REENGINEERING SCIENTIFIC SIMULATION FOR MODERN ACCELERATORS

From PME to FMM: A Modern Approach to Long-Range Force Calculation in GROMACS

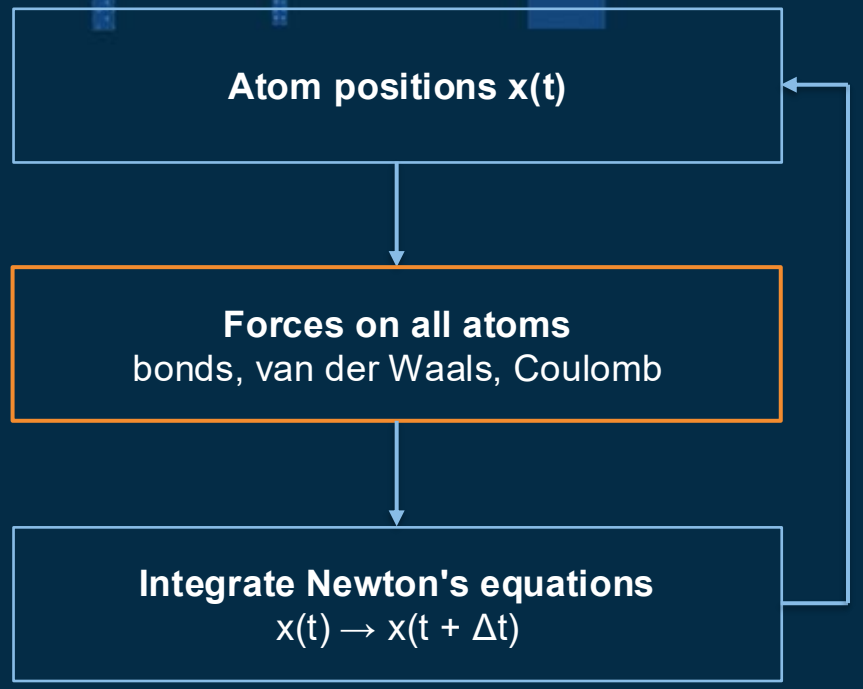
Jiamian Huang, Institute of Science Tokyo;
Muhammad Umair Sadiq, KTH Royal Institute of Technology;
Sebastian Keller, Swiss National Supercomputing Center;
Felix Thaler, Swiss National Supercomputing Center;
Berk Hess, KTH Royal Institute of Technology;
Rio Yokota, Institute of Science Tokyo



SC26 hpc unites | Chicago

Background: molecular dynamics (MD)

- Simulates the motion of atoms under classical forces: proteins, membranes, materials
- Each time step: compute the force on every atom, then move all atoms forward by $\Delta t \approx 1–4$ fs
- 1 μs of simulated time ≈ 5×10^8 steps at 2 fs, so performance is measured in **ns/day** (simulated time per wall-clock)
- Coulomb forces decay as 1/r and cannot simply be cut off: they need a **long-range solver** (PME or FMM)



repeated 10^6–10^9 times

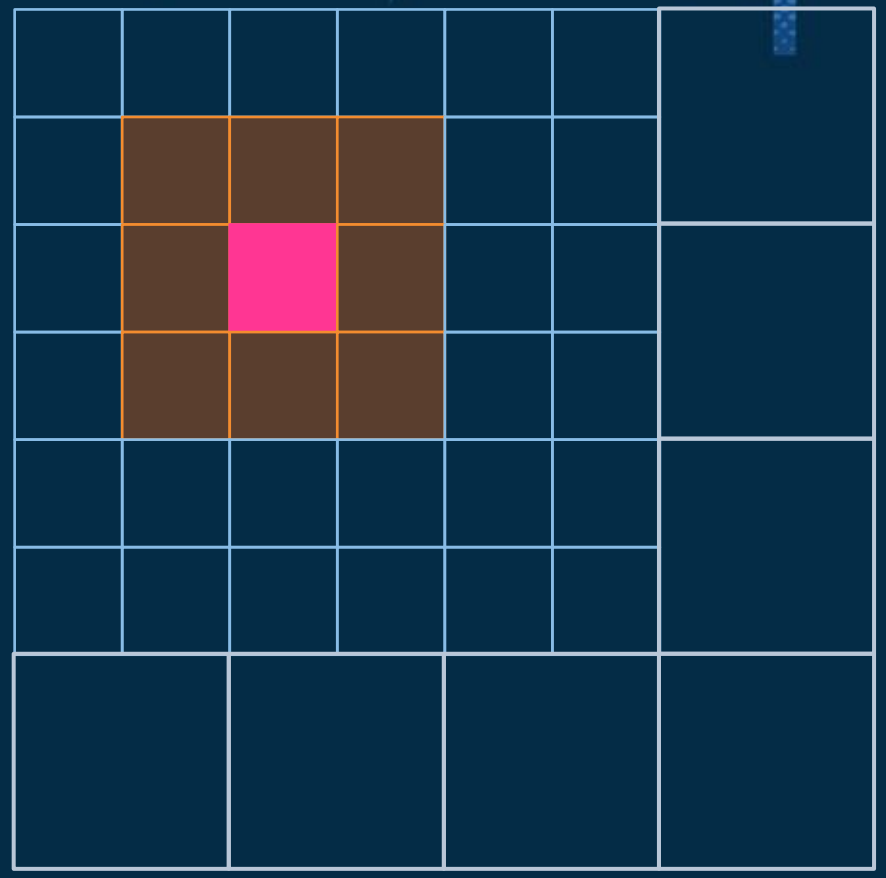
Coulomb (long range) is the hardest part to scale

