger buffers;

Results



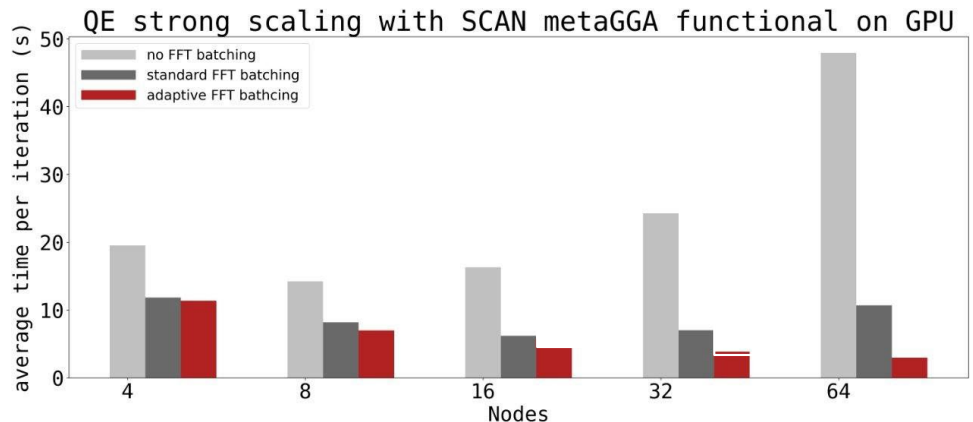
[system: MareNostrum5]



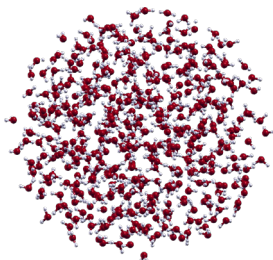
- FFT standard batching extended to metaGGA (and other parts of QE);
- Standard batching significantly improves performance;
- Parallel efficiency threshold (0.7) shifted from 8 nodes to 16 with adaptive batching.

Results

PWscf - Water (432 molecules, MGGA SCAN XC)



[system: MareNostrum5]



- FFT standard batching extended to metaGGA (and other parts of QE);
- Standard batching significantly improves performance;
- Parallel efficiency threshold (0.7) shifted from 8 nodes to 16 with adaptive batching.

Summary & Outlook

Conclusions:

- **FFTXlib** still provides significant performance advantage with PW distribution;
- we observed an increasingly **weaker dependence over different parallel schemes** at small node counts on modern machines;
- **adaptive batching** allows to maximize communication throughput and GPU occupation regardless job size.

What's next:

- ❑ with the always increasing size of GPUs and single nodes, **batching** is becoming **increasingly more relevant**, at multiple levels, not only FFTs;
- ❑ explore **autotuning extensions** beyond FFTs;
- ❑ **optimize memory** distribution with band parallelism.

Acknowledgements

QE DEVELOPERS

- [Pietro Delugas, SISSA](#)
- [Ivan Carnimeo, CNR-IOM](#)
- Oscar Baseggio, SISSA
- [Francesco Andreucci, SISSA](#)
- Stefano Baroni, SISSA, CNR-IOM
- Paolo Giannozzi, UniUD, CNR-IOM
- [Fabrizio Ferrari Ruffino, CNR-IOM](#)

QE FOUNDATION

- Francesca Garofalo

CINECA

- [Laura Bellentani](#)
- [Angela Acocella](#)
- [Sergio Orlandini](#)
- [Fabio Affinito](#)

ARGONNE NL

- Ye Luo

NVIDIA®

- Piero Altoè
- Louis Stuber

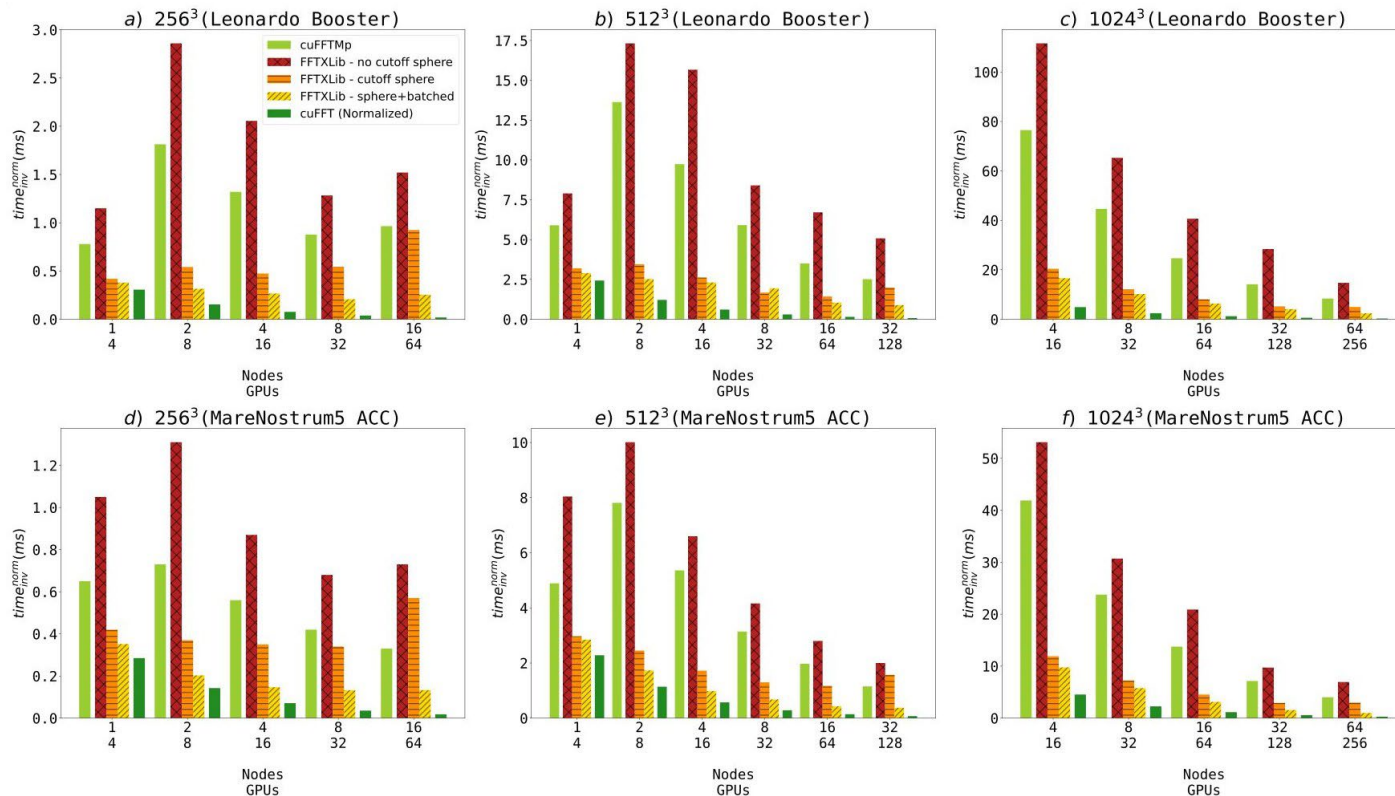
AMD®

- [Giacomo Rossi](#)
- Ossian O'Reilly
- Leopold Grinberg

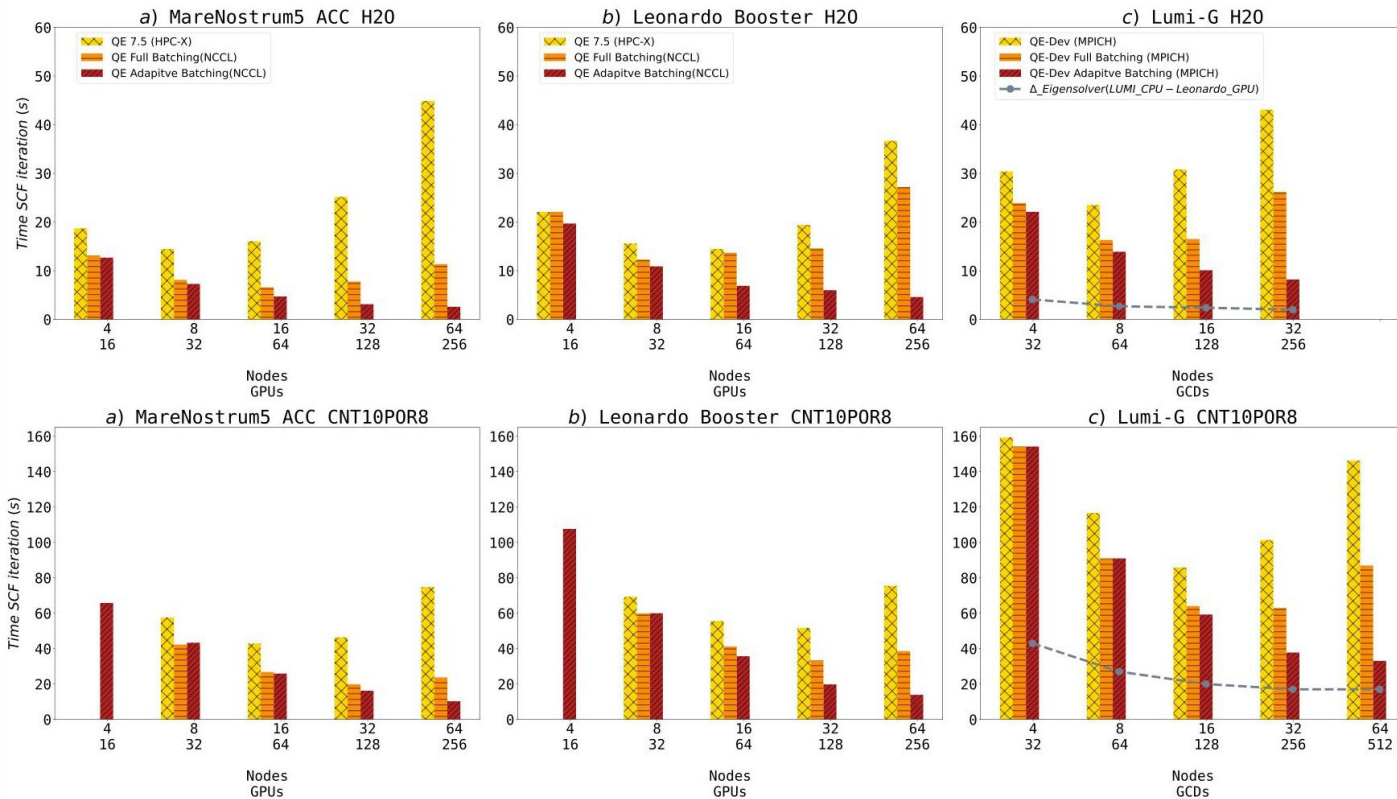
HPE®

- Pierre Carrier
- Alfio Lazzaro

FFTXlib benchmarks



More QE results



Scaling and performance aspects of the P3M method on multiple GPUs

Rodrigo A. C. Bartolomeu, René Halver and Godehard Sutmann

Simulation and Data Laboratory Complex Particle Systems

Jülich Supercomputing Centre (JSC) - Forschungszentrum Jülich

HPC Architecture Landscape

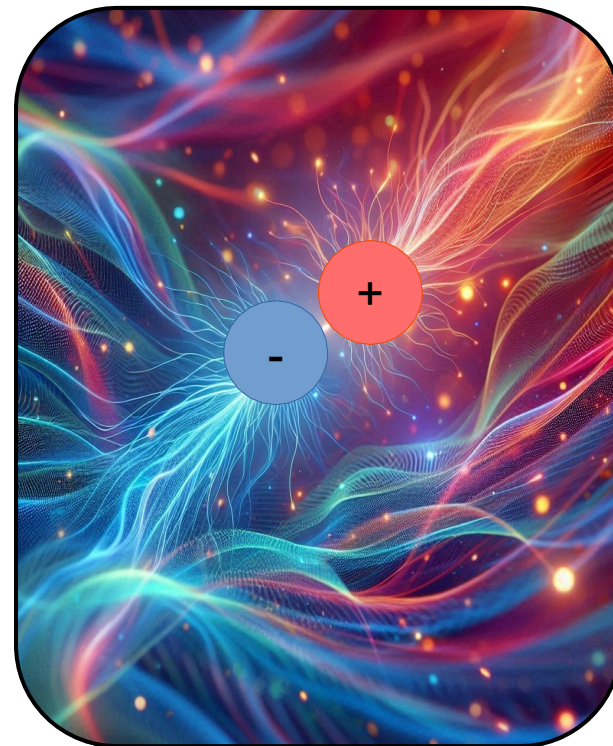
- Multiple CPU Architectures
- Multiple GPU vendors
- Multiple programming models



