GROMACS: meeting exascale portability and performance challenges

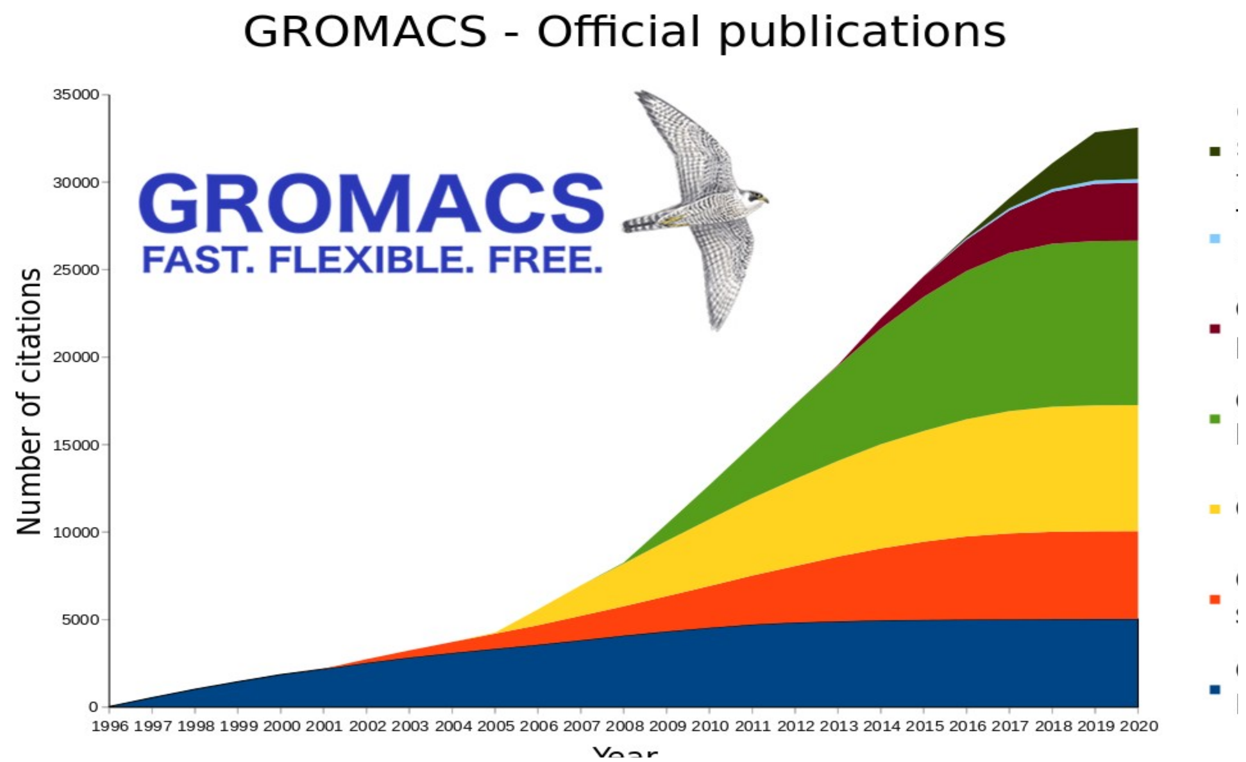
Szilárd Páll
pszilard@kth.se

ISC Workshop on Readiness of HPC Extreme-scale Applications
May 16, 2024

FAST. FLEXIBLE. FREE.
GROMACS

- **Classical MD** package
- **Large user base**: One of the top HPC codes deployed on most clusters
- **Open source**: LGPLv2
- **Open development:** code review & bug-tracker: https://gitlab.com/gromacs
  - modern dev workflow (mandatory code review for >12 years, tiered CI verification)
- **Codebase:** ~1M LOC, C++17

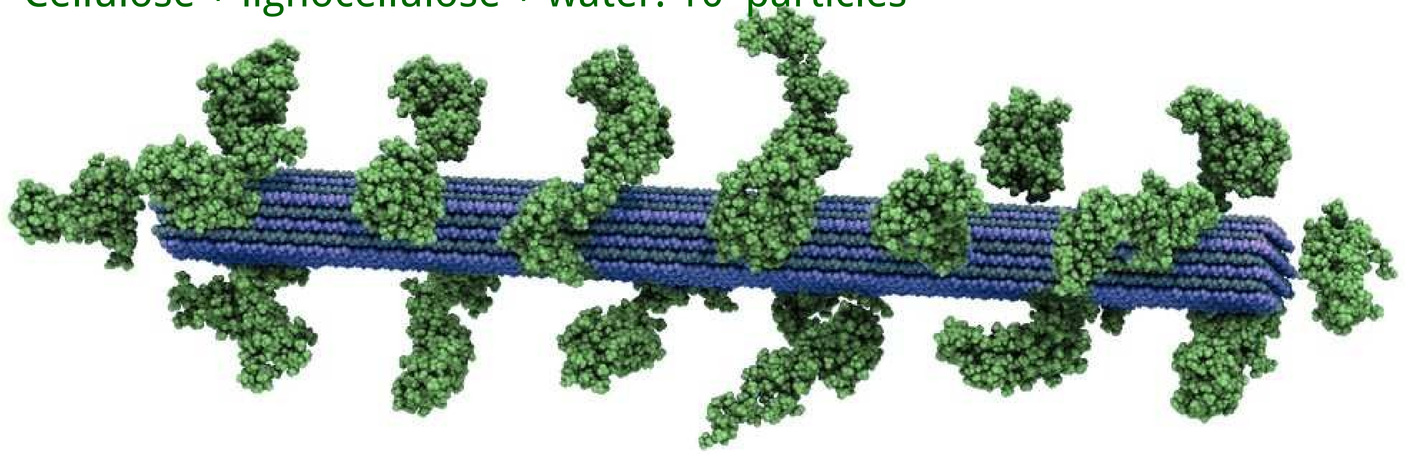
- **Focus on high performance:**
  - efficient algorithms & highly-tuned parallel code
  - bottom-up performance oriented design
- **Focus on portability:**
  - portable programming models
  - SIMD and GPU portability layers