Long-range electrostatics in GROMACS: PME

- Coulomb interaction split into a short-range part (direct, within cut-off) and a long-range part (reciprocal space)
- Smooth PME: charges spread onto a grid, Poisson equation solved with 3D FFTs, O(N log N)
- A distributed 3D FFT needs all-to-all transposes between ranks
- On hundreds of GPU nodes, this global communication can take longer than the computation
- This limits scaling of multi-million atom systems, e.g. whole-cell models over microseconds



The fast multipole method (FMM) in one slide



2D sketch of the cells seen by one target cell

One FMM evaluation

P2M particles → multipole expansion of each leaf

M2M children → parent, up the tree

M2L multipole → local expansion of a distant cell

L2L parent → children, down the tree

L2P local expansion → force on each atom

- target cell
- near field: direct pair sums
- far field via M2L, leaf level
- far field via M2L, parent level

Expansion order **p**: error falls exponentially with p, cost grows with p. Total cost O(N).

Communication: PME vs. FMM

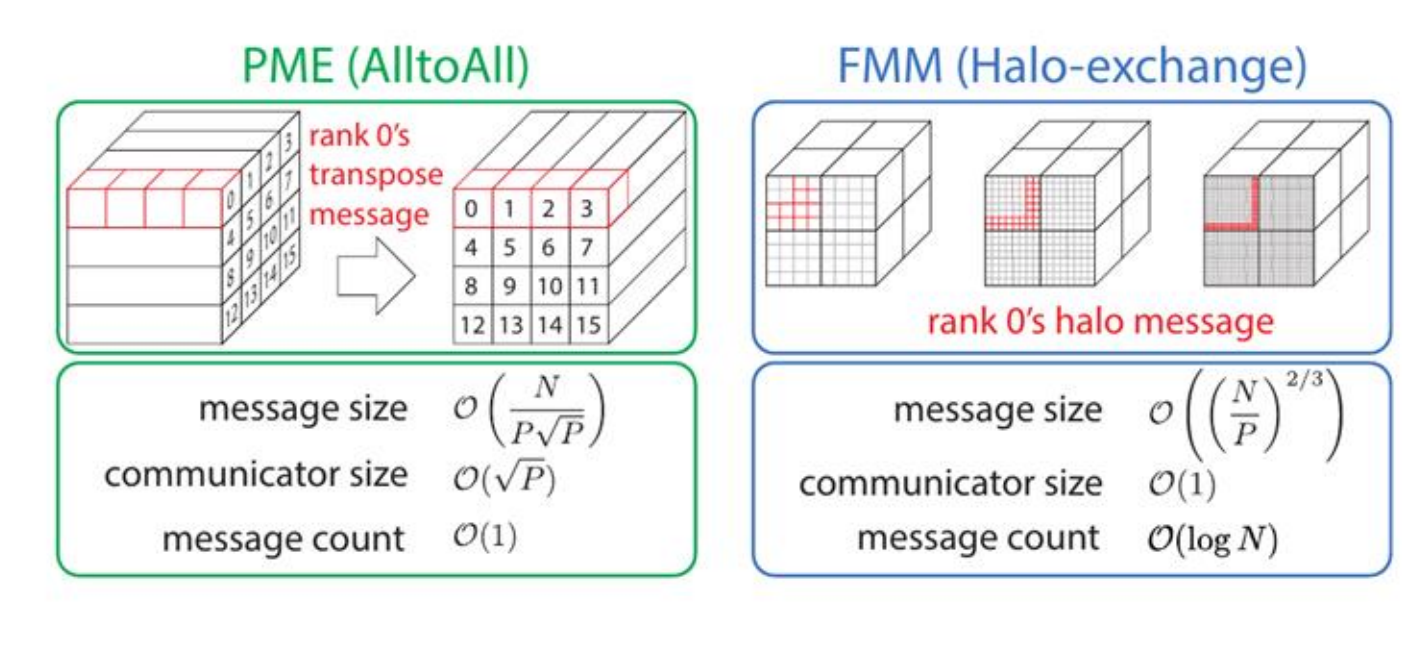


Fig. 1 Communication complexity for N atoms on P processes

- **PME:** all-to-all among O(√P) processes, a global synchronization in every step
- **FMM:** mostly local, hierarchical exchange; farther ranks exchange less data
- **Our implementation:** point-to-point with nearby ranks, plus a small coarse-level Allgather
- In scaling messages get small, so the number of partners and global synchronization matter more than bandwidth

Why revisit FMM now?