Source: EuroHPC JU

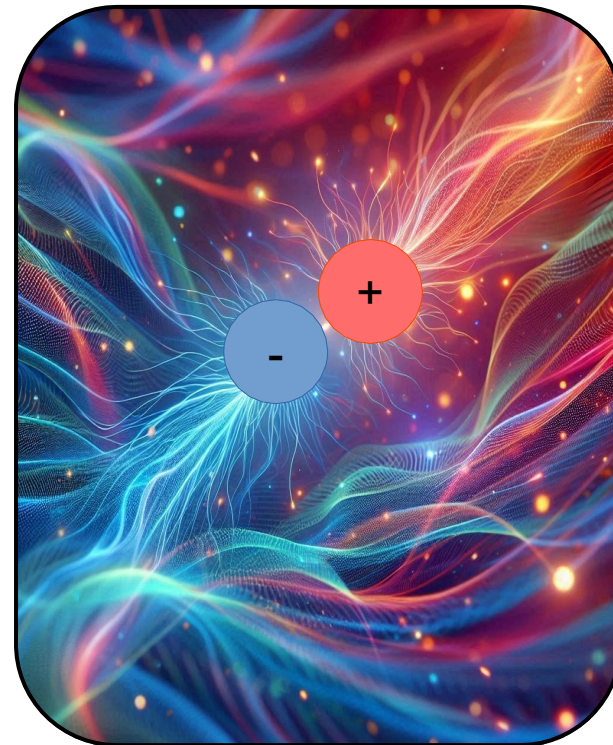
Electrostatics

- Molecular Dynamics and Soft Matter
- Long range interactions



Electrostatics

- Molecular Dynamics and Soft Matter
- Long range interactions
- **Periodic Boundary Conditions**



Electrostatics

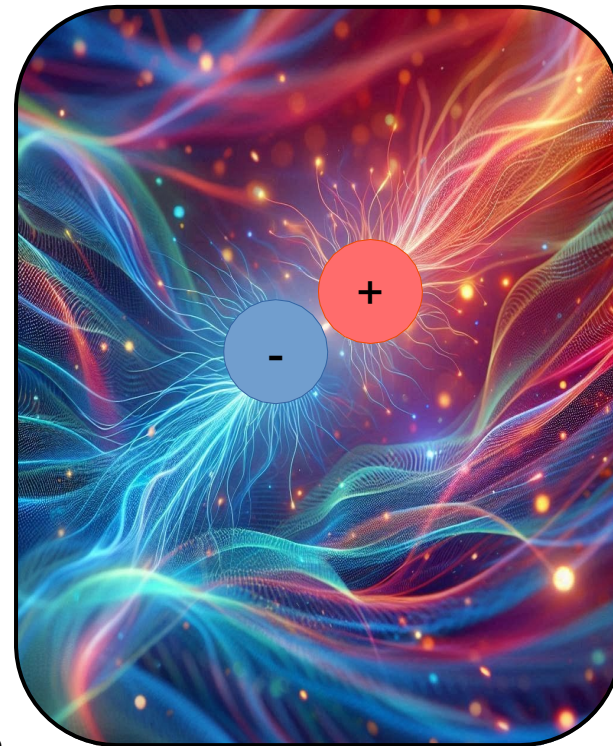
- Molecular Dynamics and Soft Matter
- Long range interactions
- **Periodic Boundary Conditions**

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

 $\tilde{\rho}(\mathbf{k})$ $\rho_M(\mathbf{r}_p)$

Ewald Summation

Particle-Particle Particle-Mesh



Why does it matter for scaling?

$$O(N^2)$$

$$O(N^{3/2})$$

$$O(N \log N)$$

- Direct Summation
- Ewald Summation
- Mesh based

Why does it matter for scaling?

$$O(N^2)$$

- Direct Summation

$$O(N^{3/2})$$

- Ewald Summation

- Fast Fourier Transforms

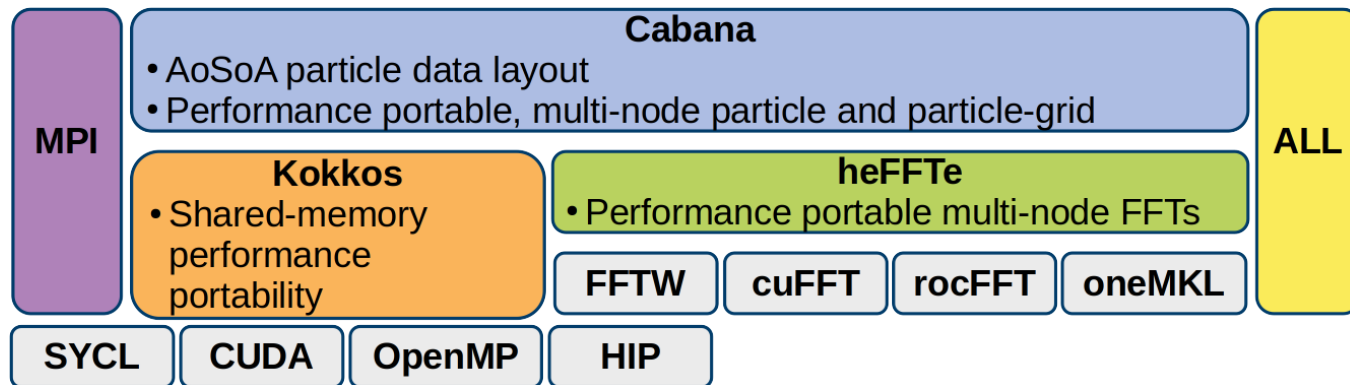
$$E(\mathbf{r}_p) = \text{FFT} \left[-i \mathbf{k} \cdot \text{FFT} [\rho_M] \hat{G}_{opt} \right] (\mathbf{r}_p)$$

GROMACS ; AMBER ; LAMMPS

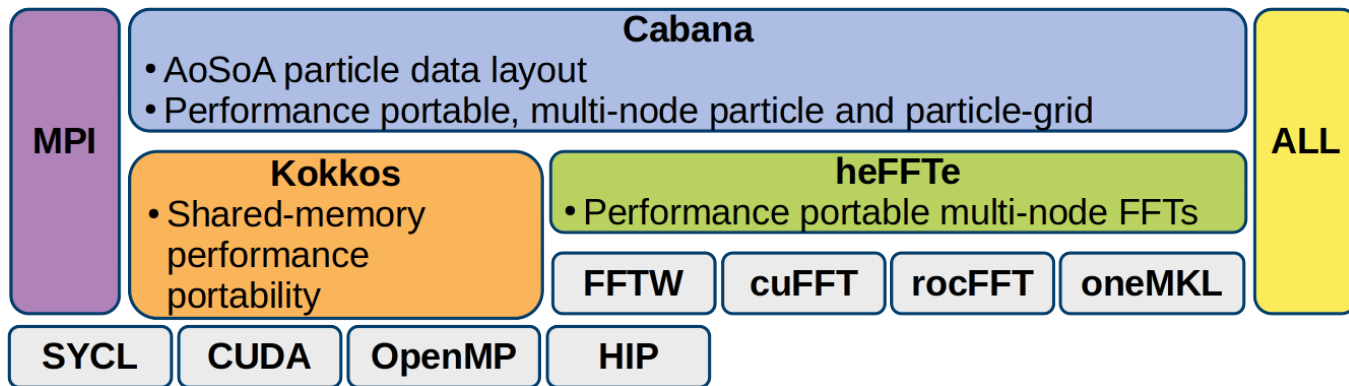
$$O(N \log N)$$

- Mesh based
- P3M
- PME
- SPME

Dependencies and CI CD

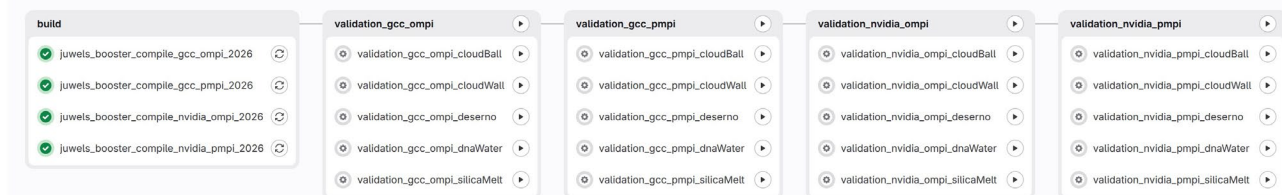


Dependencies and CI CD

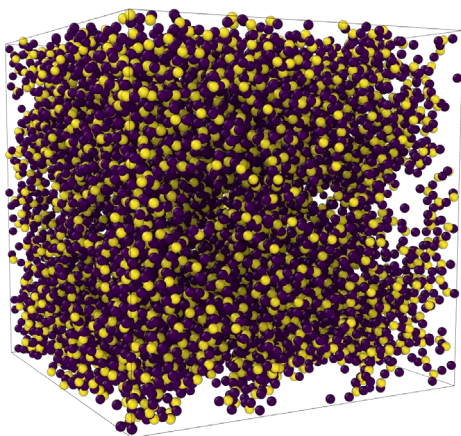


Pipeline Jobs 28 Tests 0

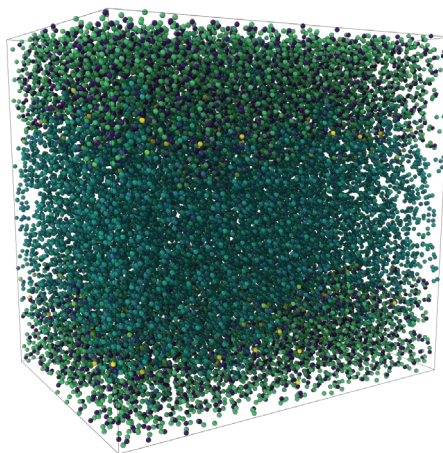
Group jobs by Stage Job dependencies



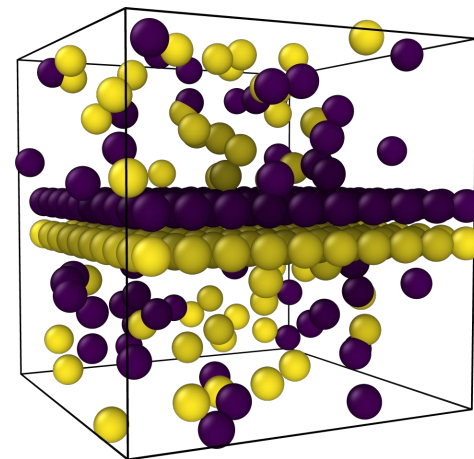
Validation Systems



Silica Melt (12960 particles)



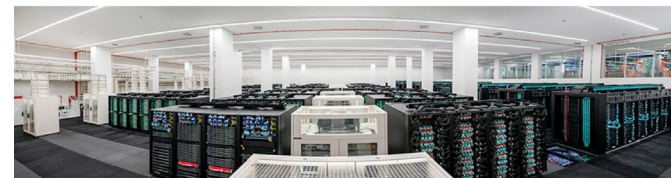
Rhodopsin(32000 particles)



Cloud Wall (300 particles)

Setup

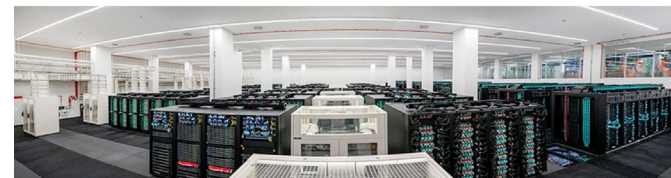
- EuroHPC JU Developer access
- Node configuration
 - 2 Xeon Platinum 8460Y CPUs
 - 4 H100 64GB HBM2 GPUs
 - 512 GB of 4800MHz DDR5



Source: EuroHPC JU

Setup

- EuroHPC JU Developer access
- Software stack
 - GCC 13.2.0
 - CUDA 12.8
 - Kokkos 4.7.2
 - Cabana 0.8.0
 - heFFTe 2.4.1 (cuFFT backend)
- Node configuration
 - 2 Xeon Platinum 8460Y CPUs
 - 4 H100 64GB HBM2 GPUs
 - 512 GB of 4800MHz DDR5

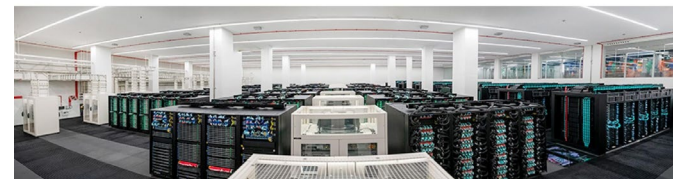


Source: EuroHPC JU

Setup

- EuroHPC JU Developer access
- Software stack
 - GCC 13.2.0
 - CUDA 12.8
 - Kokkos 4.7.2
 - Cabana 0.8.0
 - heFFTe 2.4.1 (cuFFT backend)
- -O2 optimization
- Host parallelization: OpenMP
- Device execution: CUDA

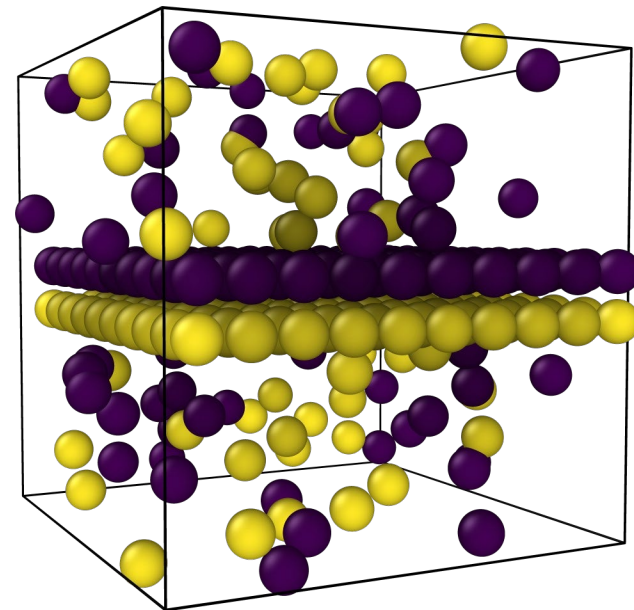
- Node configuration
 - 2 Xeon Platinum 8460Y CPUs
 - 4 H100 64GB HBM2 GPUs
 - 512 GB of 4800MHz DDR5