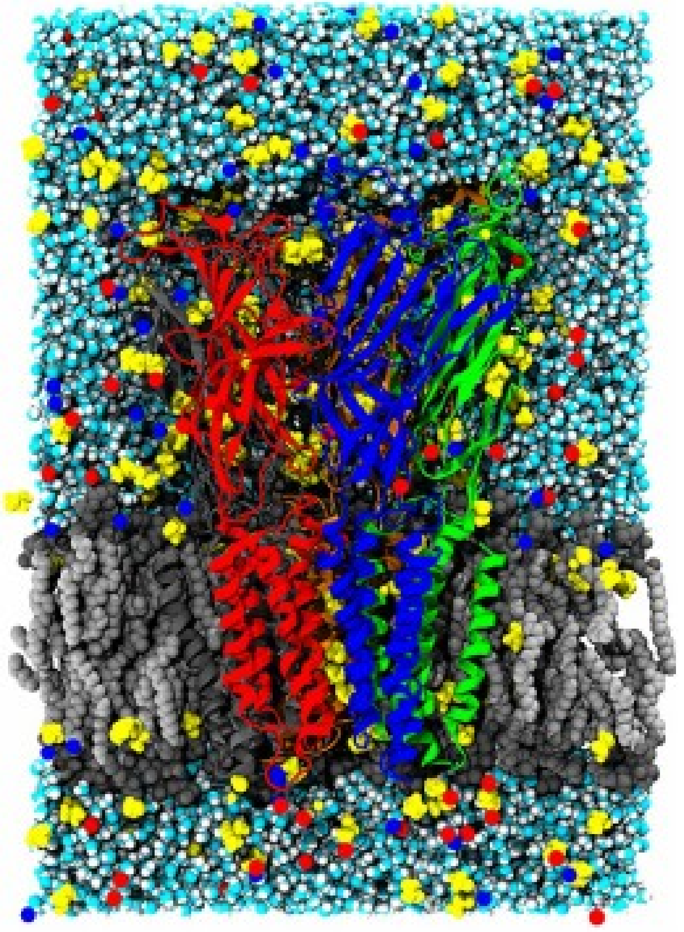
Molecular simulation: use-cases

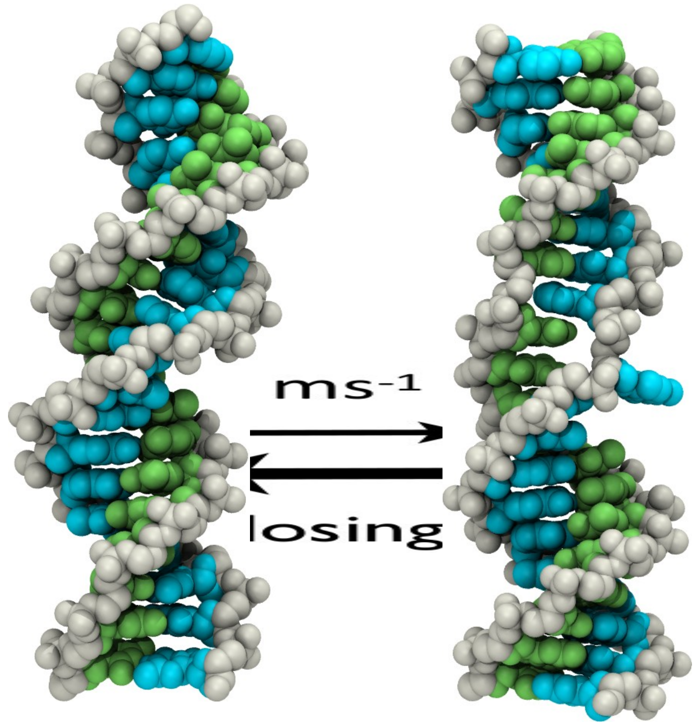
Bio-molecular MD

Cellulose + lignocellulose + water: 10^7 particles



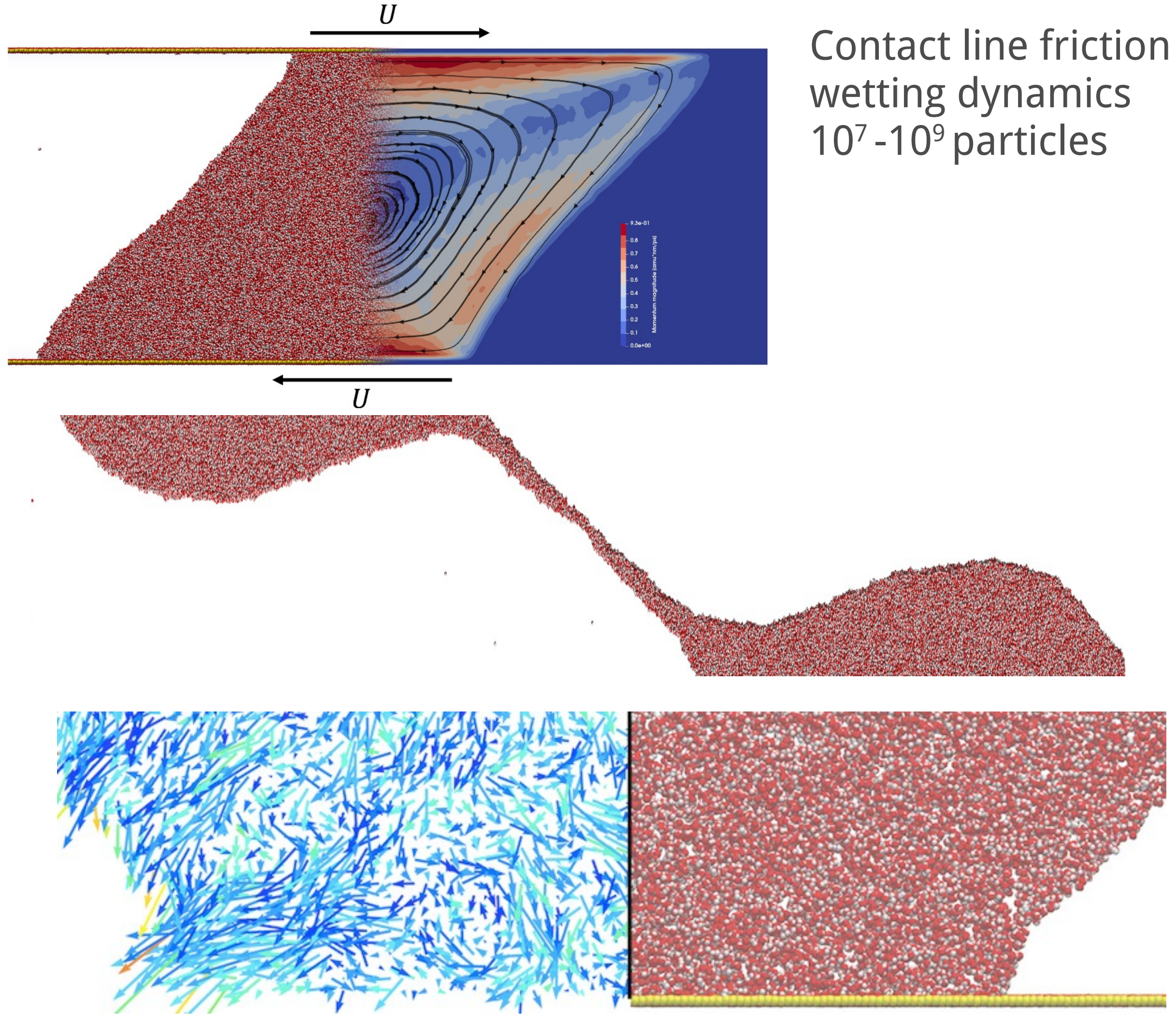
Membrane protein: 10^5 particles





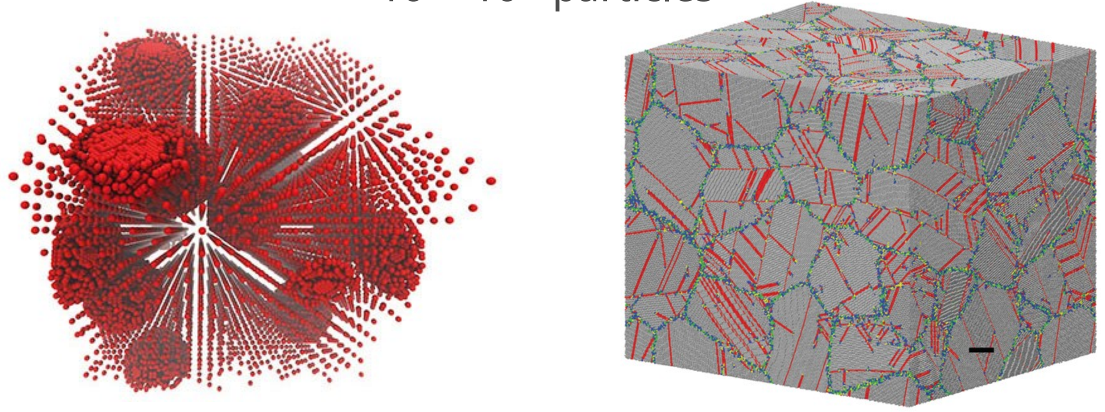
DNA base-pair opening: 10^4 particles

Materials MD



Contact line friction & wetting dynamics
10^7-10^9 particles

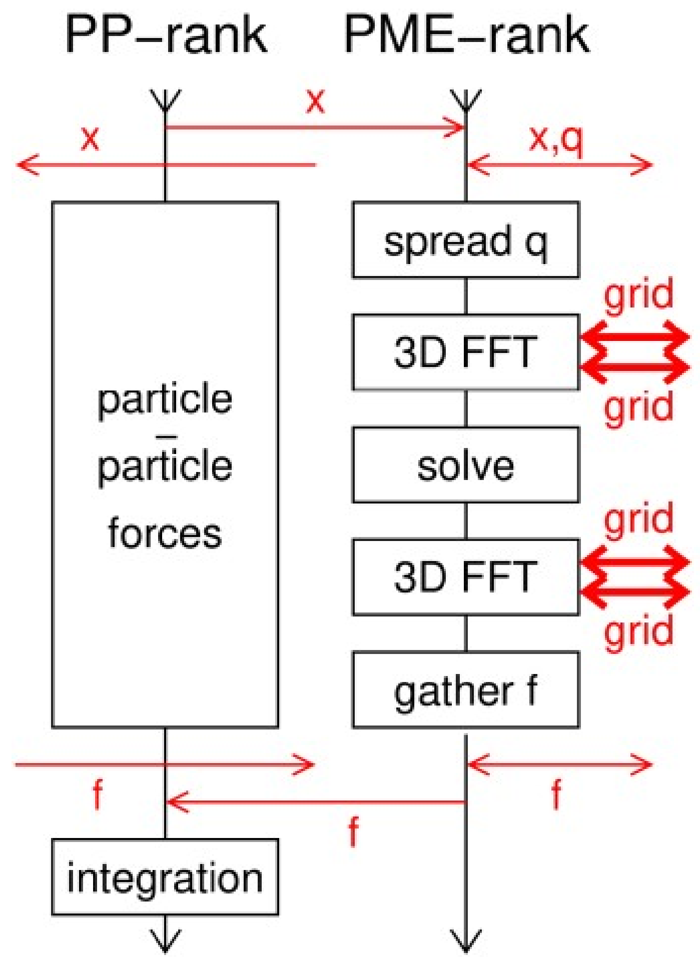
Nucleation in nano-crystals:
10^{10}-10^{12} particles