What favors FMM today

- M2L, the most expensive far-field operator, maps naturally to GEMM, so it can run on Tensor Cores
- Compute throughput grows faster than network bandwidth
- FMM is O(N) with local, hierarchical communication

What held FMM back

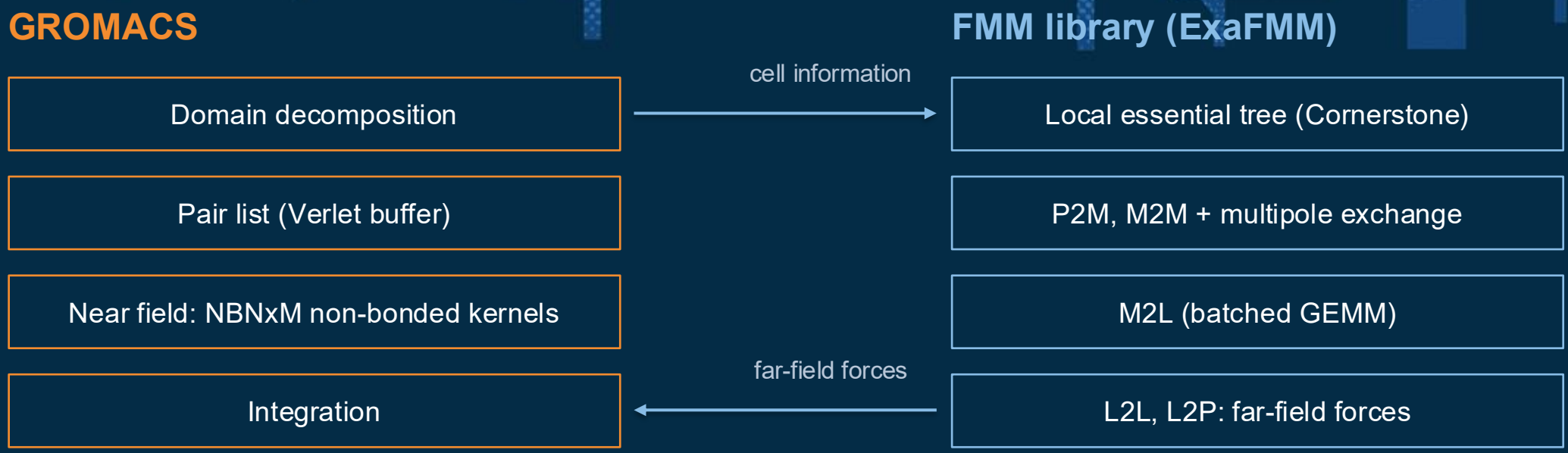
- Larger constant than PME; PME has decades of tuning
- GPU FMM codes rebuild the octree and re-sort particles every step
- With 1 to 4 fs time steps, atoms barely move between steps, so most of this work is redundant
- For small and medium systems the overhead outweighs the O(N) advantage
- **Closest prior work:** GPU FMM in GROMACS (Kohnke et al. 2020, 2021) matched PME accuracy, but was slower than PME

Contributions

1. **GEMM-based M2L** M2L translations batched into dense matrix multiplications
2. **Hybrid near/far field** GROMACS' NBNxM kernels compute the near field, FMM only the far field
3. **GPU-built local essential trees** Cornerstone octree, point-to-point + Allgather exchange
4. **Tree reuse** octree and interaction lists rebuilt only on GROMACS pair-list updates
5. **Asynchronous GPU execution** FMM in its own CUDA stream, overlapping GROMACS kernels

Result: on ALPS (up to 128 nodes, 4× GH200 each), FMM beats GROMACS' PME from 4 nodes on and is up to **3.52× faster** on a 44M-atom system (64 nodes), with energy drift within GROMACS tolerances.