Source: EuroHPC JU

System and parameters

- Creates an artificial dipole
- High long-range contribution



Base system: Cloud Wall (300 particles)

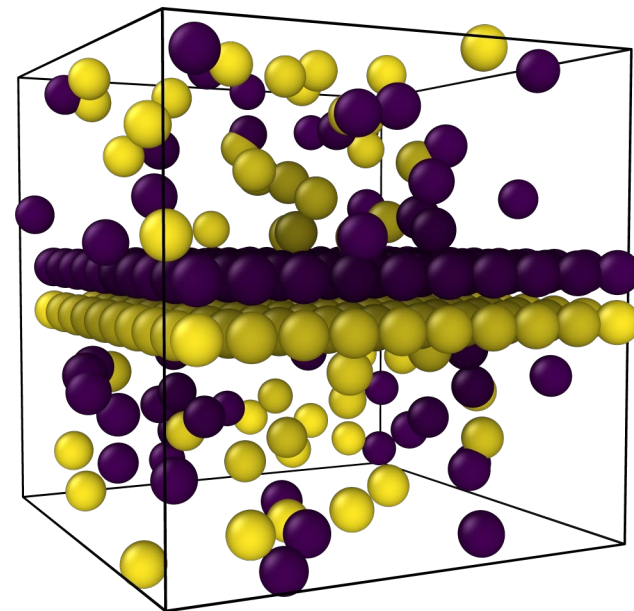
System and parameters

- Creates an artificial dipole
- High long-range contribution

Reciprocal space parameters of the P3M solver:

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



Base system: Cloud Wall (300 particles)