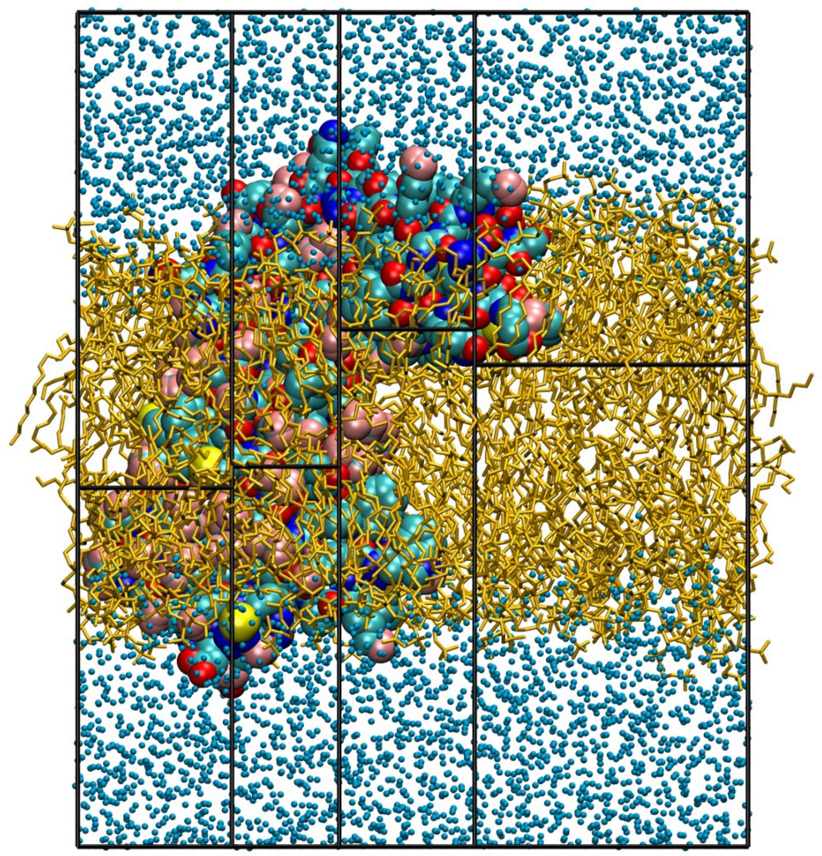


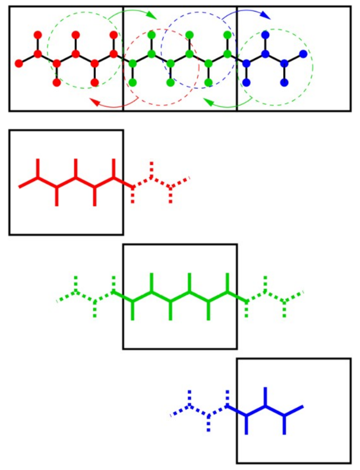
GROMACS parallelization overview

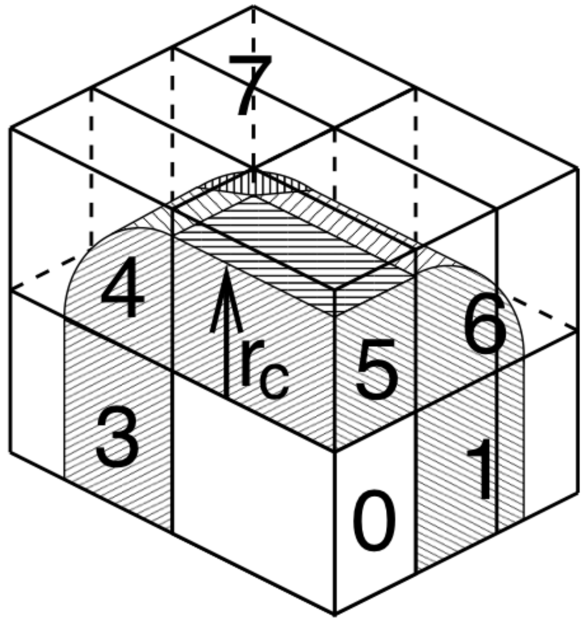
- Multi-level parallelism:
  - SIMD / threading / NUMA / async offload / MPI

- Hierarchical parallelization: target each level of hardware parallelism
  - MPI: SPMD / MPMD; thread-MPI
  - OpenMP multithreading + locality optimizations
  - CUDA, OpenCL, SYCL (through GPU abstraction layer)
  - SIMD: 14 flavors (SIMD library / abstraction layer



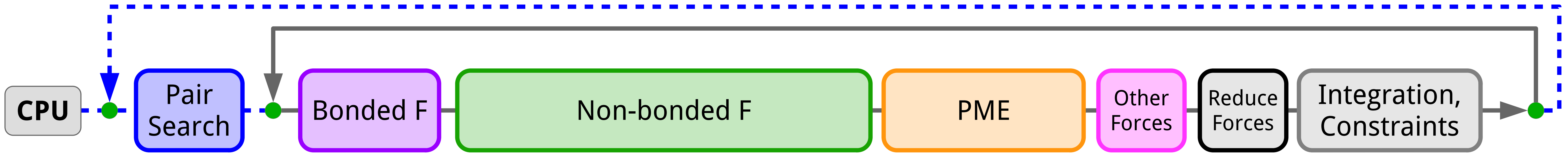






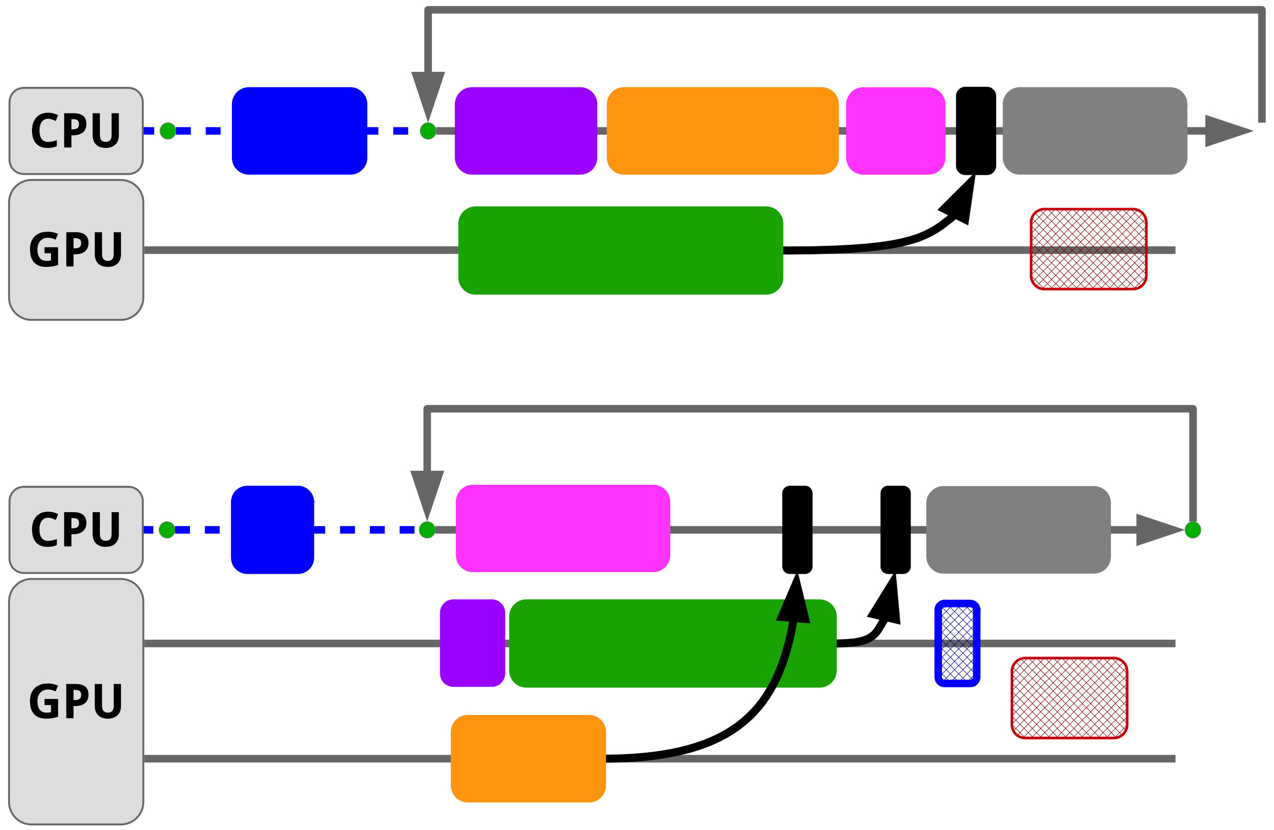
GROMACS on GPUs: embracing heterogeneity