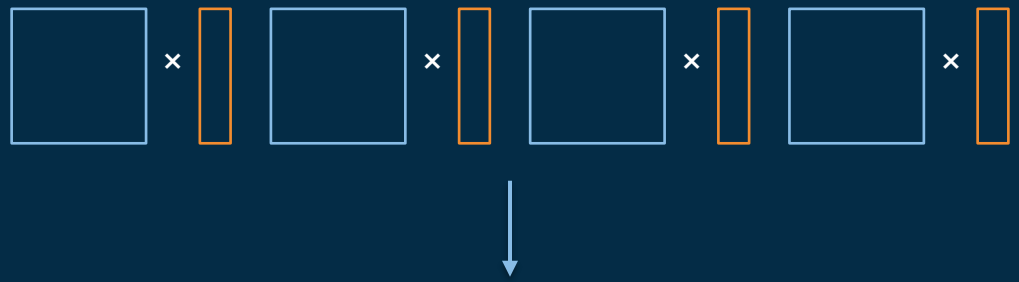
Integration: GROMACS near field, FMM far field



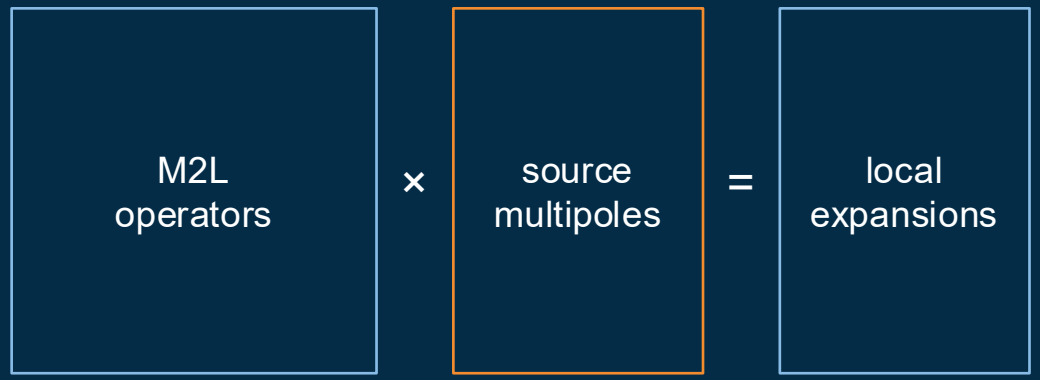
- Same cut-offs, switch functions and Verlet buffer as standard GROMACS
- LJ range must be covered by direct interactions: cell size ≥ 0.6 nm, about 21 atoms per cell, ≥ 900 direct pairs per atom. **The large direct range allows a low order, p = 4 or 5.**