Validation at large scale

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
```

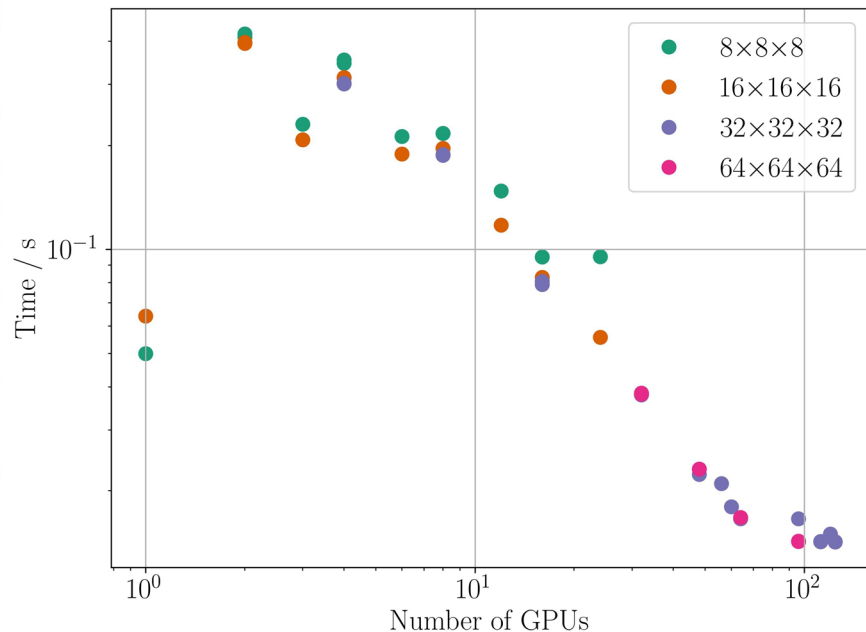
RMS potential : 4.02801676349598e-07

RMS fields: 7.64805475718963e-08 8.4353410644898e-08 3.97391771302327e-07

Time spent in run [Ewald] (avg, min, max): 2086.3770867 s 2085.901671 s 2086.535621 s

Time spent in run [P3M] (avg, min, max): 0.5853885 s 0.585388 s 0.585389 s

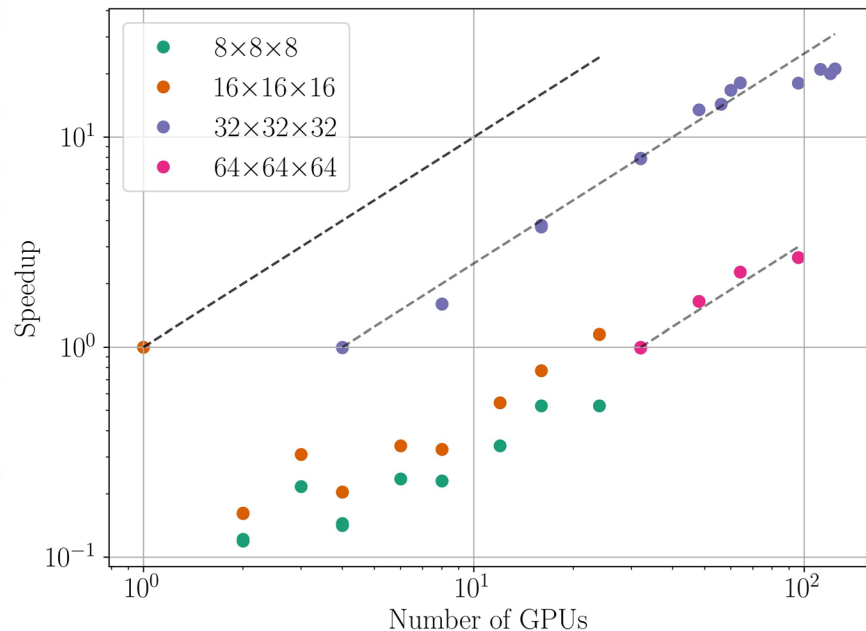
Benchmarks



Strong-scaling benchmark results. Replication numbers (8 to 64) on 1 to 128 GPUs (1 to 32 nodes) on MN5 Acc.

- Normalized to smallest system size
- Parallel overhead
- FFT dominates

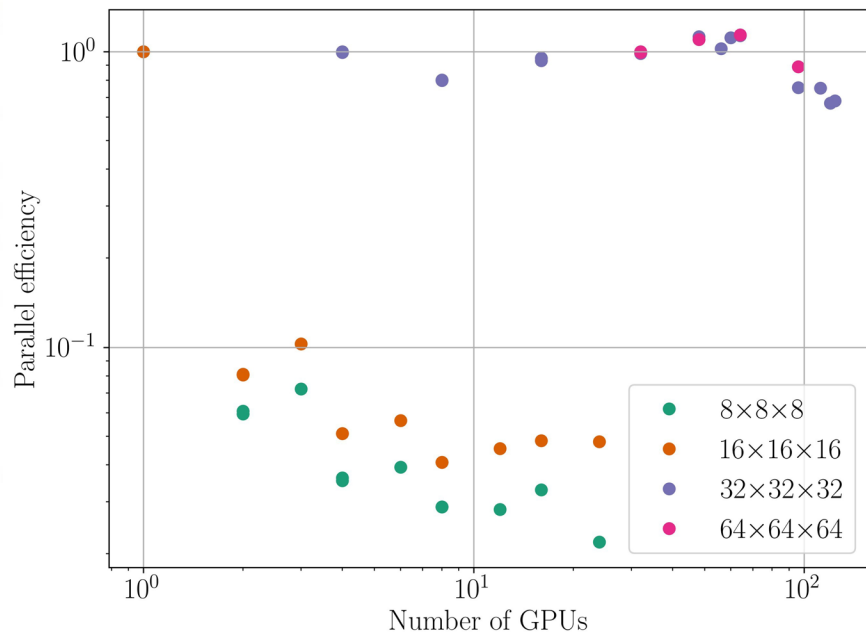
Speedup



- Multi-GPU parallel overhead cannot be compensated for small configurations
- Optimal scalability for large systems

Dashed lines represent the ideal speedup taking as reference the smallest amount of resources that could compute the system size.

Parallel efficiency



Strong-scaling benchmark results. Replication numbers (8 to 64) on 1 to 128 GPUs (1 to 32 nodes) on MN5 Acc.

- Parallel efficiency remains close to unity for larger systems
- Cache and load-balancing effects
- Computational intensity

x

Communication overhead

Conclusions

- P3M solver runs efficiently on MN5
 - Viability on modern GPU-cluster hardware

Conclusions

- P3M solver runs efficiently on MN5
 - Viability on modern GPU-cluster hardware
- FFT grid fits into the memory of a single GPU?
 - single-GPU execution is optimal
 - two or more GPUs adds parallel overhead

Conclusions

- P3M solver runs efficiently on MN5
 - Viability on modern GPU-cluster hardware
- FFT grid fits into the memory of a single GPU?
 - single-GPU execution is optimal
 - two or more GPUs adds parallel overhead
- For large grids k-space portion scales
 - adding GPUs can improve performance

Conclusions

- P3M solver runs efficiently on MN5
 - Viability on modern GPU-cluster hardware
- FFT grid fits into the memory of a single GPU?
 - single-GPU execution is optimal
 - two or more GPUs adds parallel overhead
- For large grids k-space portion scales
 - adding GPUs can improve performance
- Crossover region is system dependent

Outlook

- Systematic parametric study
 - memory-size vs. parallel-overhead crossover

Outlook

- Systematic parametric study
 - memory-size vs. parallel-overhead crossover
- Dynamic load-balancing strategies
 - Partition adjustment between real- and k-space

Outlook

- Systematic parametric study
 - memory-size vs. parallel-overhead crossover
- Dynamic load-balancing strategies
 - Partition adjustment between real- and k-space
- Real-space splitting parameter

Outlook

- Systematic parametric study
 - memory-size vs. parallel-overhead crossover
- Dynamic load-balancing strategies
 - Partition adjustment between real- and k-space
- Real-space splitting parameter
- Coupling the library into ESPResSo

Outlook

- Systematic parametric study
 - memory-size vs. parallel-overhead crossover