Homogeneous scheme

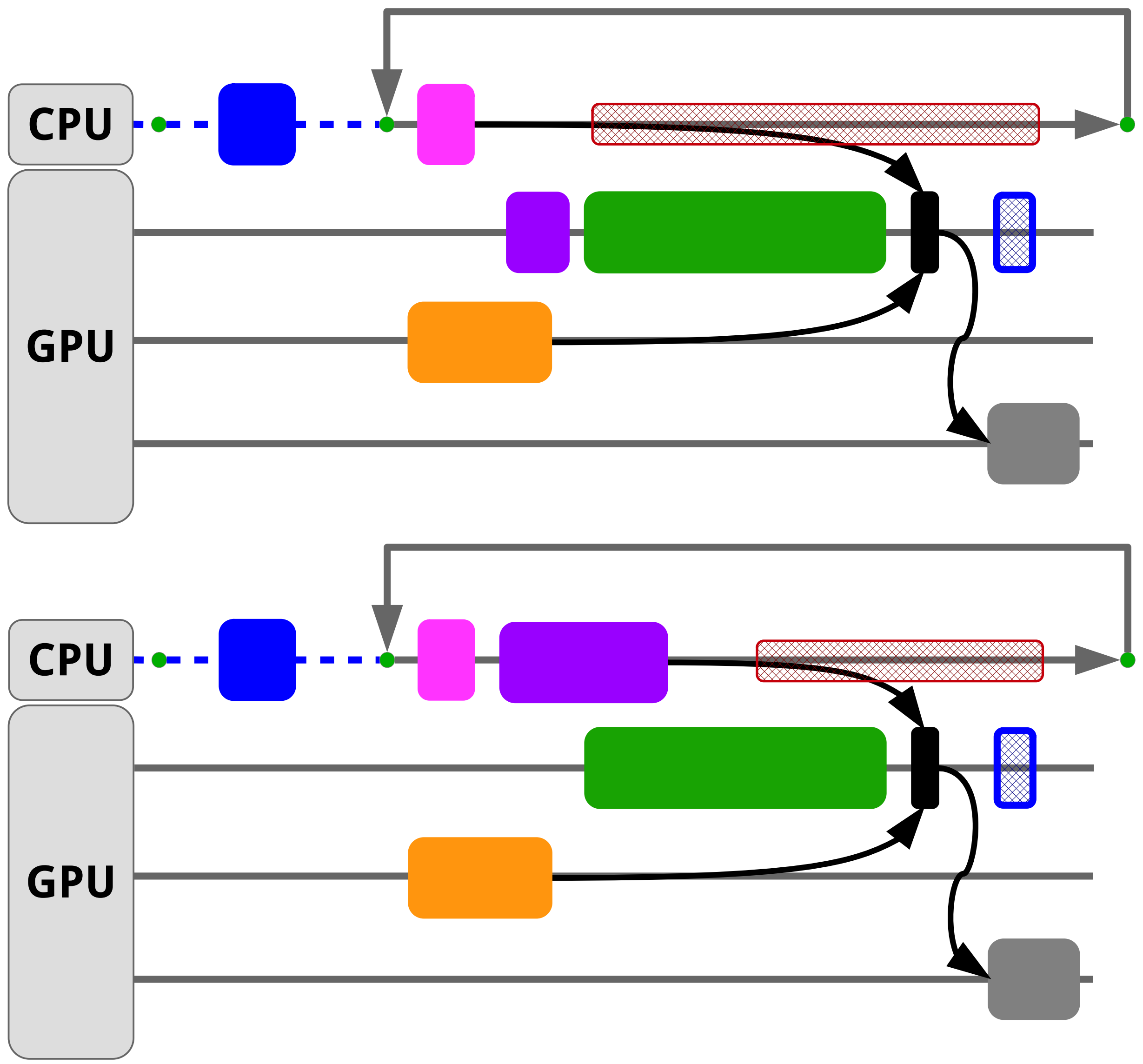


Heterogeneous schemes

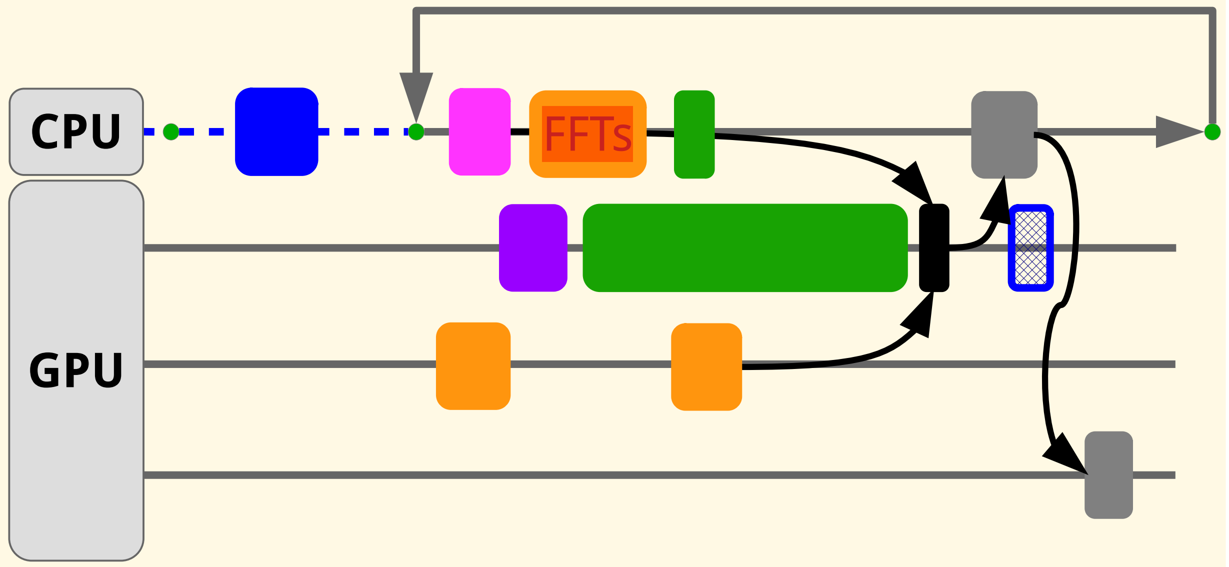
Force offload parallelization



GPU-resident parallelization

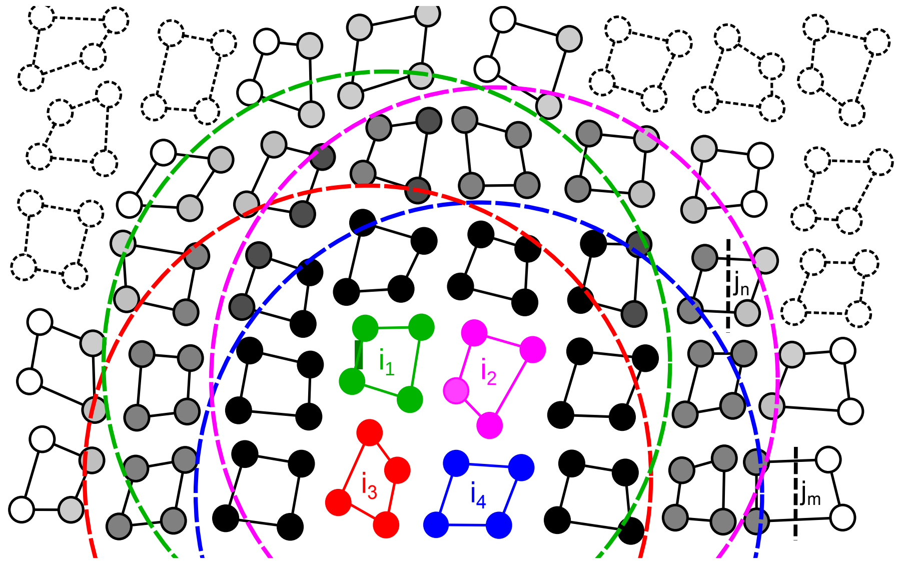


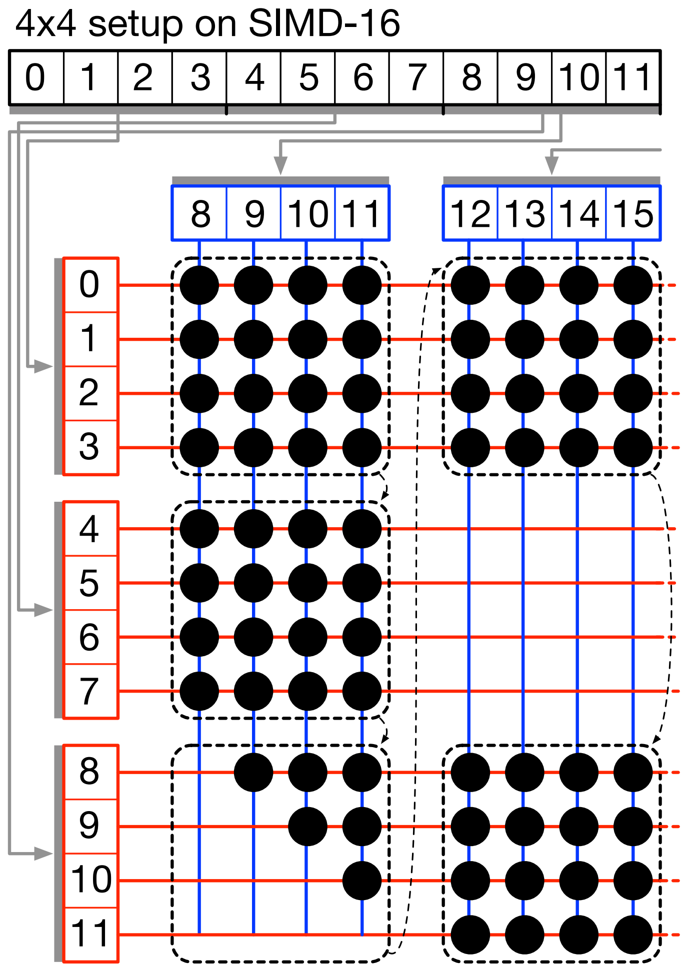
Future: back to partial offload? (APUs)