M2L as batched GEMM

Conventional: one operator × vector per cell pair

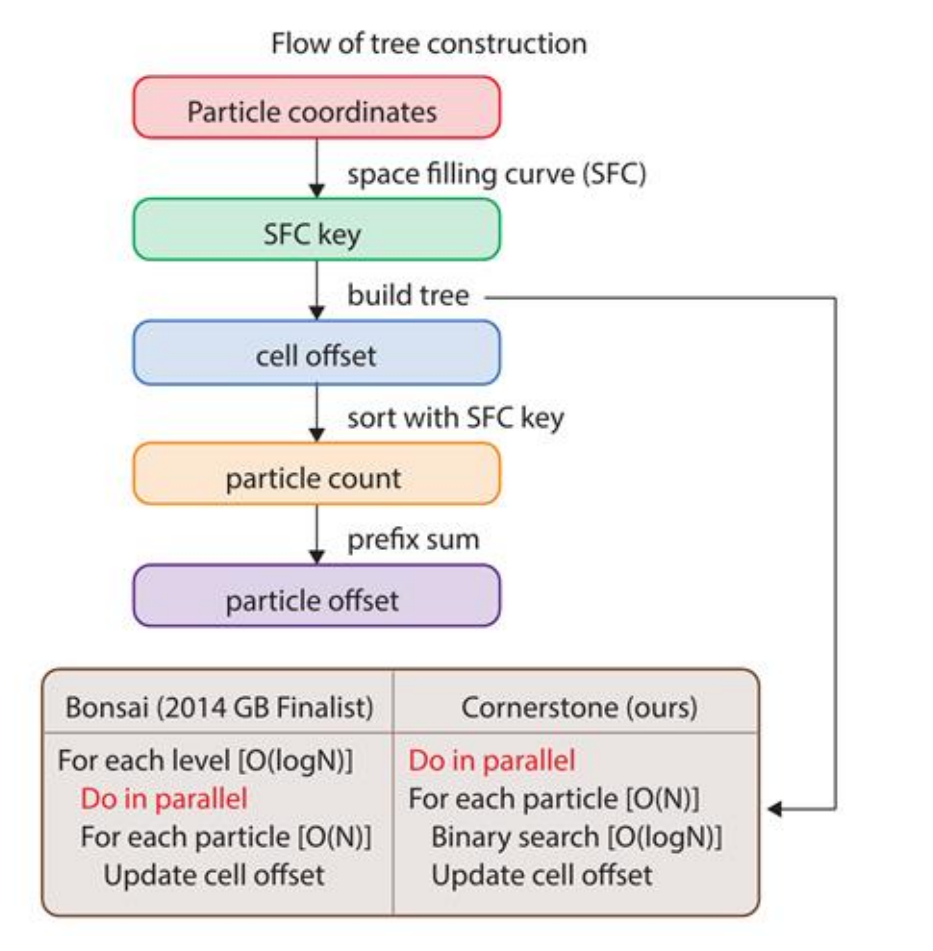


Ours: one large matrix multiplication



- M2L is the most expensive far-field stage
- **Tensor Cores:** GEMM can use matrix units with much higher peak throughput than CUDA cores
- **Memory:** tiling and shared-memory reuse reduce global-memory traffic
- Moves M2L from latency-bound to compute-bound

Octree construction on the GPU (Cornerstone)



- Four steps: SFC keys, build tree (cell offsets), sort by key, prefix sum for particle offsets
- Fully parallel over cells, unlike level-by-level construction in Bonsai
- Tree arrays (keys, child indices, centers) stored per level, ordered by SFC key
- Only parallel-for, radix sort and prefix sum: maps well to GPUs
- Morton (Z-order) curve: cheaper than Hilbert; domain decomposition stays in GROMACS

Multipole communication

Previously: one MPI_Alltoallv, irregular pattern and high run-time overhead. Now two stages:

Nearby ranks: point-to-point

- Halo-like exchange of partial octrees with nearby ranks
- Pattern repeats for hundreds of steps: persistent requests (MPI_Send_init / MPI_Recv_init, MPI_Startall / MPI_Waitall)
- **20–30% faster** than MPI_Isend / MPI_Irecv

Distant ranks: MPI_Allgather

- On the regular domain-decomposition grid and above
- Only **1 to 4 cells per rank**
- Not communication-optimal, but vendor-optimized collectives (often switch-accelerated) beat a hand-written hierarchy
- Enabled from 64 ranks; below that, point-to-point covers the whole domain

Tree reuse synchronized with the pair list



full update every nstlist steps (nstlist = 100 in our runs), operator-only steps in between

Full update step

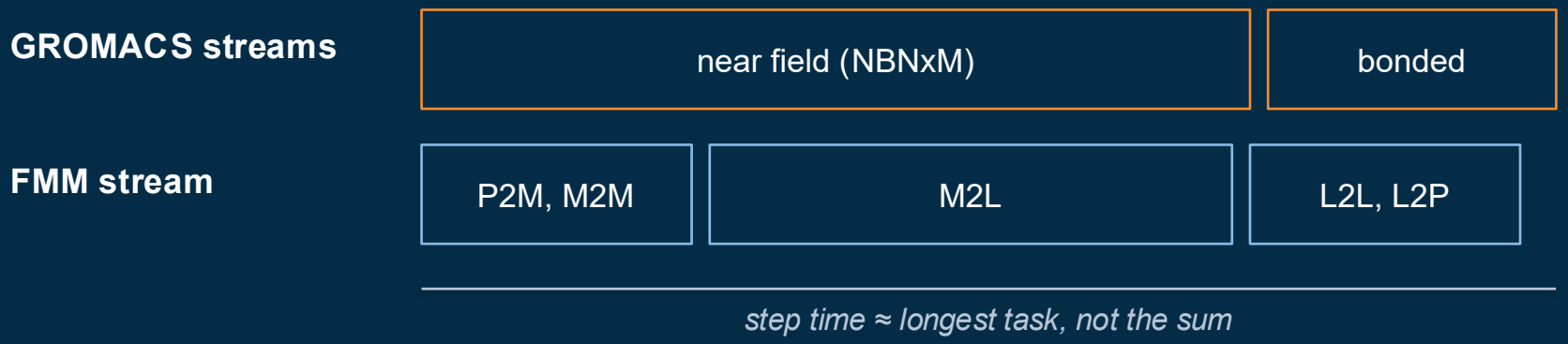
- Particles sorted by SFC key, cell membership determined
- Target tree updated from the static source tree
- New M2L interaction lists

Operator-only steps (nstlist − 1)

- Tree and interaction lists reused
- Only coordinates change
- Run P2M, M2M, M2L, L2L, L2P

Asynchronous, GPU-resident execution

One MD step on one GPU (schematic)



- Dedicated non-blocking CUDA stream, launched as soon as multipoles are ready
- Octree built on the GPU; expansions and interaction lists stay in device memory
- **Current limitation:** the GROMACS–FMM interface still passes coordinates through the CPU, and FMM cannot overlap GROMACS' halo exchange