- Dynamic load-balancing strategies
 - Partition adjustment between real- and k-space
- Real-space splitting parameter
- Coupling the library into ESPResSo
- Comparison with AMD GPUs
 - LUMI Acc access granted (EHPC-DEV-2025D11-078)

Scaling and performance aspects of the P3M method on multiple GPUs

Rodrigo A. C. Bartolomeu, René Halver and Godehard Sutmann

Simulation and Data Laboratory Complex Particle Systems

Jülich Supercomputing Centre (JSC) - Forschungszentrum Jülich





EuroHPC and
User Days
22→24 September 2026

Dublin Ireland
22→24
September 2026



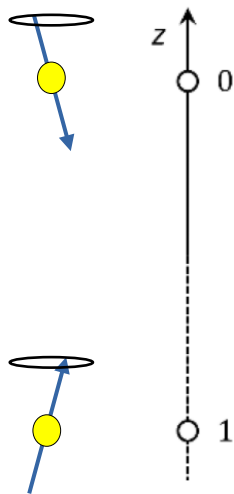
High-throughput discovery of magnetic molecules for quantum technologies.

Alessandro Lunghi

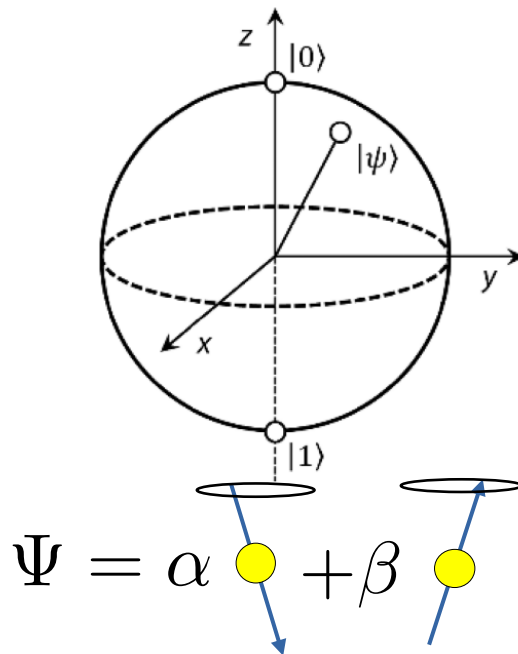
Trinity College Dublin - lunghia@tcd.ie

Spin is at the origin of magnetism, but also exhibit quantum behaviour at the nano and atomic scale

Classical bit

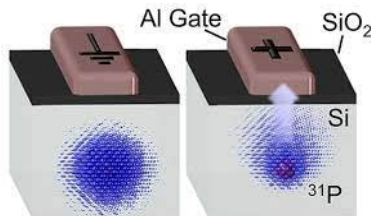
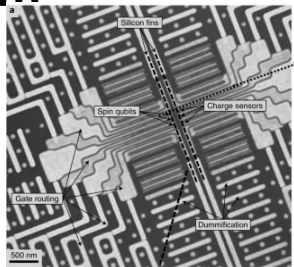


Quantum bit
Qubit

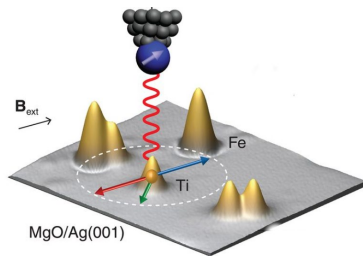
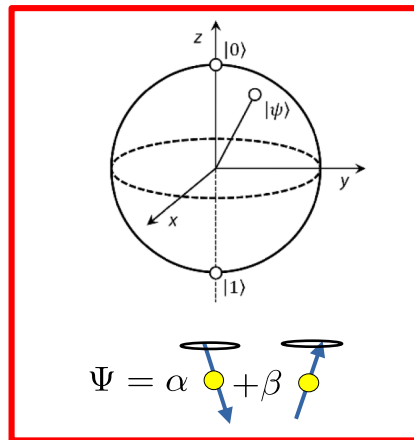


Spin is an ideal degree of freedom for the realization of quantum technologies

Silicon Quantum dots

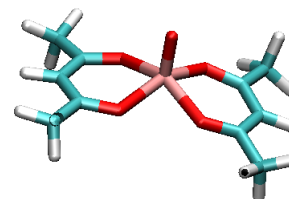
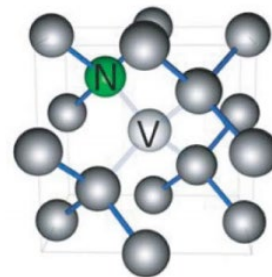


Dopants in Semiconductors



Atoms on Surface

Solid-state defects



Magnetic Molecules

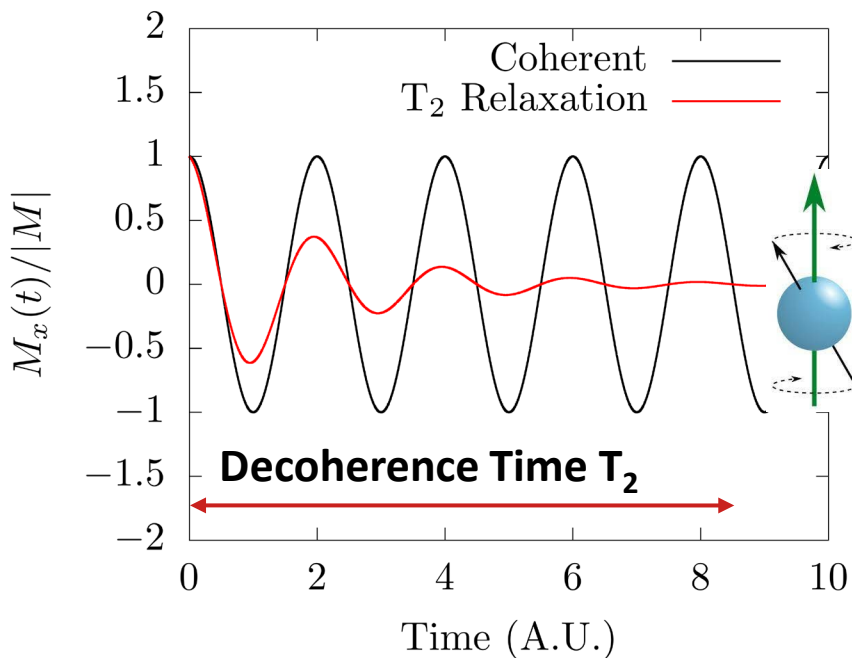
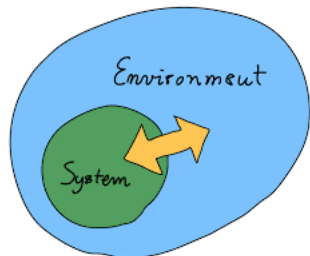
Decoherence is the “loss of QM behaviour” due to the interaction of qubits with the environment

Isolated Qubit:

$$i\hbar|\dot{\psi}\rangle = \hat{H}|\psi\rangle$$

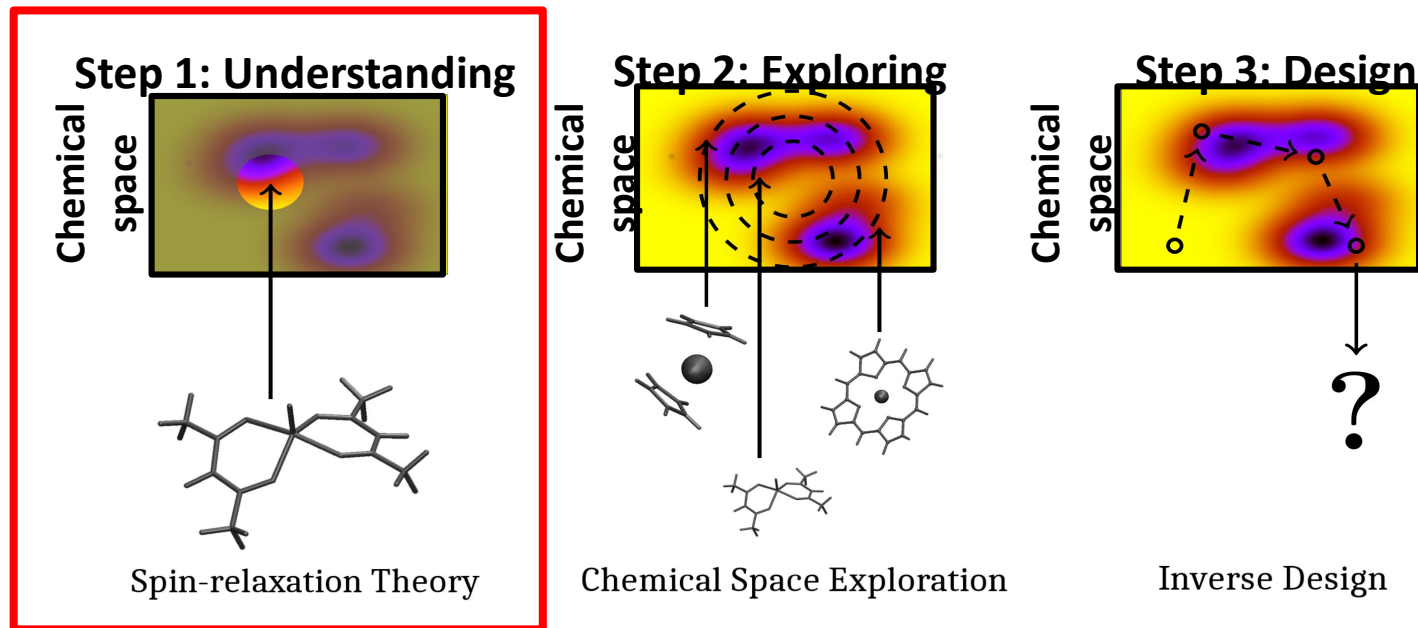
Qubit + Environment:

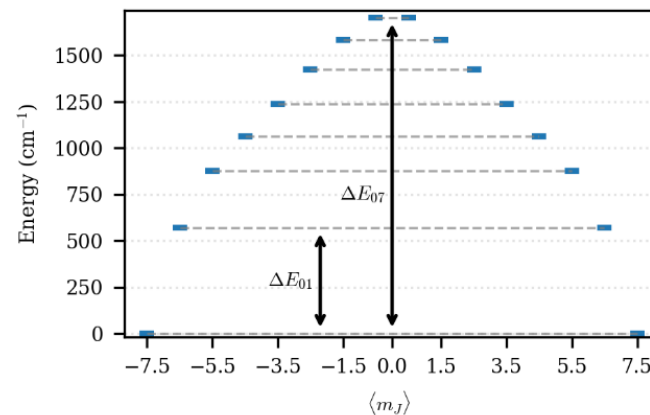
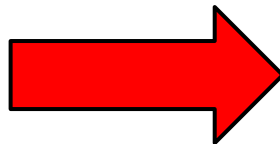
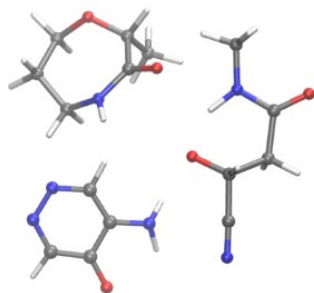
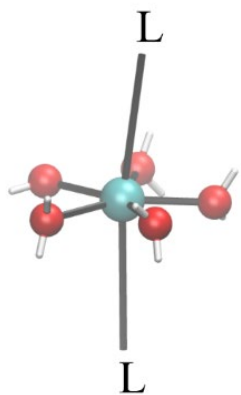
~~$$i\hbar|\dot{\psi}\rangle = \hat{H}|\psi\rangle$$~~



What does it take to design a molecular qubit with target properties?

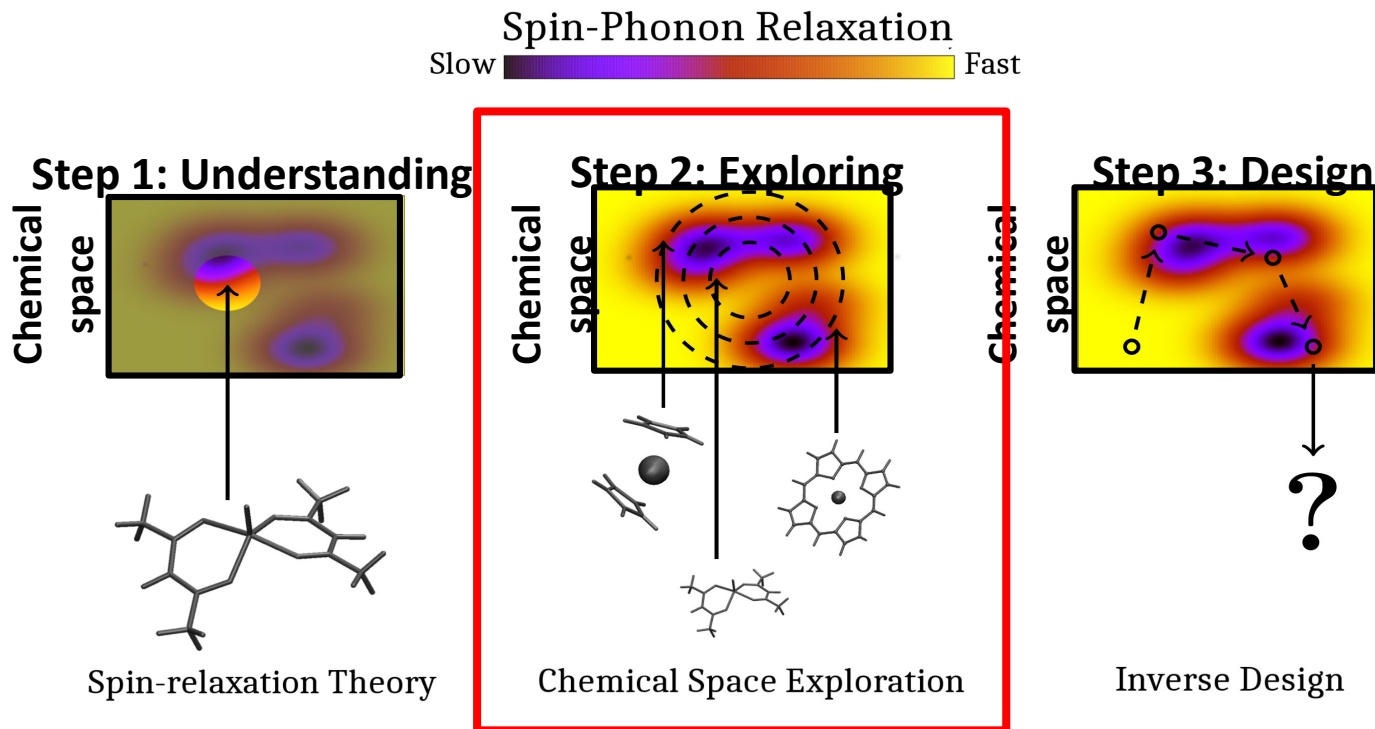
Spin-Phonon Relaxation
Slow  Fast

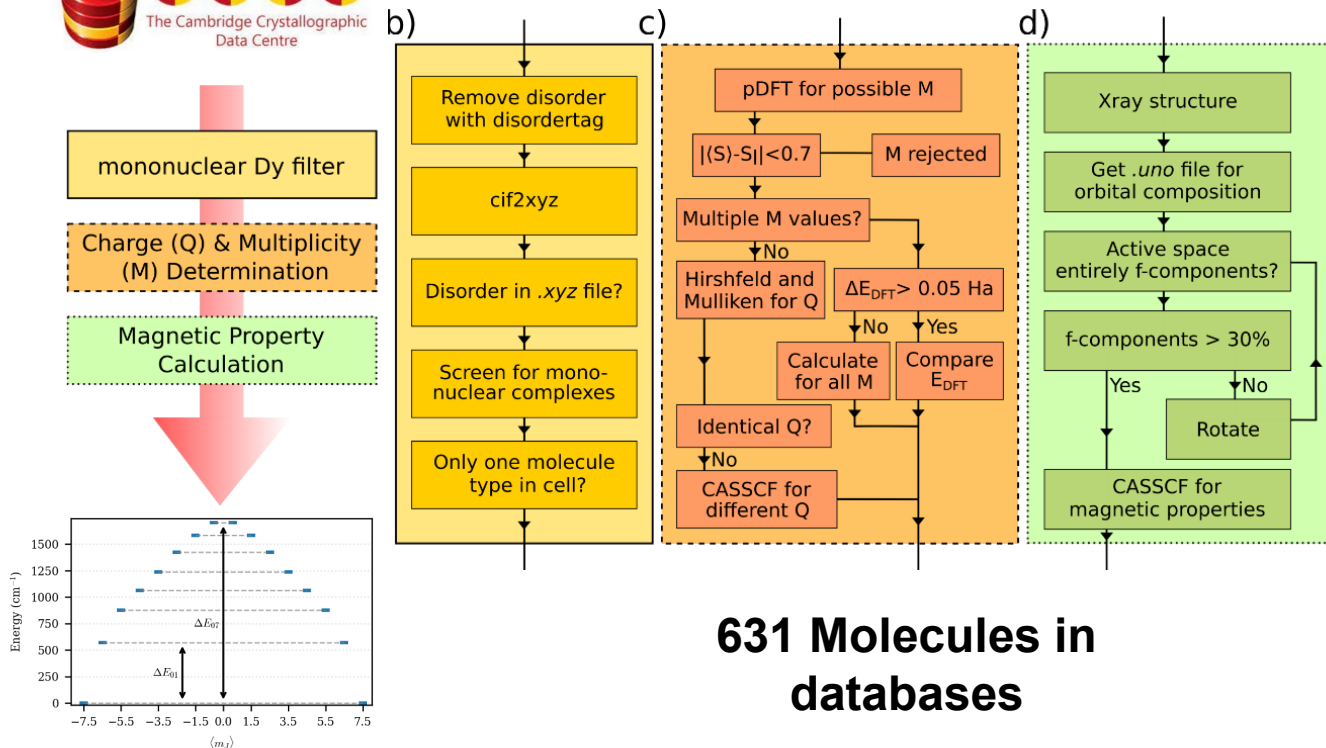


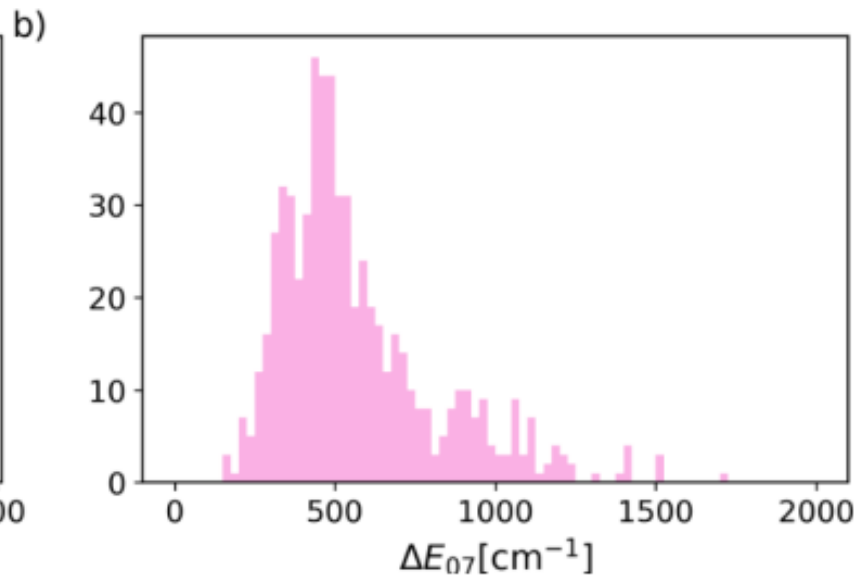
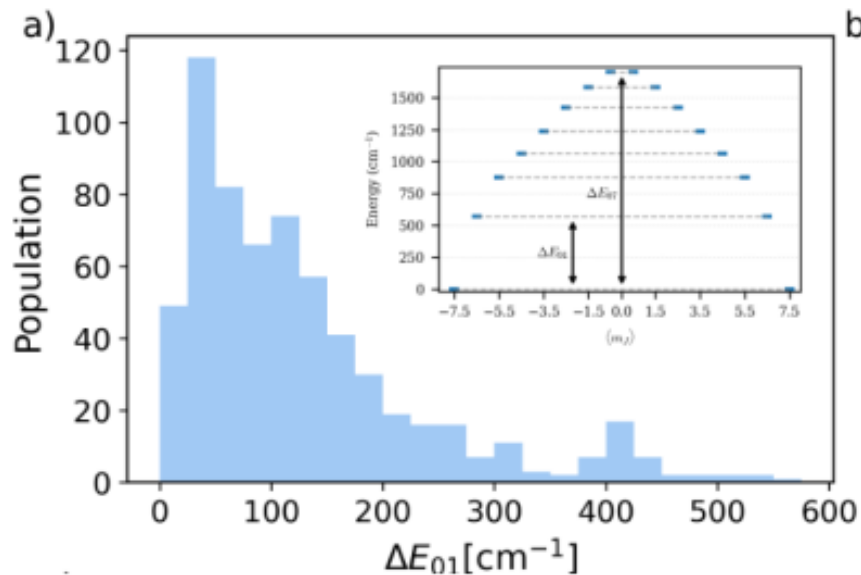


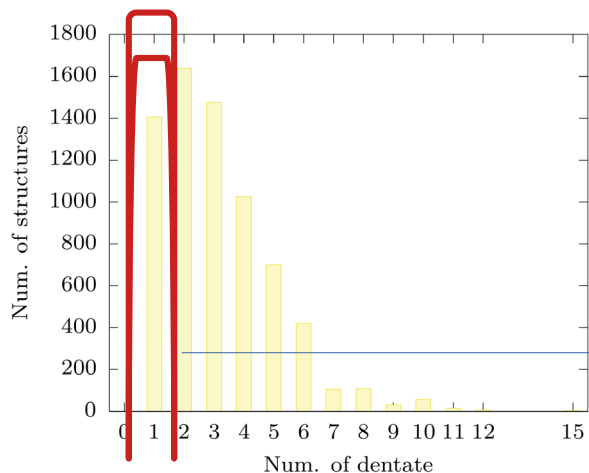
The magnetic moment of the molecule experiences an energy barrier that prevents reorientations, slowing down its dynamics

What does it take to design a molecular qubit with target properties?





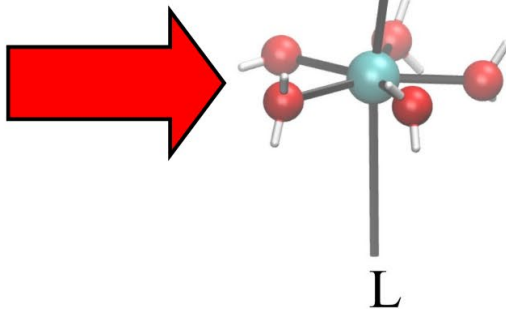
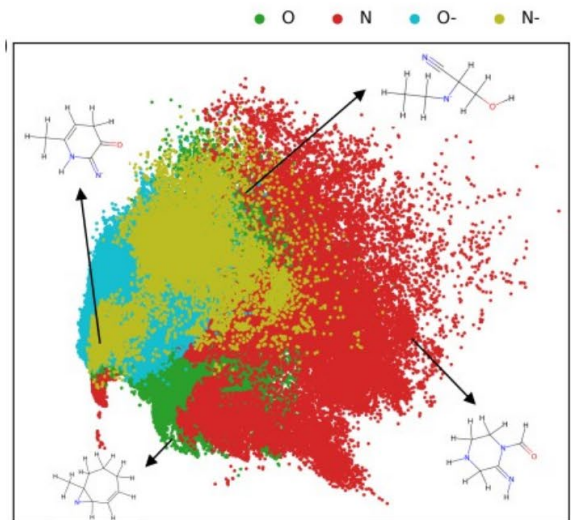




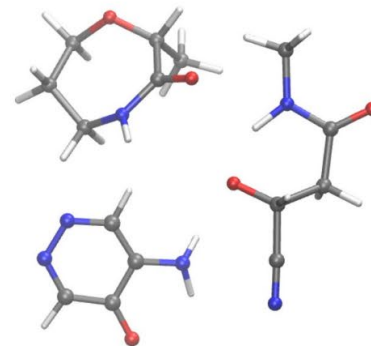
**You can generate almost a million
4-coordinate compounds with any
pair of 1400 ligands**

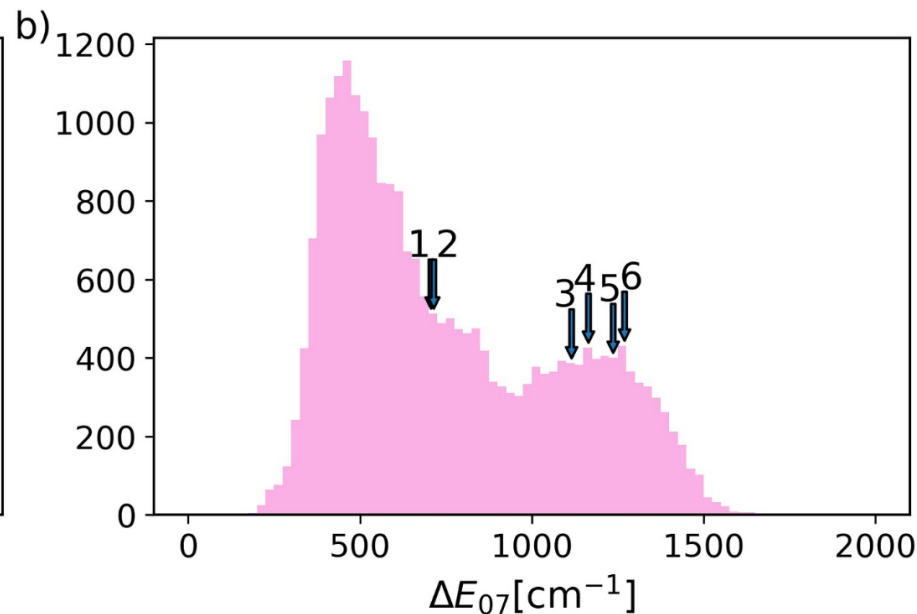
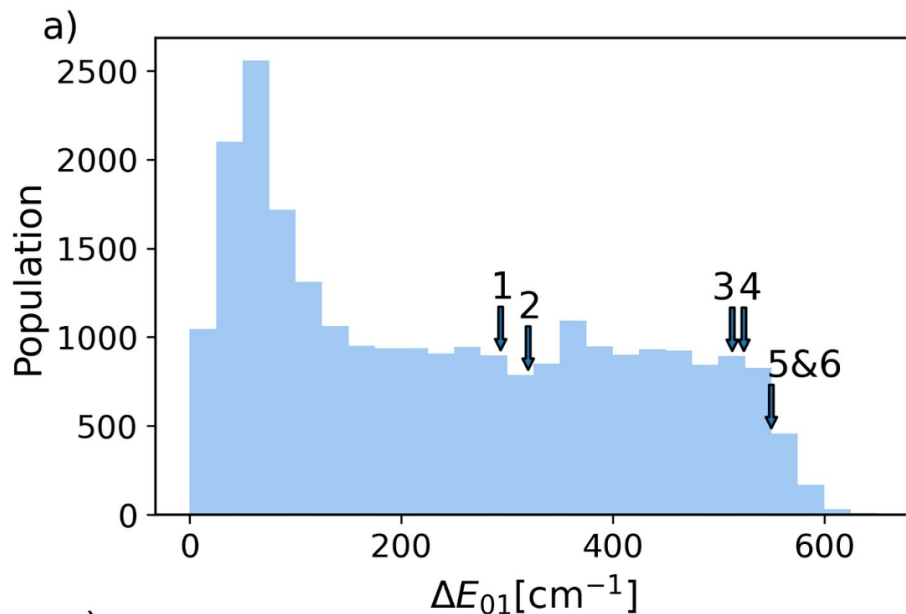
$$\binom{1400}{2} = 979,300$$

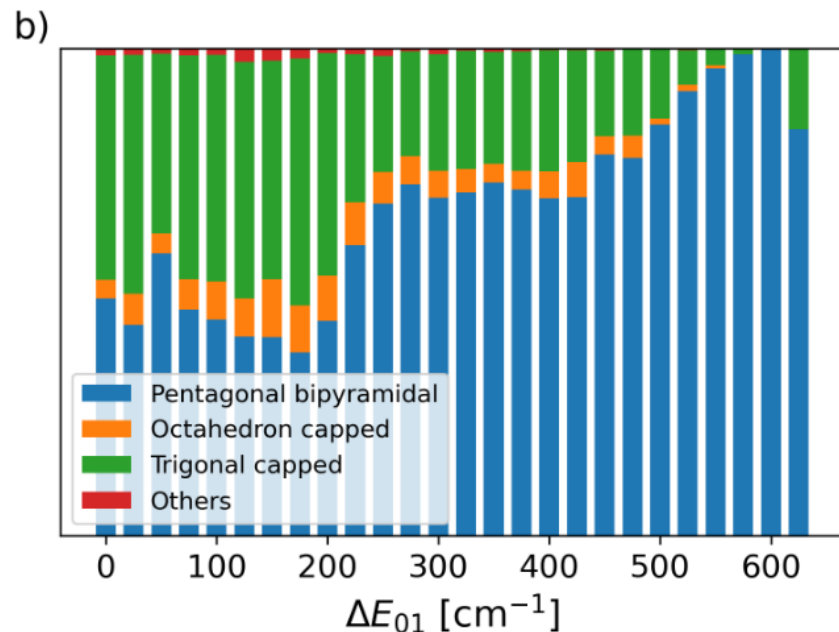
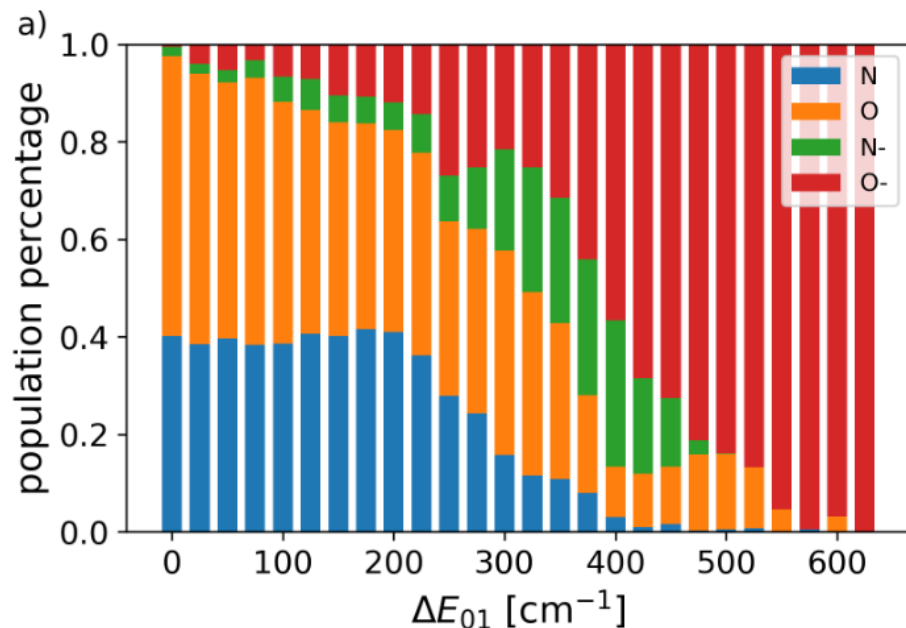
**With monodentate ligands alone we have an incredibly large
chemical space at disposal**

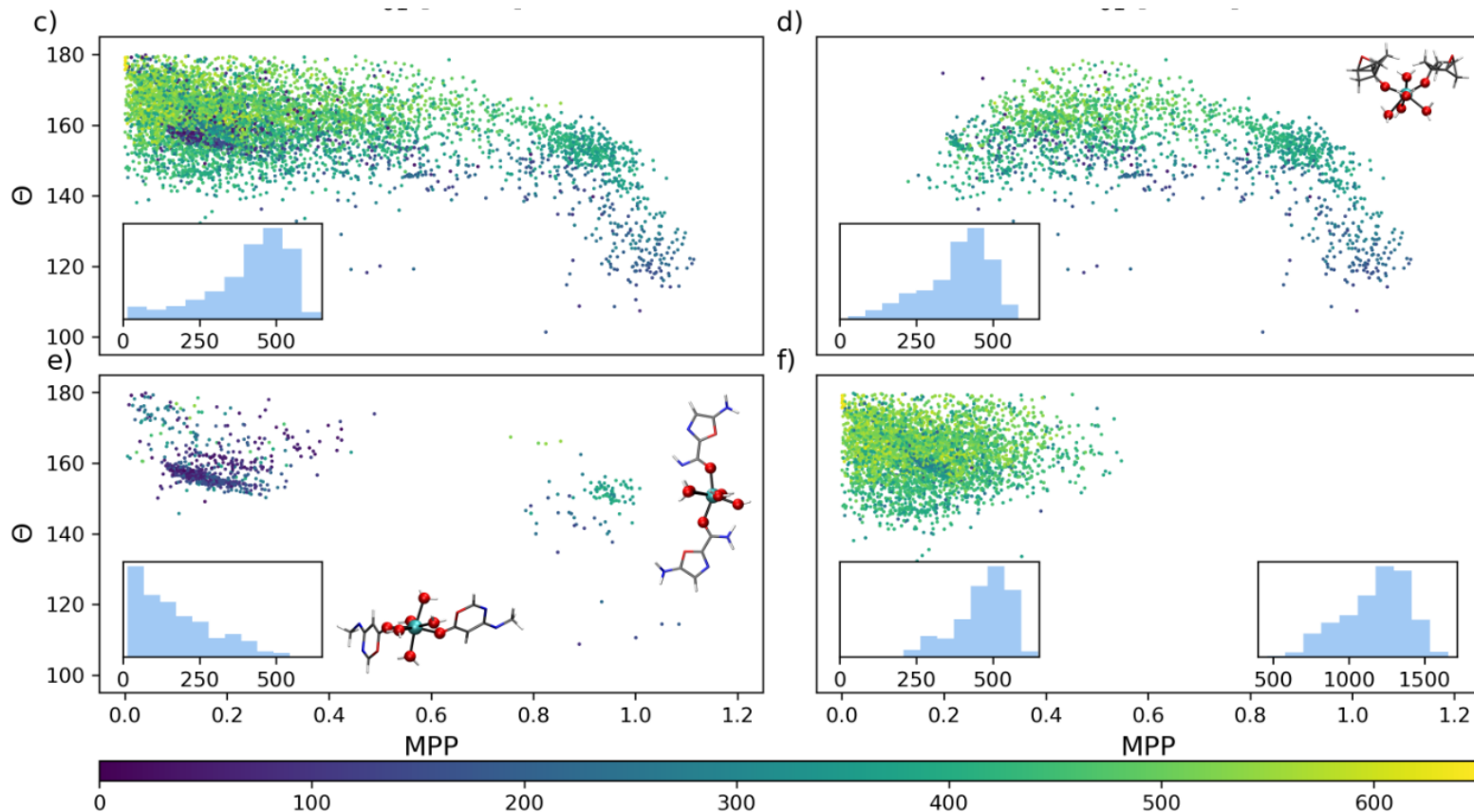


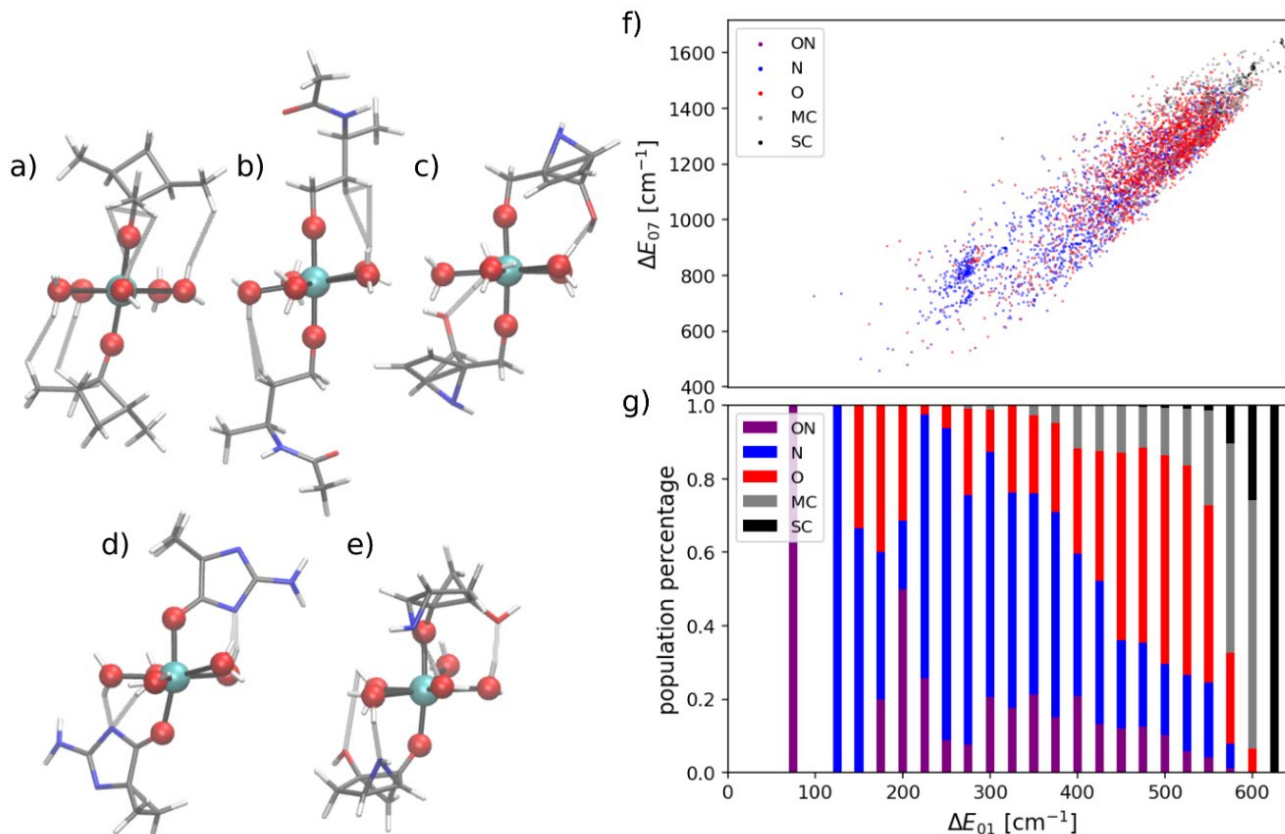