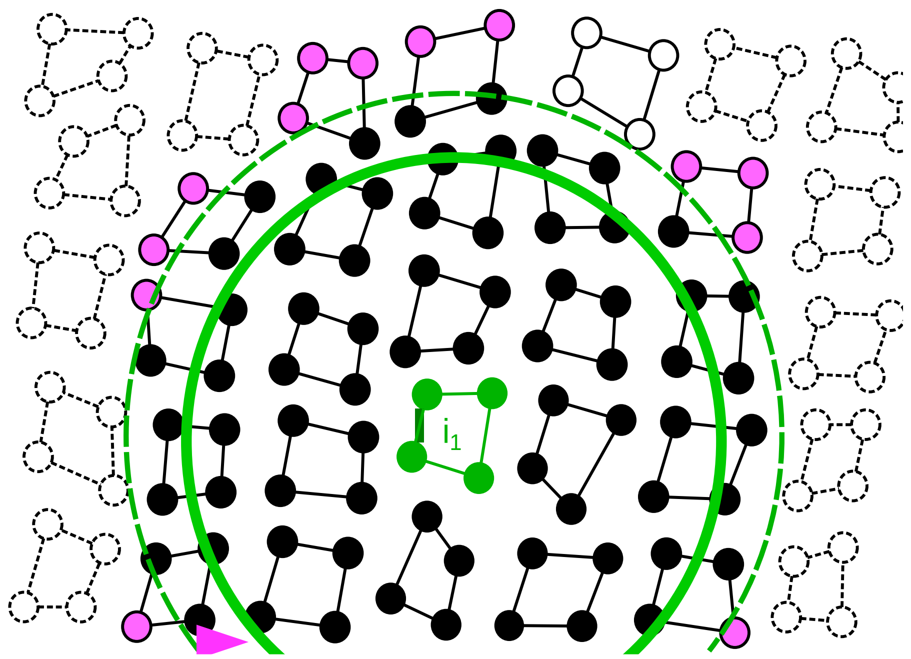
Long-term readiness efforts: algorithm redesign for modern architectures

Cluster pair-interaction algorithm for SIMD/SIMT

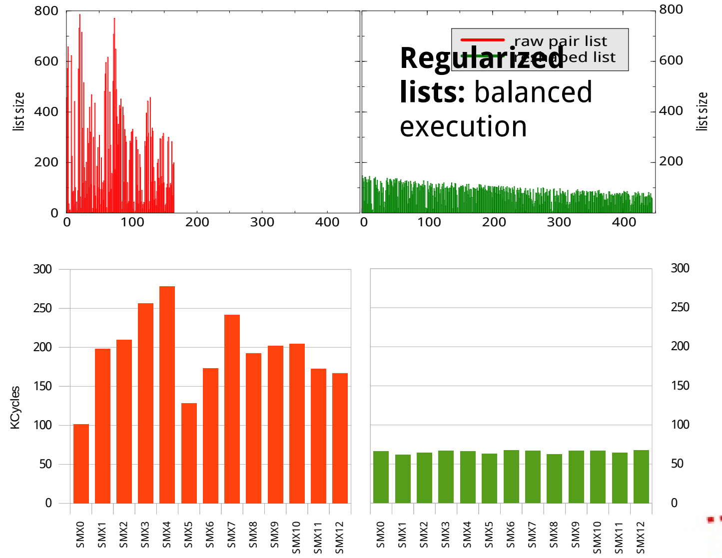


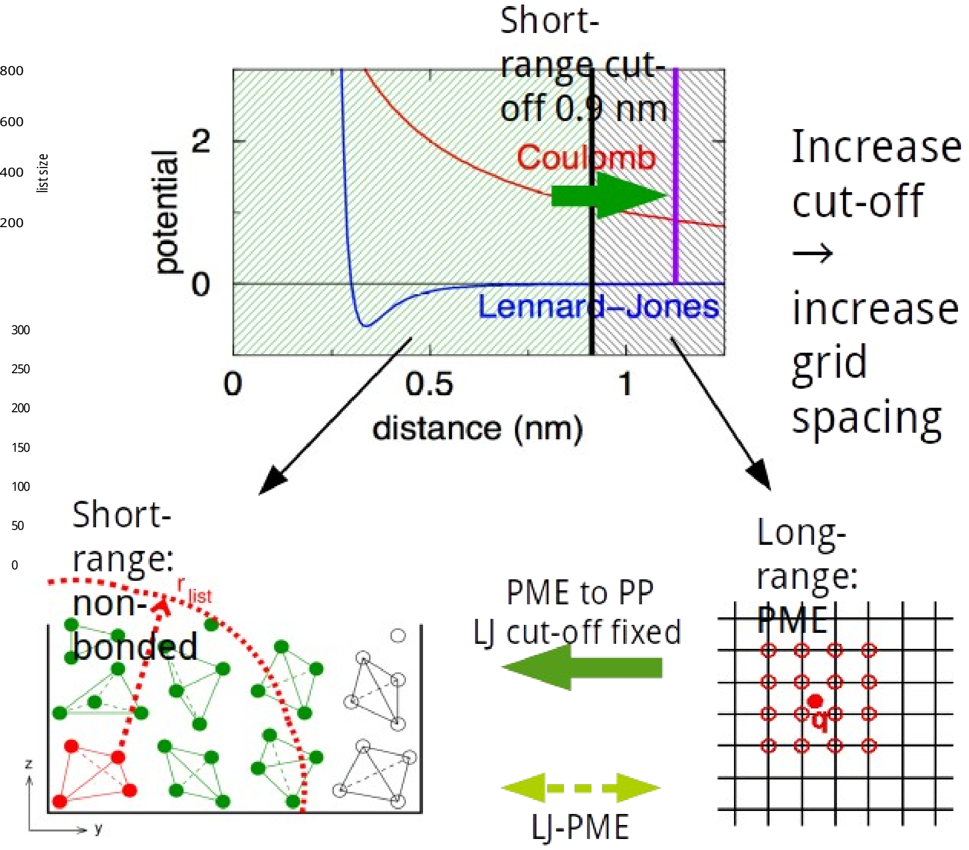


Accuracy-based automated list buffer improves SIMD algorithm parallel efficiency

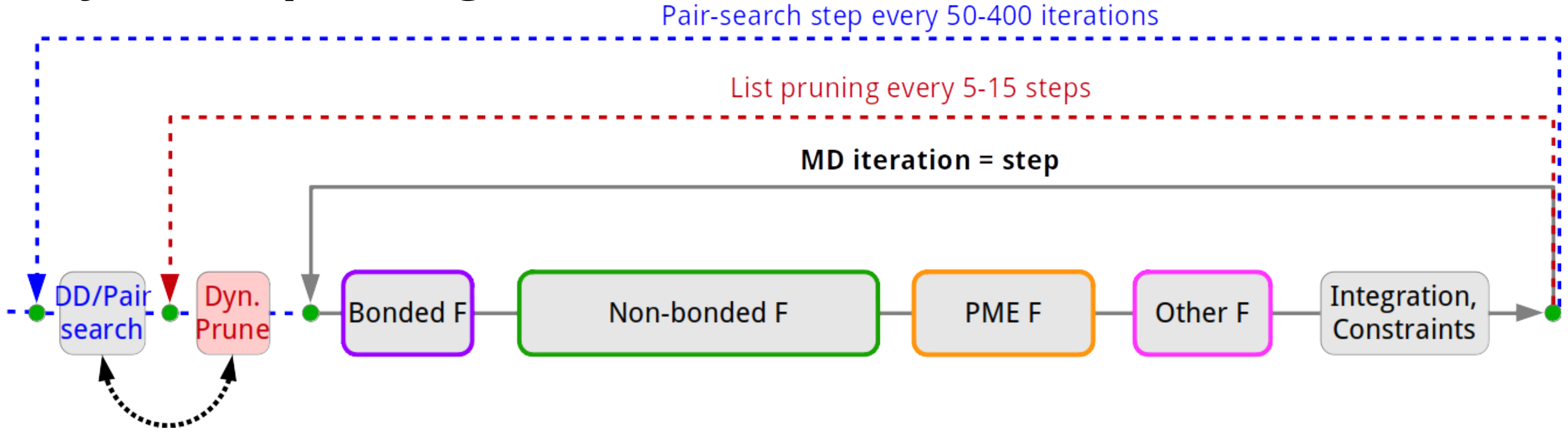


Multi-level heterogeneous data and task load-balancing: intra-GPU, intra-node, inter-node