Experimental setup

System	Atoms	Used for
Random atoms	24,000	FMM accuracy vs. direct sum
Pure Coulomb water	24,000	Forces vs. PME
Grappa (ethanol–water)	768,000	Accuracy, energy drift, scaling
Water box	44,000,000	Large-scale scaling

Machine: ALPS (CSCS)

- 1 to 128 nodes
- 4× NVIDIA GH200 per node, one GPU per rank
- NVLink within a node, Slingshot 11 between nodes

Comparison settings

- PME: GROMACS with cuFFTMp (distributed GPU FFT), 1/4 of the ranks dedicated to PME
- Coordinate update on the CPU for both PME and FMM (current FMM interface limitation)
- FMM: order p = 4, M2L on Tensor Cores, tree rebuilt with the pair list (nstlist = 100)

Accuracy

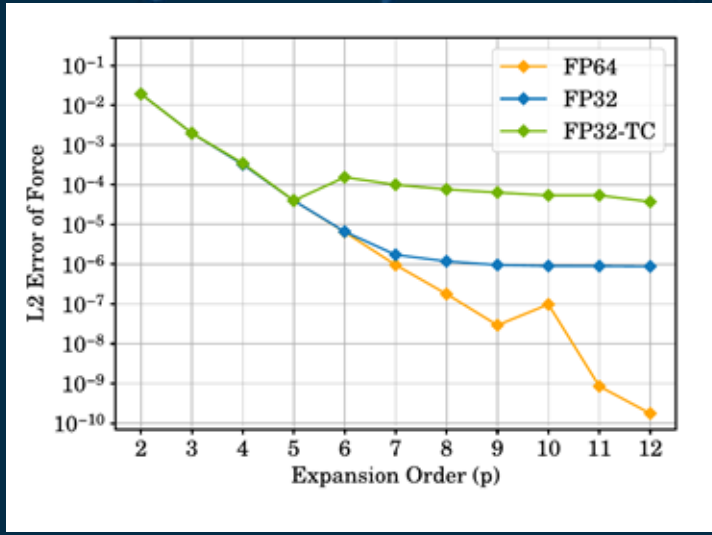


Fig. 4 Force error vs. direct sum

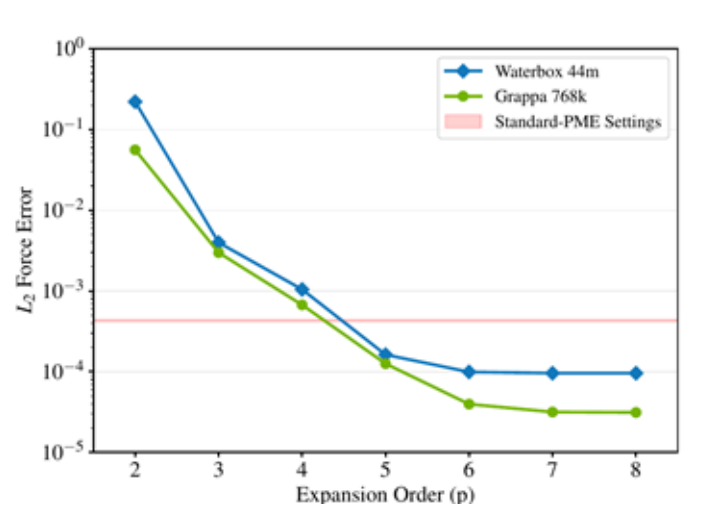


Fig. 5 Force error vs. accurate PME

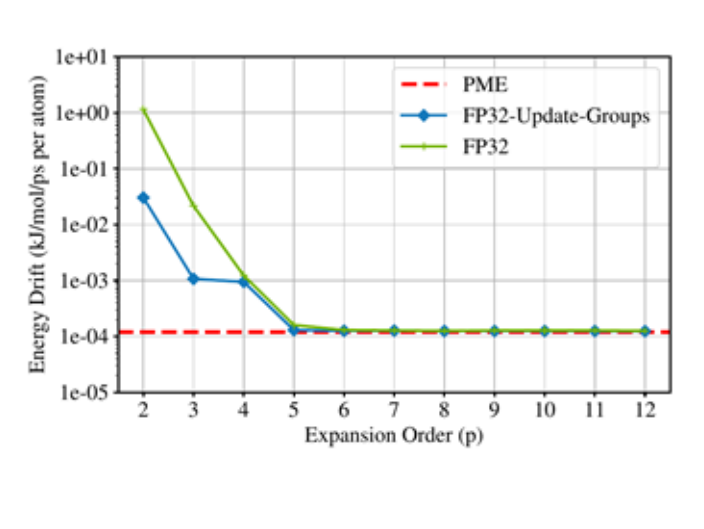


Fig. 6 Energy drift (Grappa 768k)