L:



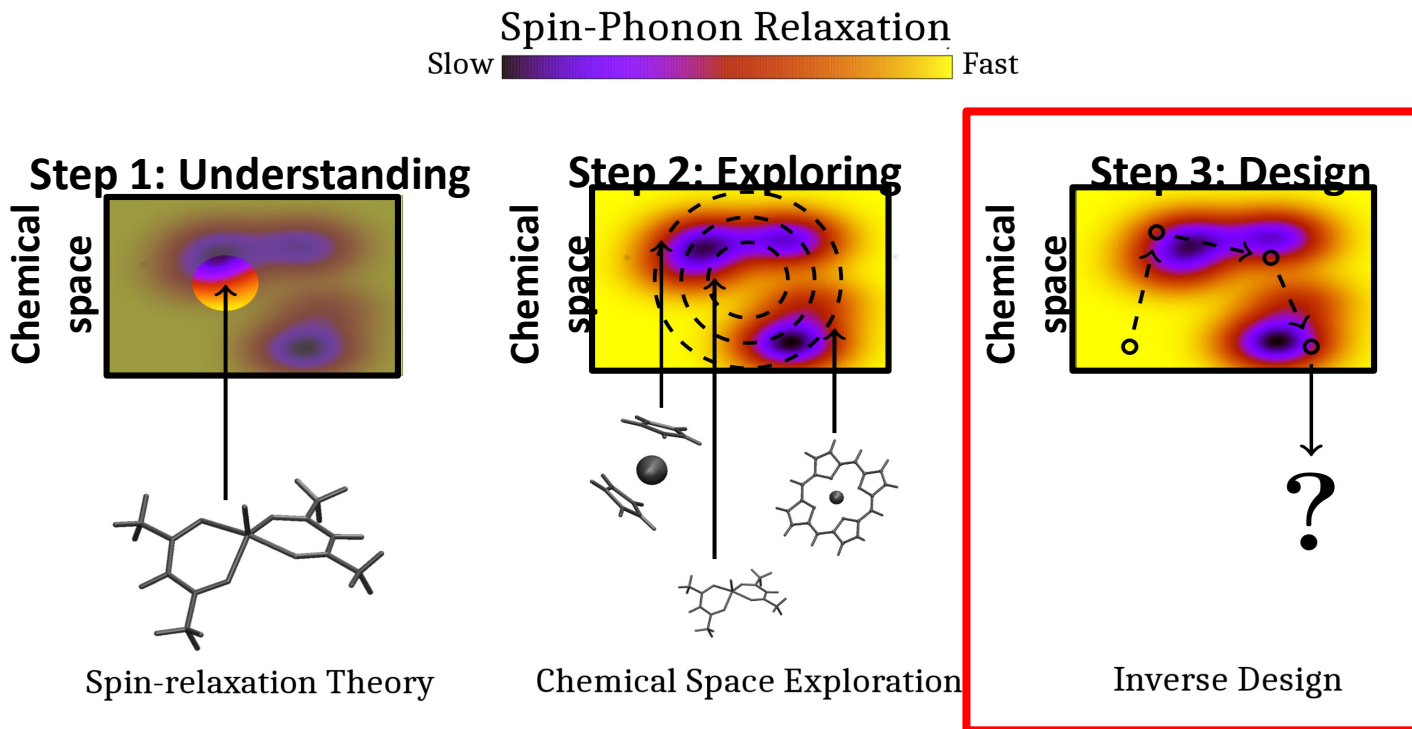




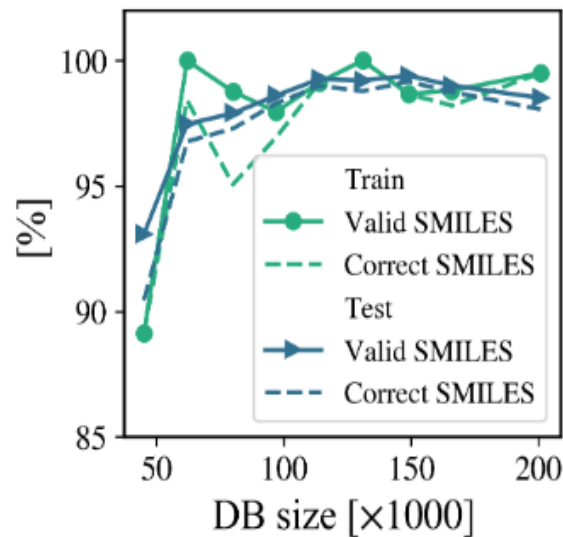
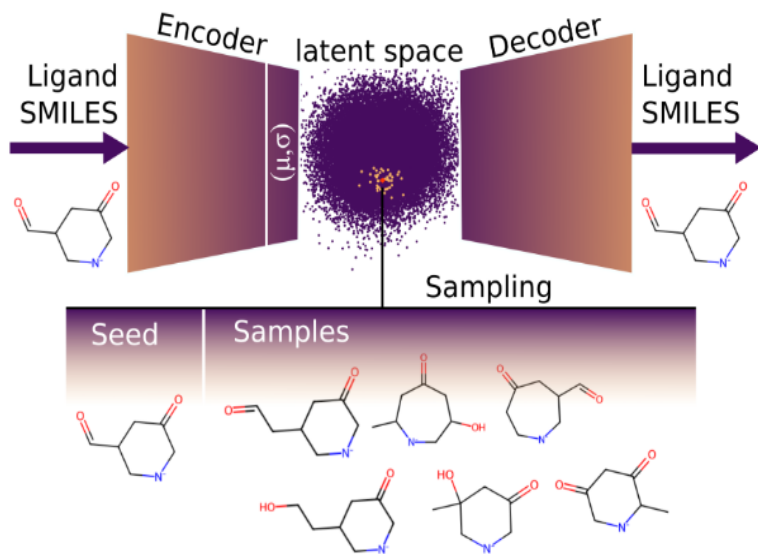




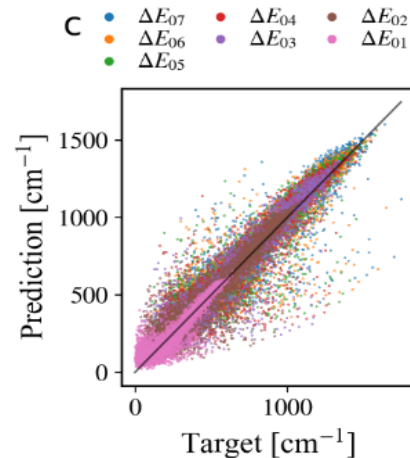
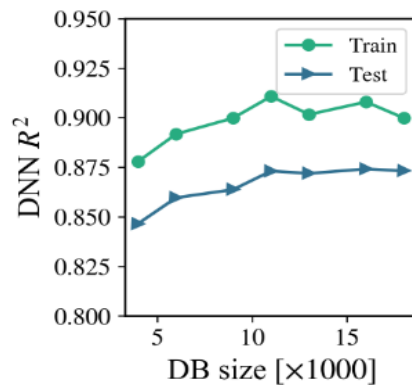
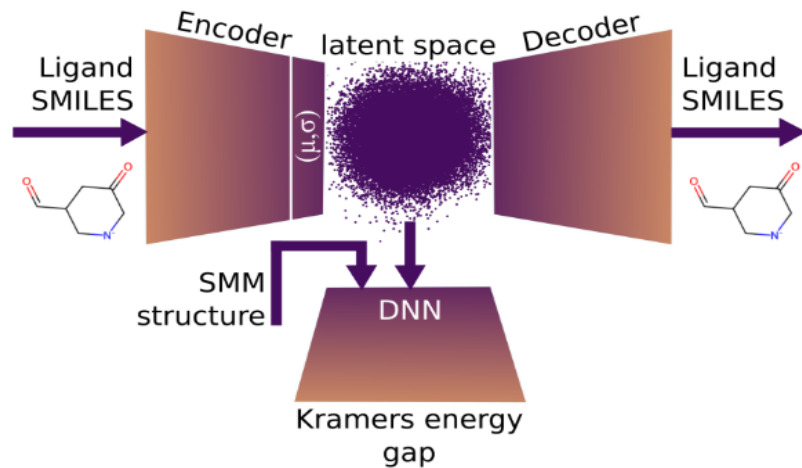
What does it take to design a molecular qubit with target properties?



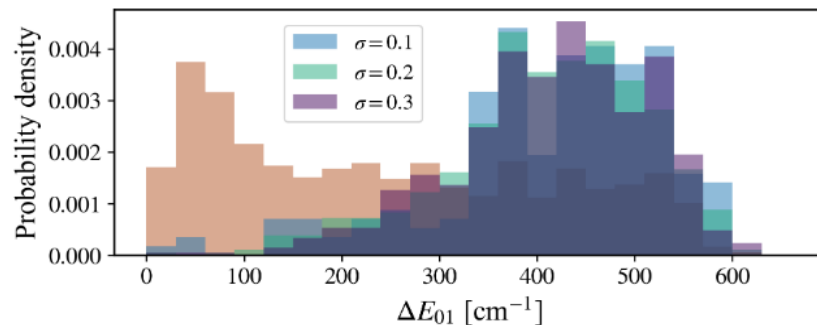
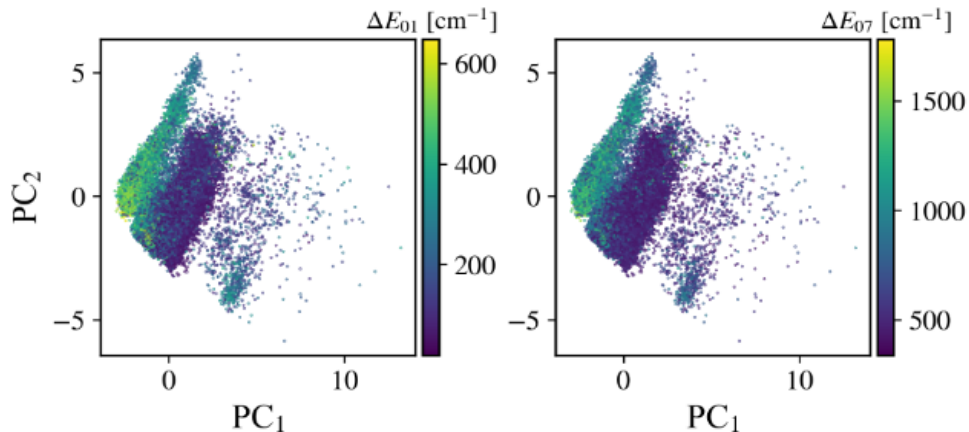
Variational Autoencoders allow the generation of organic ligands



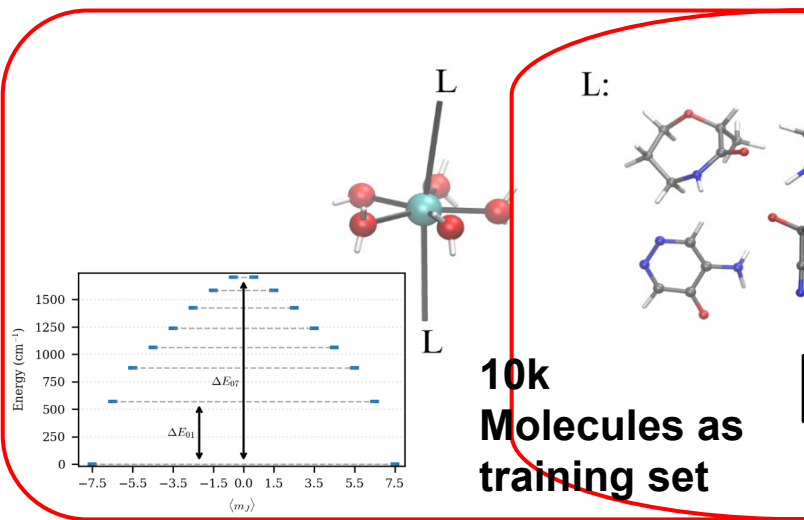
The VAE latent space is expanded with structural properties and a property predictor is trained jointly



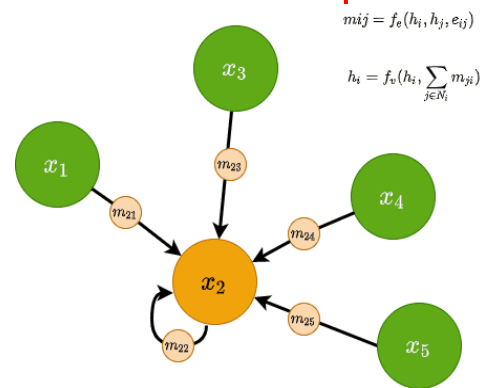
The VAE latent space is becomes structures and allows for the sampling of only good candidates



Graph-neural networks can accurately learn to predict excitations in Dy complexes

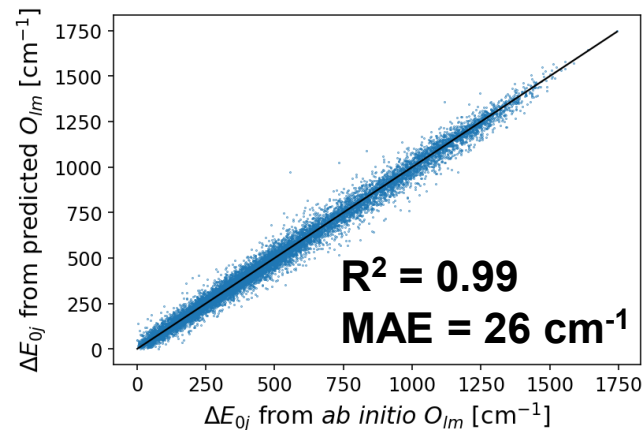
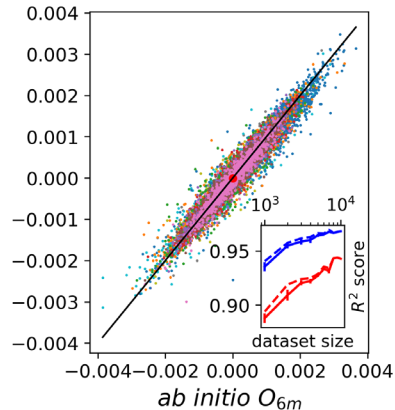
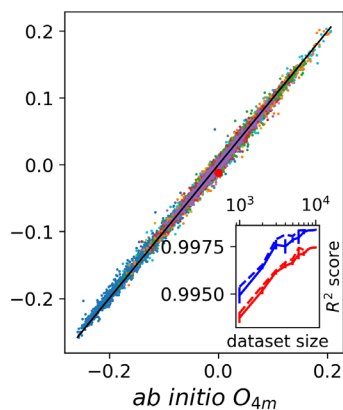
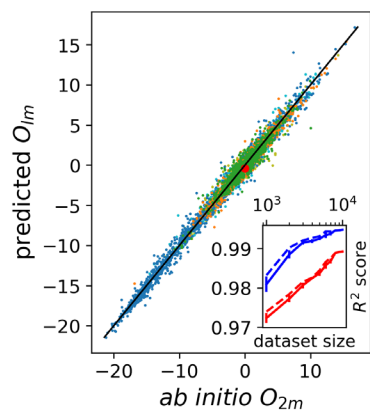


Equvariant graph neural networks



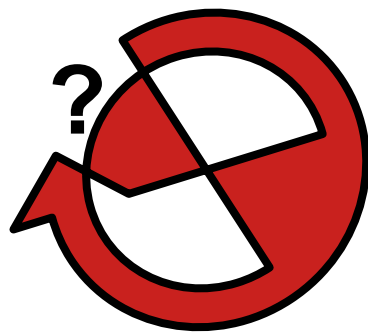
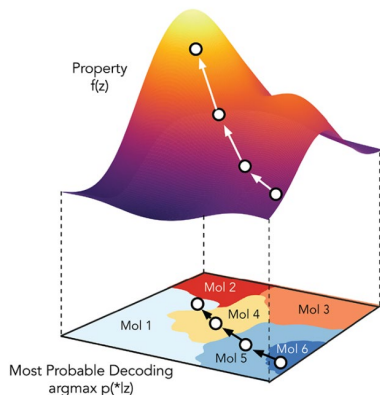
Graph-neural networks can accurately learn to predict excitations in Dy complexes

$$\hat{H} = \sum_{l=2,4,6} \sum_{m=-l}^{2J_i} B_{l,m} \hat{O}_{l,m}(\vec{J})$$



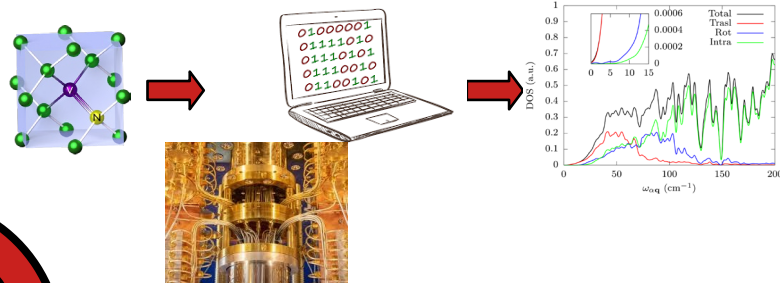
Open quantum system theory

$$\frac{d\hat{\rho}^s(t)}{dt} = -\frac{1}{\hbar^2} \int_0^t ds \operatorname{tr}_B[\hat{H}_{sph}(t), [\hat{H}_{sph}(s), \hat{\rho}(s)]]$$

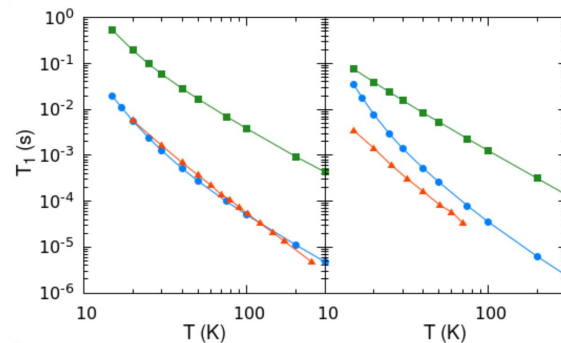
Generative
Molecular DesignMachine learning
Acceleration

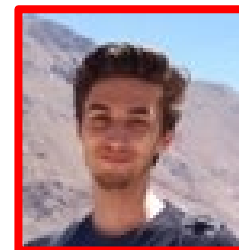
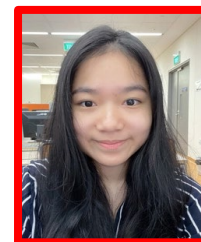
$$T_m^l = \sum_a^{N_a} T_m^l(a) = \sum_a^{N_a} o[\{c(a)\}, \{B(a)\}] \bar{Y}_m^l(a)$$

Quantum Chemistry



Decoherence theory





Alessandro Lunghi

School of Physics, AMBER and CRANN Institute, Trinity College, Dublin 2, Ireland

email: lunghia@tcd.ie

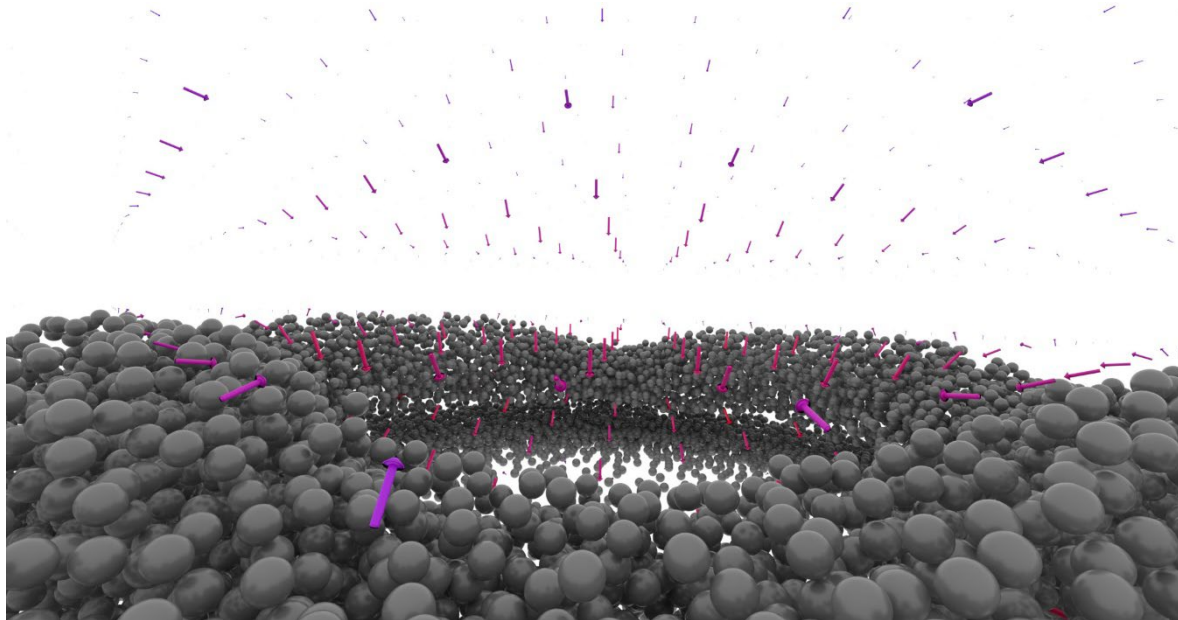
Coupling ESPResSo and waLBerla for scalable soft matter simulations with hydrodynamic interactions

Jean-Noël Grad, Ingo Tischler, Alexander Reinauer, Hideki Kobayashi,
Michael Kuron, Frederik Hennig, Markus Holzer, Pedro Santos Neves,
Satish Kamath, Christian Holm, Rudolf Weeber

CoE MultiXscale

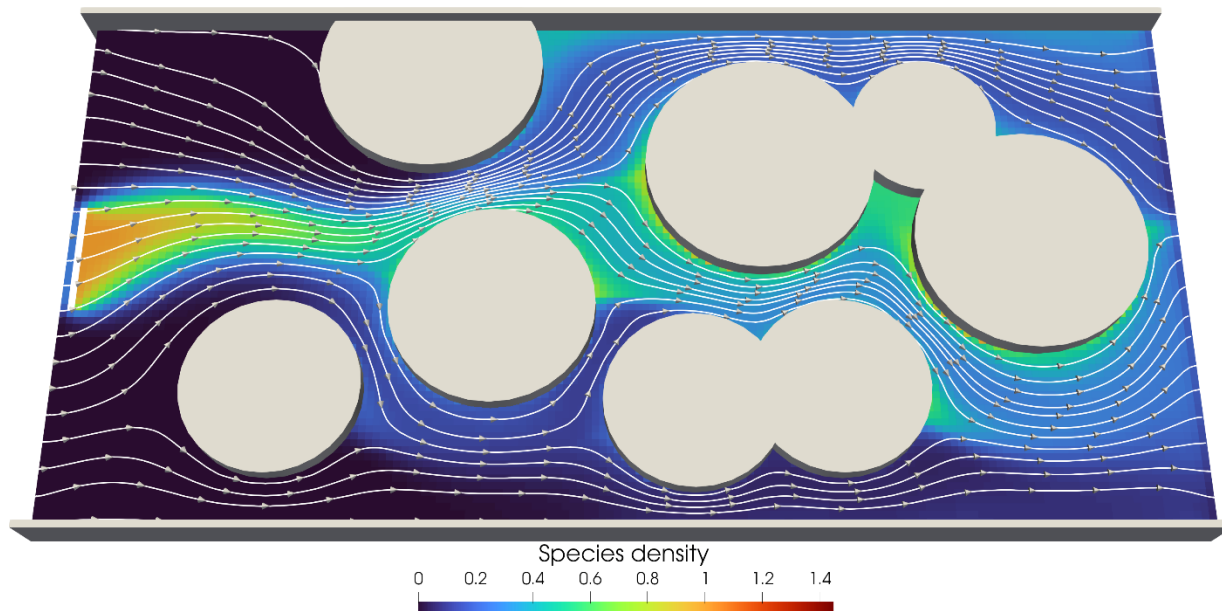
Multiscale modeling

- particles represent groups of atoms
- fluid is a continuum



Continuum mechanics

- dissolved species advect and diffuse
- species react on catalytic surfaces
- ionic species need electrostatics



The ESPResSo pilot application

- scalable couplings across length- and time-scales
- performance portability
 - replace custom codes by pre-exascale-ready libraries
 - use code generation to remain relevant on new hardware
- re-usable interfaces
 - waLBerla bridge for molecular dynamics packages

Automatic code generation

- rapid prototyping in Jupyter notebooks
- live execution and plotting
- export to C++/CUDA