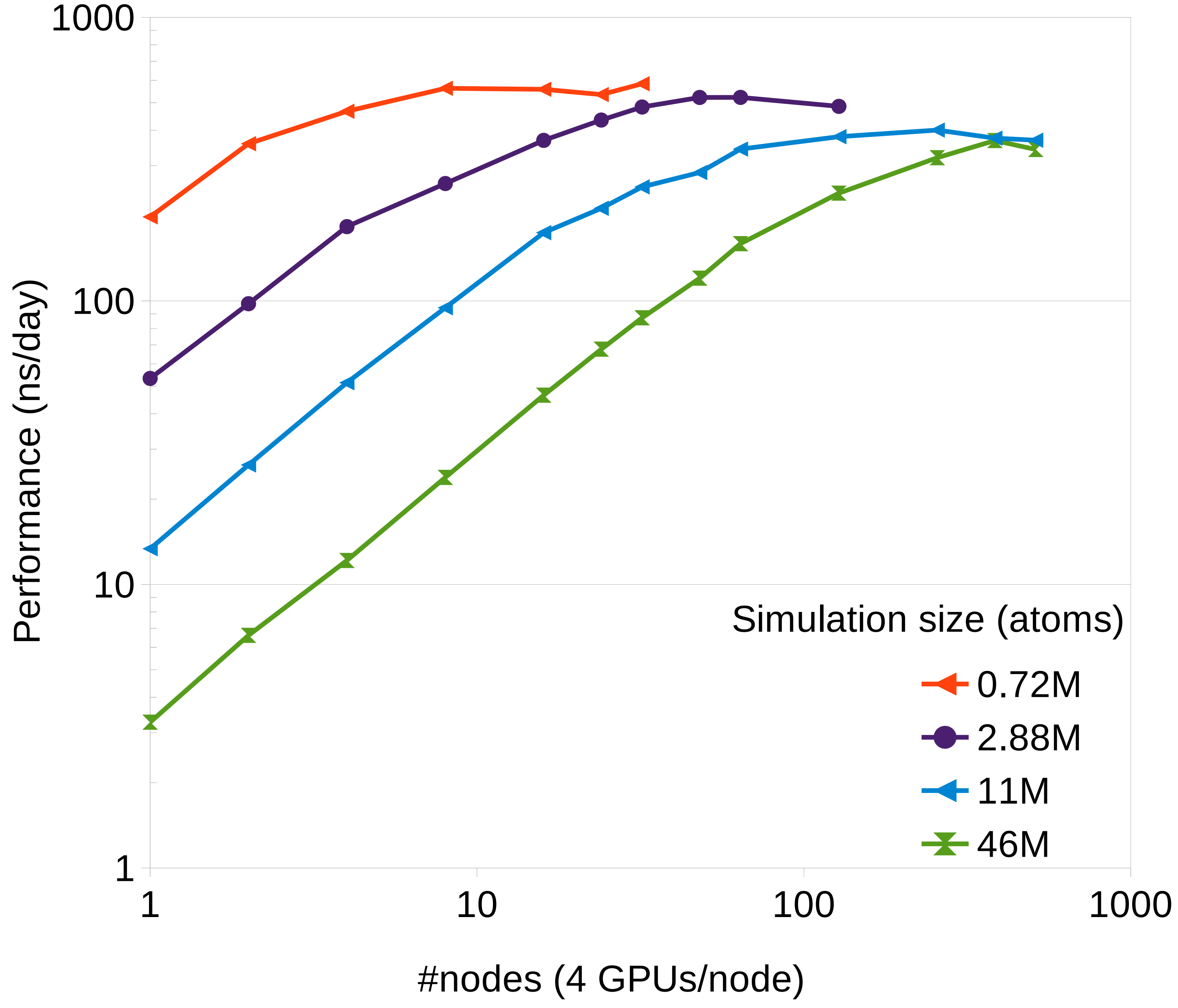
Dual pair list with dynamic pruning



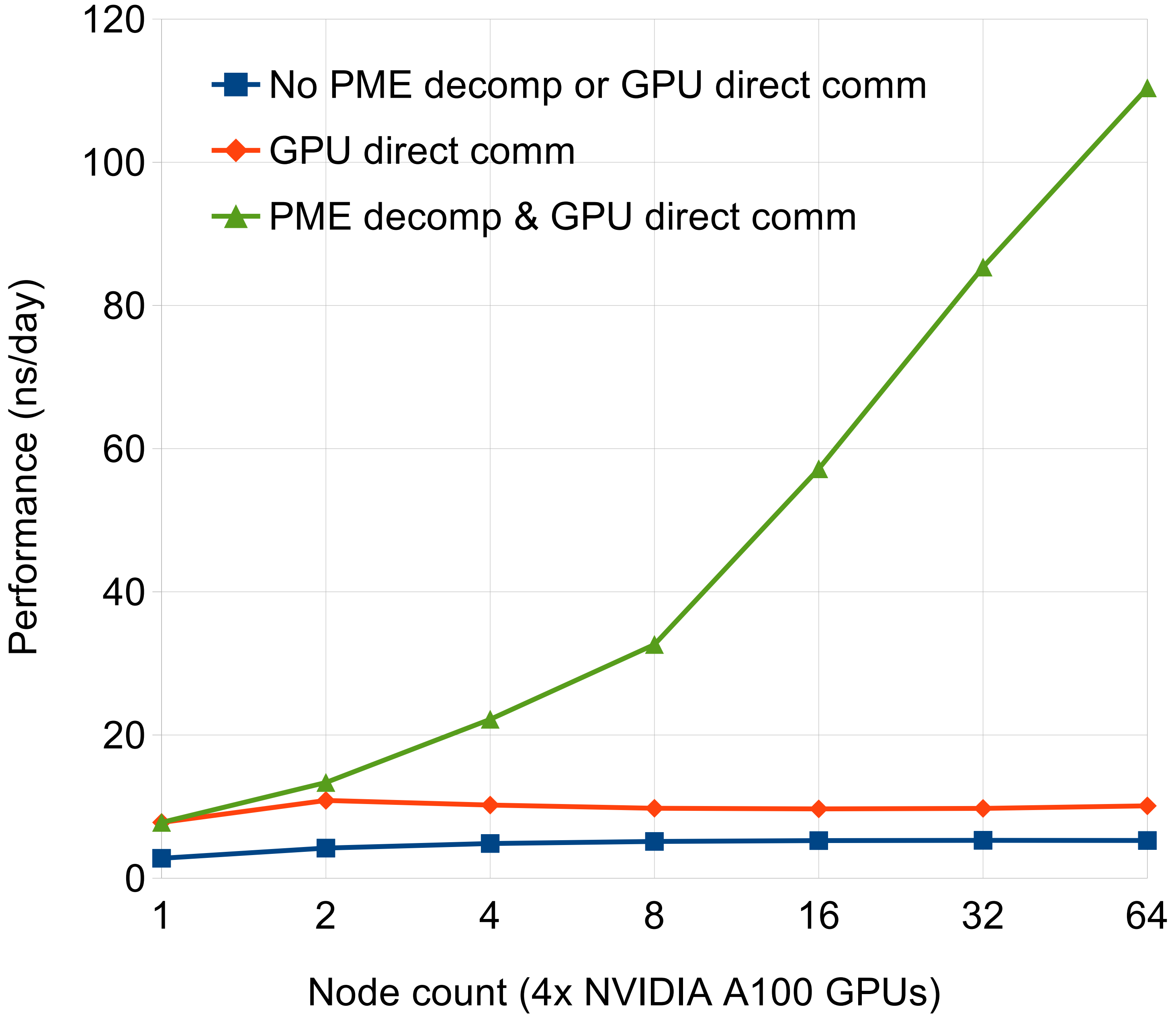
Long-term readiness efforts: algorithm redesign for modern architectures (cont)