- **Forces:** exponential convergence in p; from **p = 5** more accurate than PME with standard settings
- **Energy drift:** within the GROMACS default tolerance from p = 4 (tolerance: 5e-03)

Scaling: 44M-atom water box

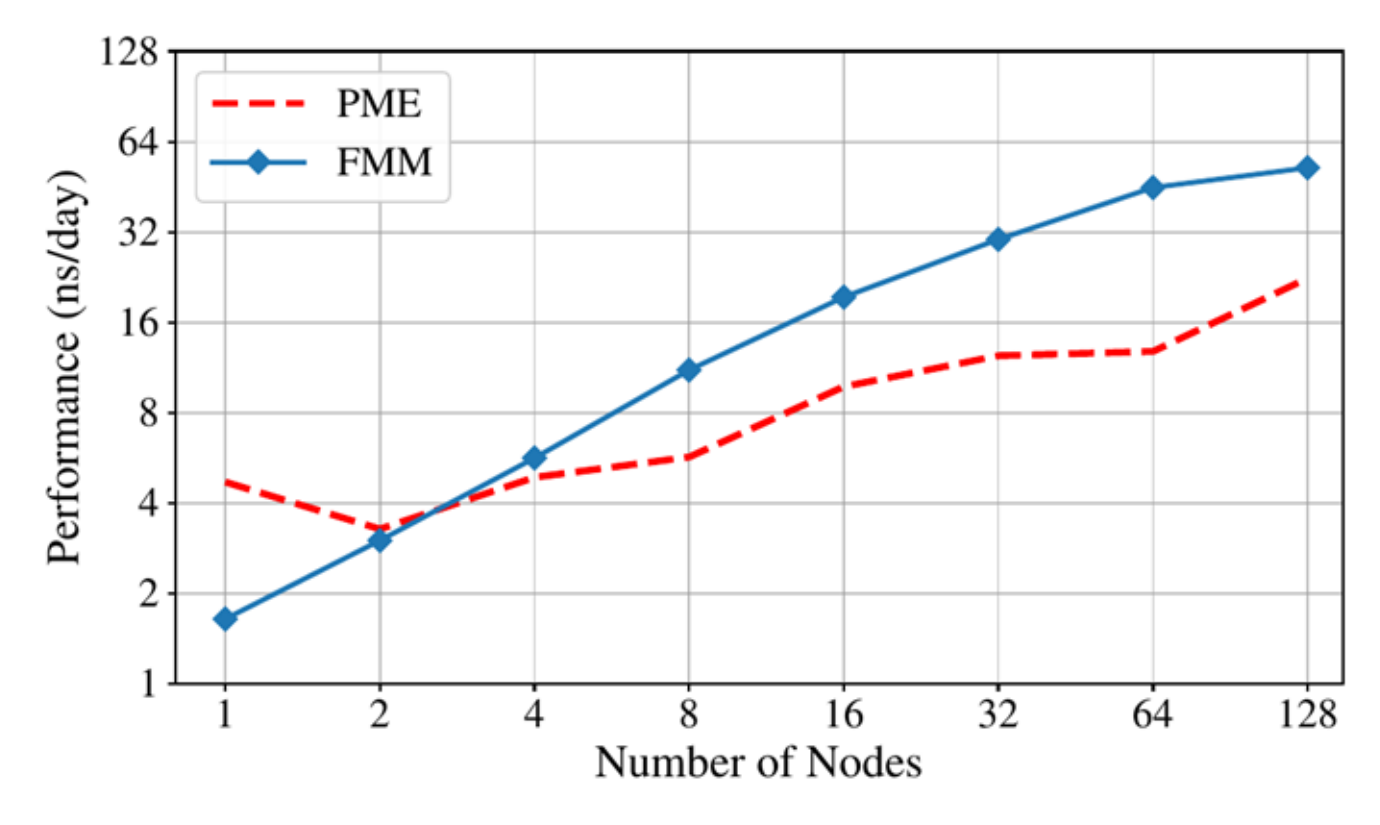


Fig. 8 ns/day vs. nodes, multipole order 4

- Crossover at 4 nodes: **5.181 vs. 4.872 ns/day**
- PME gains little between 16 and 64 nodes
- Largest gap at 64 nodes: **3.52×** (~45 vs. ~13 ns/day)
- At 128 nodes PME recovers, gap ~2.4×; FMM still highest overall (~52 ns/day)