```
In [1]: from pystencils.session import *  
from lbmpy.session import *  
lbm_config = LBMConfig(stencil=LBStencil(Stencil.D2Q9), method=Method.MRT_RAW)  
lb_method = create_lb_method(lbm_config=lbm_config)  
lb_method
```

Out[1]: **Moment-Based Method**

Stencil: D2Q9

Zero-Centered Storage: ✓

Force Model: None

Continuous Hydrodynamic Maxwellian Equilibrium

Compressible: ✗

Deviation Only: ✓

Order: 2

$$f(\rho, (u_0, u_1), (v_0, v_1)) = \frac{3\delta_\rho e^{-\frac{3v_0^2}{2} - \frac{3v_1^2}{2}}}{2\pi} - \frac{3e^{-\frac{3v_0^2}{2} - \frac{3v_1^2}{2}}}{2\pi} + \frac{3e^{-\frac{3(-u_0+v_0)^2}{2} - \frac{3(-u_1+v_1)^2}{2}}}{2\pi}$$

Relaxation Info

	Moment	Eq. Value	Relaxation Rate
	1	δ_ρ	ω
	x	u_0	ω
	y	u_1	ω
	x^2	$\frac{\delta_\rho}{3} + u_0^2$	ω
	y^2	$\frac{\delta_\rho}{3} + u_1^2$	ω

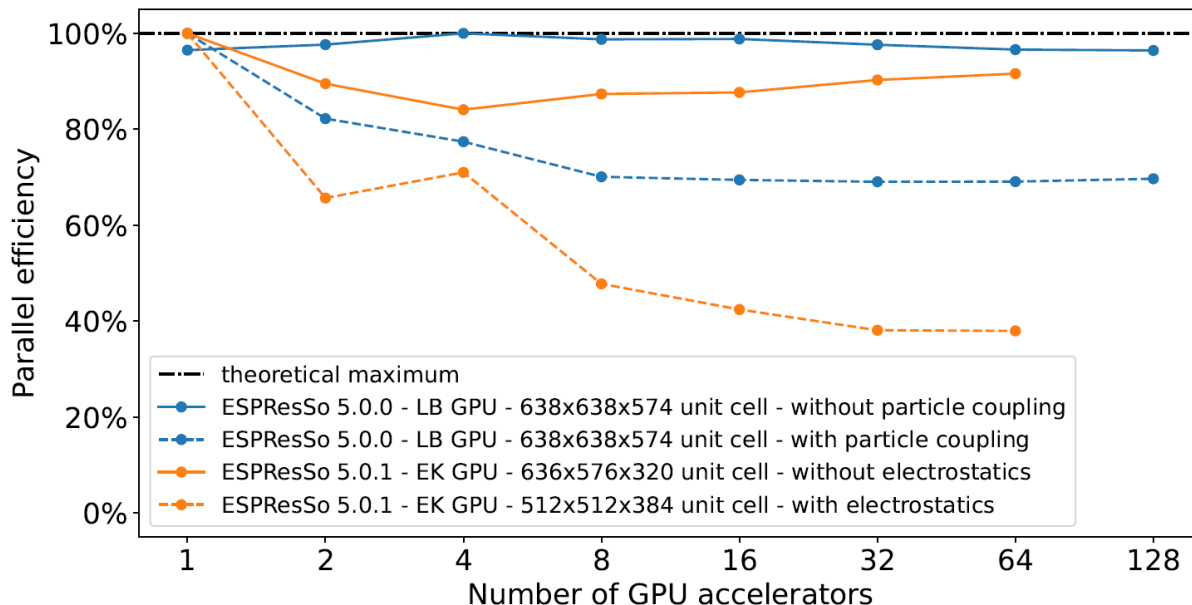
Hybrid programming model

- shared memory parallelism: 1 MPI rank per accelerator, multiple threads per rank
- kokkos defines data structures and execution policies, Cabana extends them for molecular dynamics