Direct GPU communication with proven strong scaling

Domain decomposition strong scaling: ethanol 0.72-46M atoms

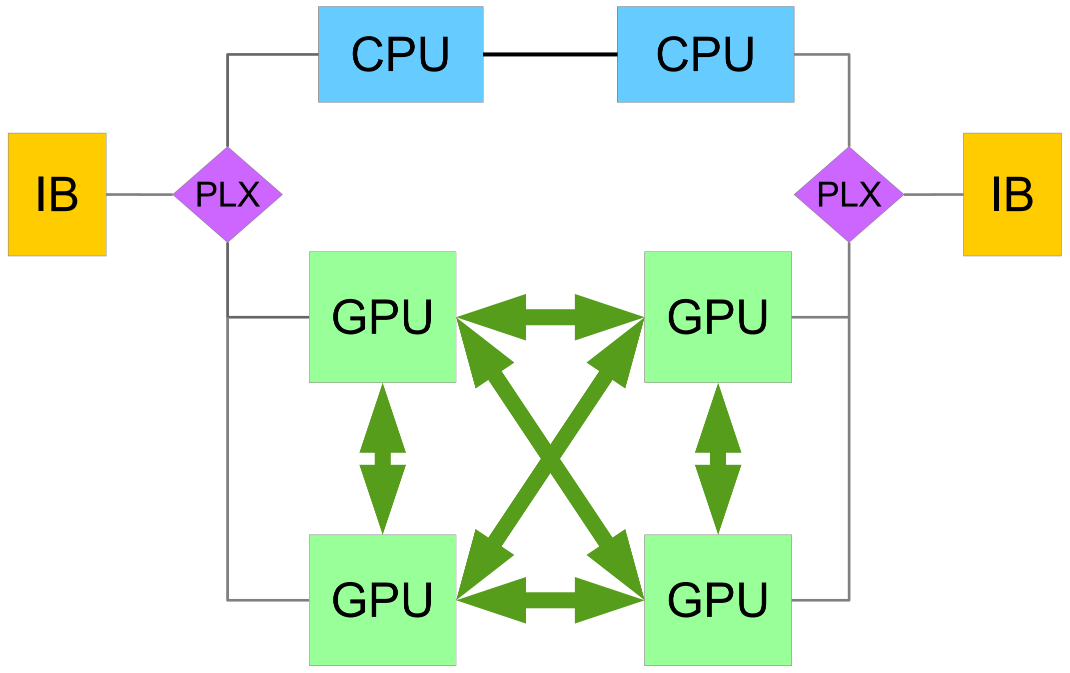


Strong scaling with PME and cuFFTmp: benchPEP-h 12M atoms

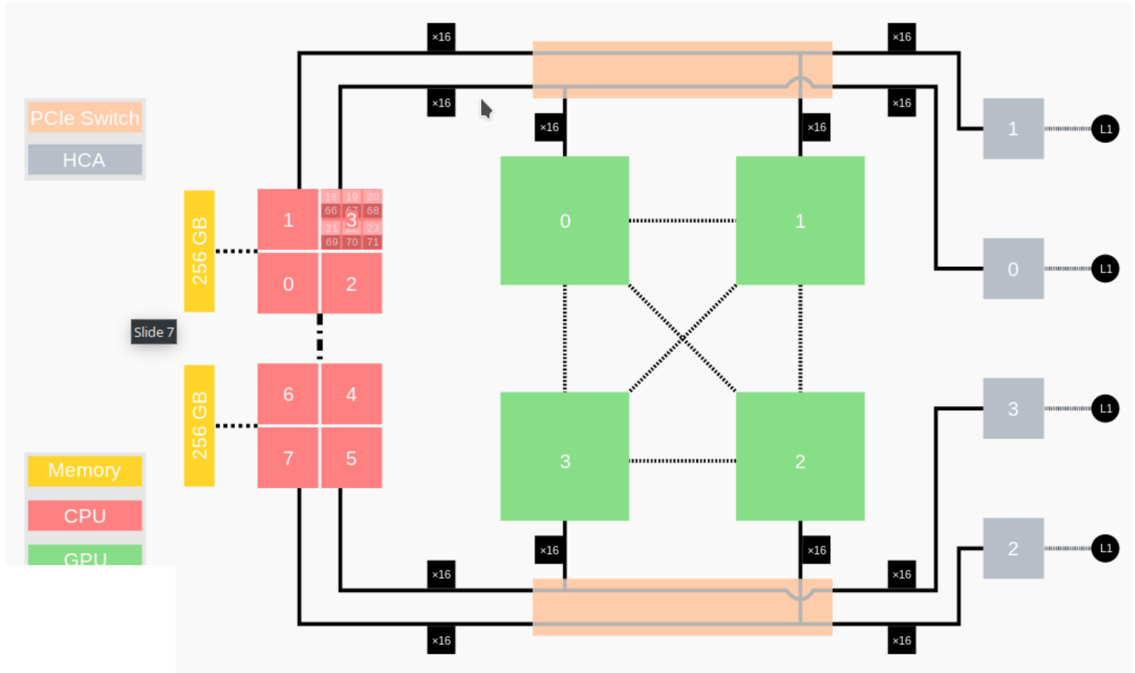


Portable programming models needed!

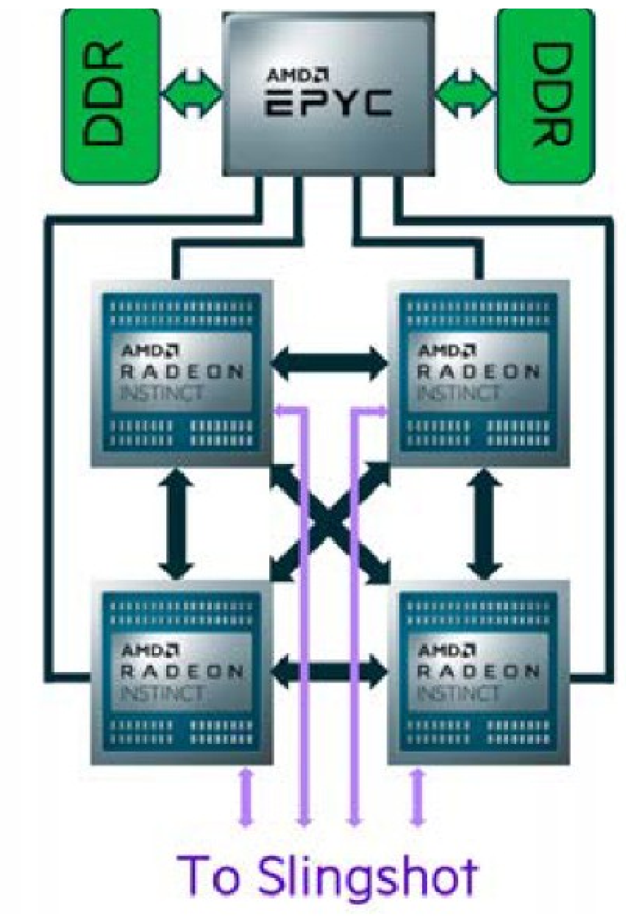
CSC Puhti: 2 Intel CPU + 4 NVIDIA GPU+ NVlink, 2 NIC



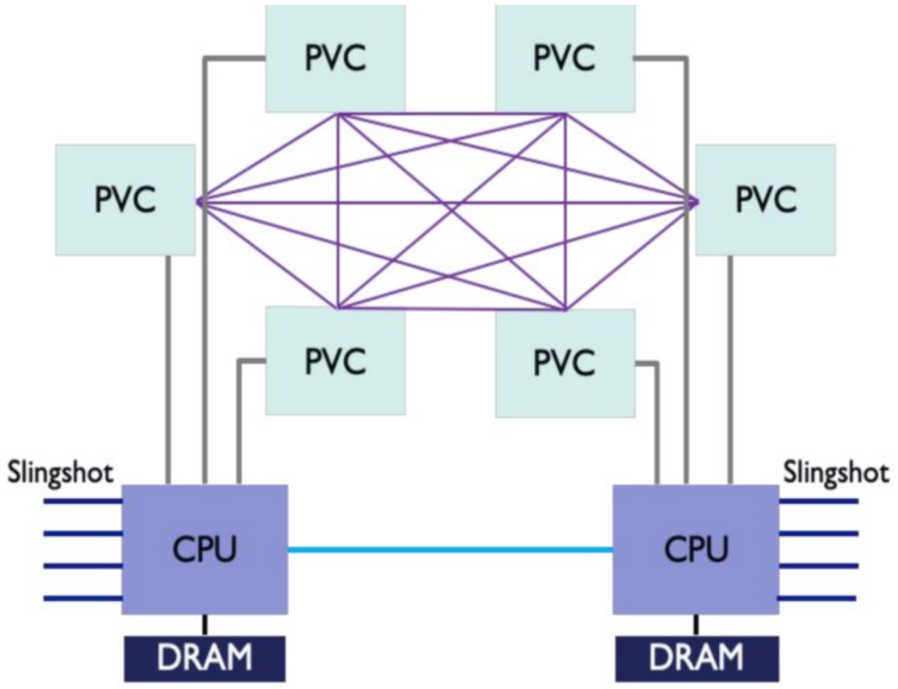
JUWELS-Booster: 2 AMD CPUs, 4 NVIDIA GPUs, NVlink + 4 NIC



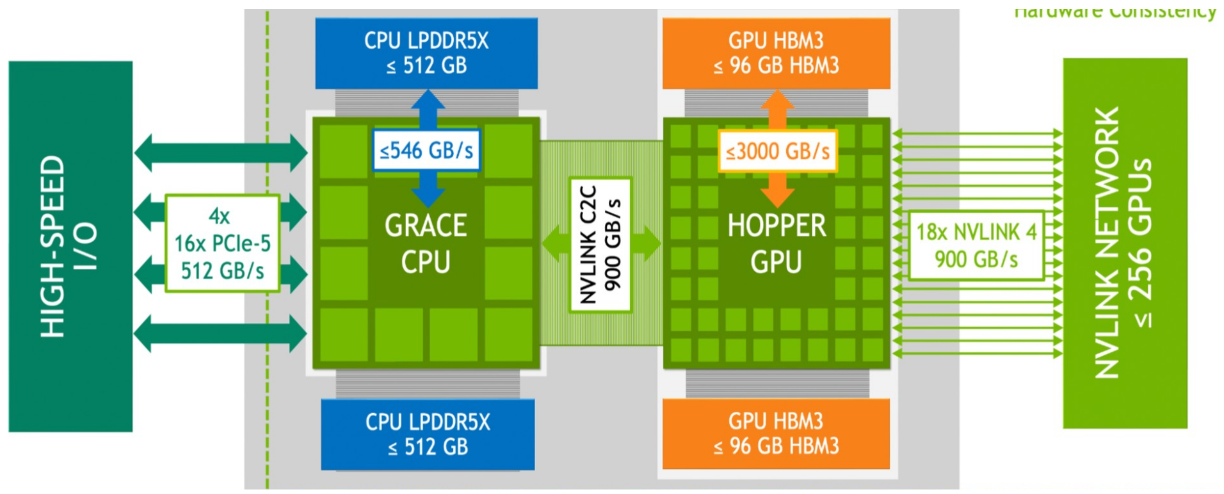
AMD CPU+GPU Exascale architecture: LUMI, Frontier



Intel CPU+GPU Exascale architecture: Aurora



JSC Jupiter 4x NVIDIA Grace-Hopper + Nvlink + 4 NIC



Evolution of GPU hardware & API support

1st heterogeneous parallelization:
Force offload mode: Nonbonded
CUDA v4.6 (2013)

OpenCL portablity backend
(AMD / NVIDIA)
NB force offload
v5.1 (2015)

Force offload:
PME, dual pair list
CUDA & OpenCL
v2016

OpenCL improvements &
Intel support
Force offload:
bondeds (CUDA)
v2019

CUDA: GPU-resident mode,
early support for
direct-GPU comms
v2020

SYCL: early support on Intel
Better FEP support
& improvements
in CUDA
v2021

Direct GPU comm:
optimized P2P &
CUDA-aware MPI;,
PME decomposition
with HeFFT backend
v2022