Scaling: Grappa 768k

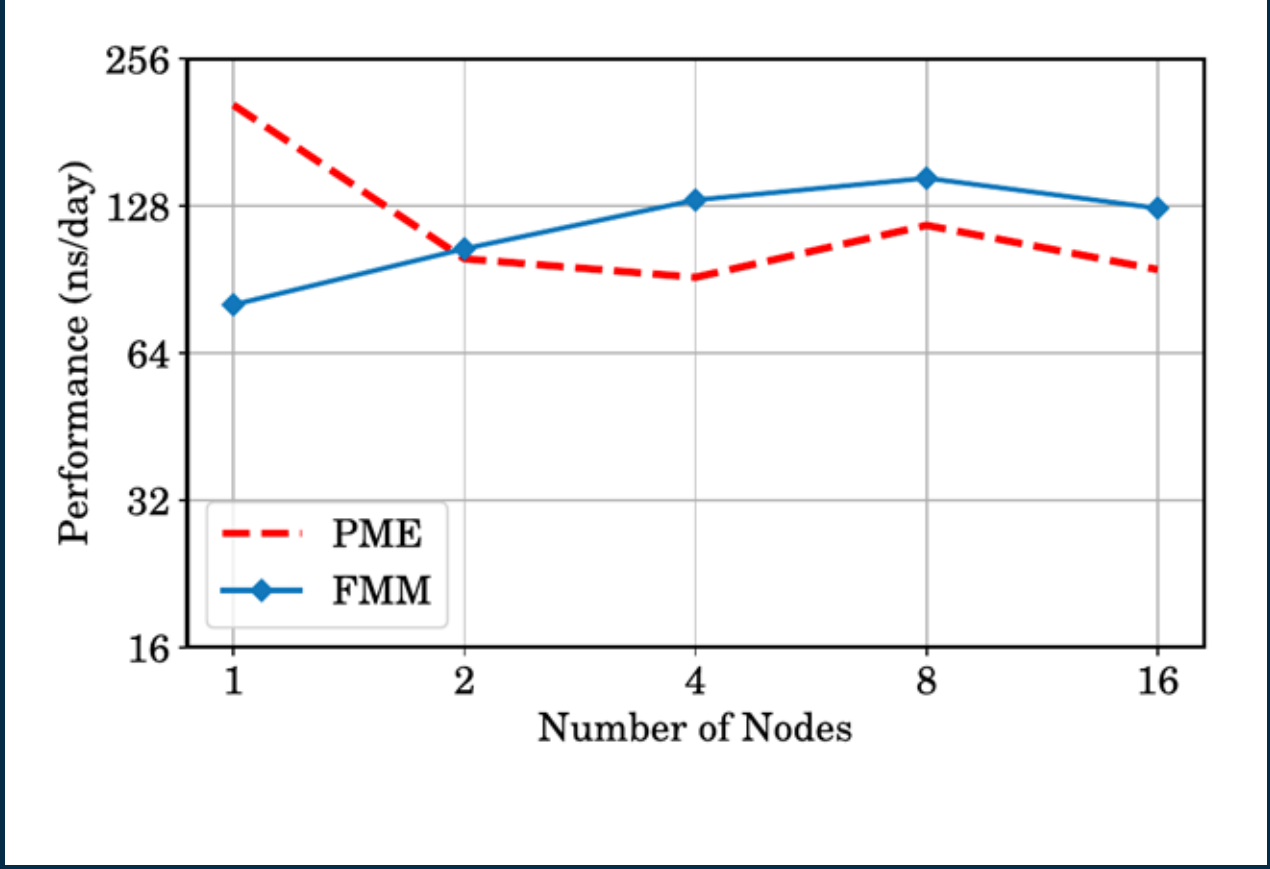


Fig. 9 ns/day vs. nodes, multipole order 4

- **Fastest overall: PME on 1 node** (~200 ns/day); FMM peaks at ~145 ns/day on 8 nodes
- At equal node count, FMM is ahead from 2 nodes on
- Visible step from NVLink (intra-node) to Slingshot (inter-node)
- Crossover depends on system size and interconnect

Limitations and future work

Current limitations

- Coordinates go through the CPU before reaching the FMM on the GPU
- Generic MDModule interface: no overlap of multipole work with the coordinate/force halo exchange
- LJ interactions must lie within the FMM direct range

Next steps

- Fully GPU-resident coordinates and forces
- Overlap multipole work with halo exchange
- Lower the crossover for small systems
- Wider validation on biomolecular systems

All reported speedups are with these limitations in place.

SC26

Summary

- GPU FMM integrated into GROMACS as an alternative to PME for long-range electrostatics
  - GEMM-based M2L, GROMACS near-field kernels, GPU-built LETs, hybrid MPI, tree reuse with the pair list
- Accuracy comparable to PME at p = 4–5; energy drift within GROMACS tolerance from p = 4
- Faster than PME from 4 nodes on, up to **3.52×** on a 44M-atom system (64 ALPS nodes)
- 768k atoms: FMM ahead at equal node count from 2 nodes, but single-node PME remains the fastest configuration

Thank you. Questions?



SC26 hpc unites | Chicago