```
for (auto id = first; id <= last; ++id) { // old serial code  
    kernel(id);  
}  
// new parallel code  
Kokkos::RangePolicy policy(first, last + 1);  
Kokkos::parallel_for(policy, kernel);
```

Benchmarks on MareNostrum 5

- weak scaling of thermalised fluids
- particle coupling introduces latency
- FFT involves a lot of communication



Acknowledgments

Links to projects:

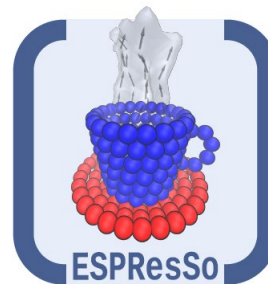
- MultiXscale: www.multixscale.eu
- ESPResSo: espressomd.org
- waLBerla: www.walberla.net
- Cabana: github.com/ECP-copa/Cabana
- kokkos: kokkos.org

LinkedIn:

- [@multixscale](https://www.linkedin.com/company/multixscale)

YouTube:

- [@MultiXscale](https://www.youtube.com/channel/UC...)
- [@espressosimulationpackage7832](https://www.youtube.com/channel/UC...)



Funded by the European Union. This work has received funding from the European High Performance Computing JU and countries participating in the project under grant agreement No [101093169](#). Funded by the German Research Foundation (DFG) grant No [528726435](#) and by the German Federal Ministry of Education and Research (BMFTR) grant No [16HPC095](#).