GPU-resident SYCL
on Intel (dpc++) and
AMD (hipSYCL);
v2022

CUDA-graph
single/multi-GPU,
cuFFTmp support
v2023

SYCL bonded offload
and PME
decomposition with
HeFFTe (Intel/AMD)
v2023

CUDA-graph opt
post-prune pair-list
sort in CUDA
Early work on GPU-initiated comm.
v2024

SYCL AMD
optimizations
runtime
improvements
v2024

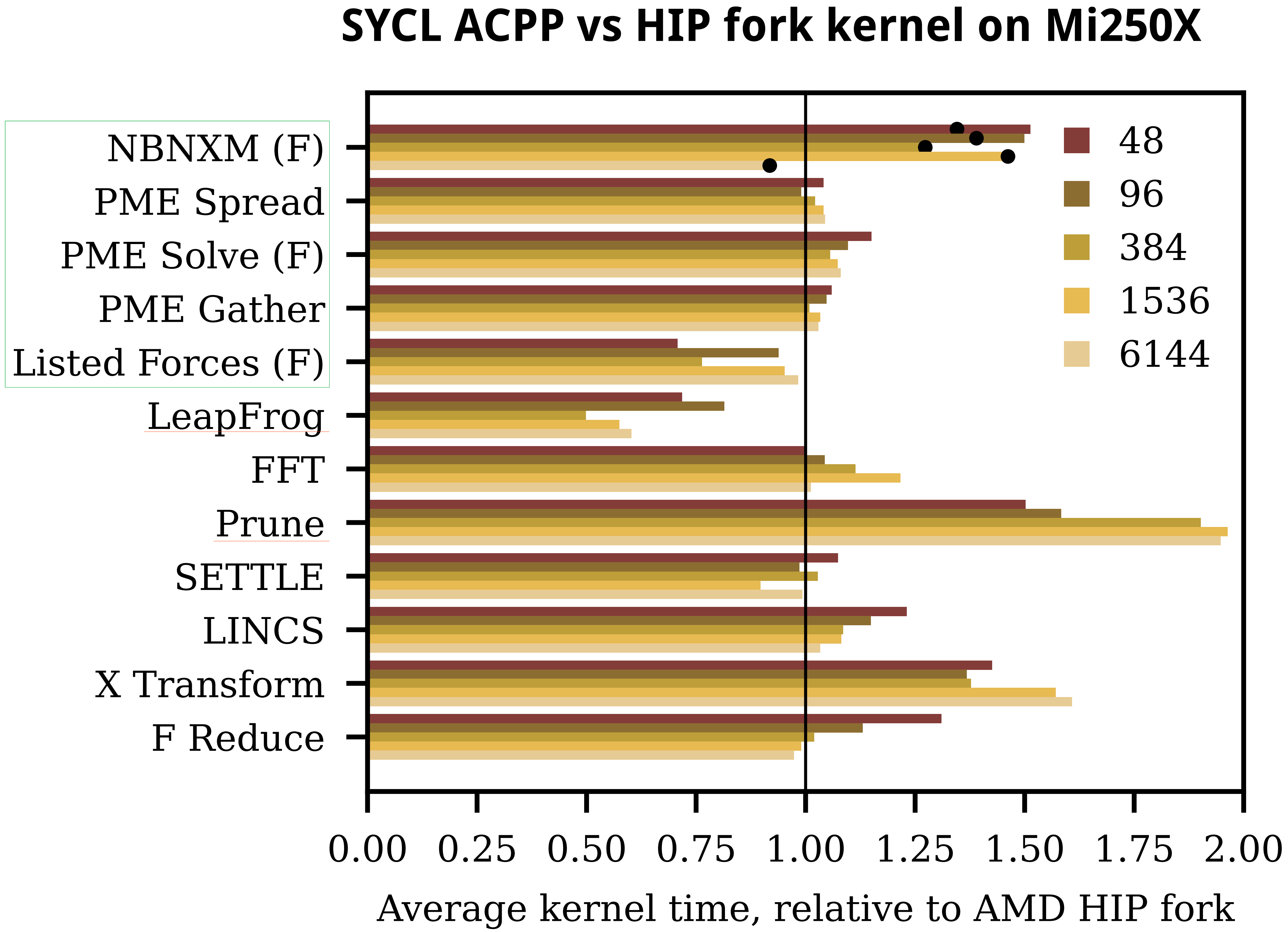
State of the SYCL backend in GROMACS 2024

- Feature support:
  - close to parity with CUDA backend (no P2P intra-node comm, WIP graph scheduling)
  - primary portability backend to replace OpenCL (broader feature support)
- Hardware support:
  - **Intel** (production): desktop & server
  - **AMD** (production): CDNA and (some) RDNA*

    *due to poor ROCm support for some consumer hardware OpenCL is still needed
  - NVIDIA (portability): all desktop and server
- Runtime support:
  - **DPC++ for Intel** (NVIDIA and AMD support experimental)
  - **AdaptiveCPP (hipSYCL) on AMD** and NVIDIA
- Library integration: MKL, VkFFT, rocFFT, HeFFTe

SYCL for AMD systems: kernels

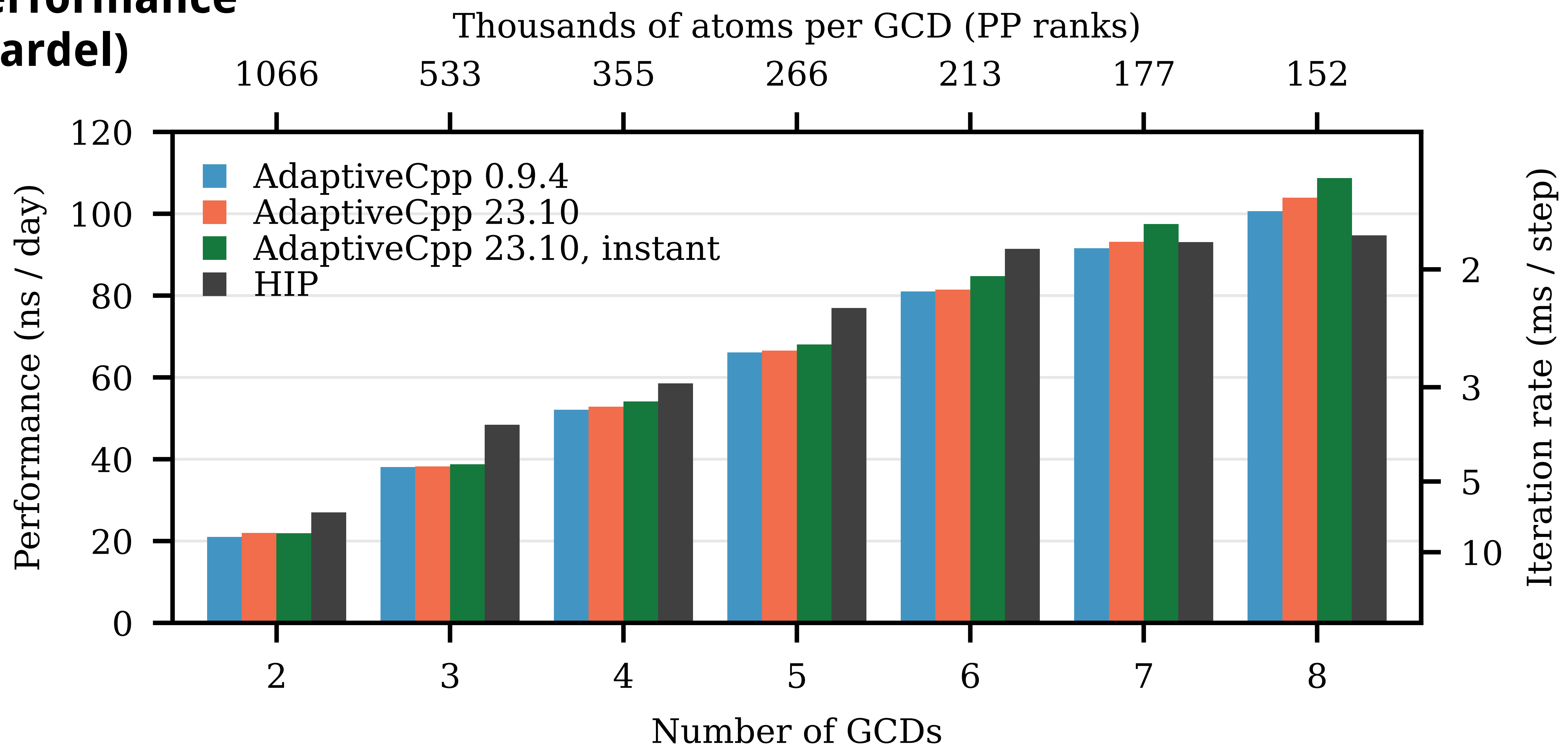
- Kernels close in performance with native
  - some complex kernels slower due to compiler issues
  - a few compiler bug / codegen workarounds not ported over:
    - maintainability / tech debt concerns
  - some kernels faster
- Note: implementations have diverged (HIP fork based on 2021-beta vs upstream 2024)



https://arxiv.org/pdf/2405.01420

SYCL on AMD systems: intra-node performance

GROMACS SYCL performance on Cray EX235a (Dardel)

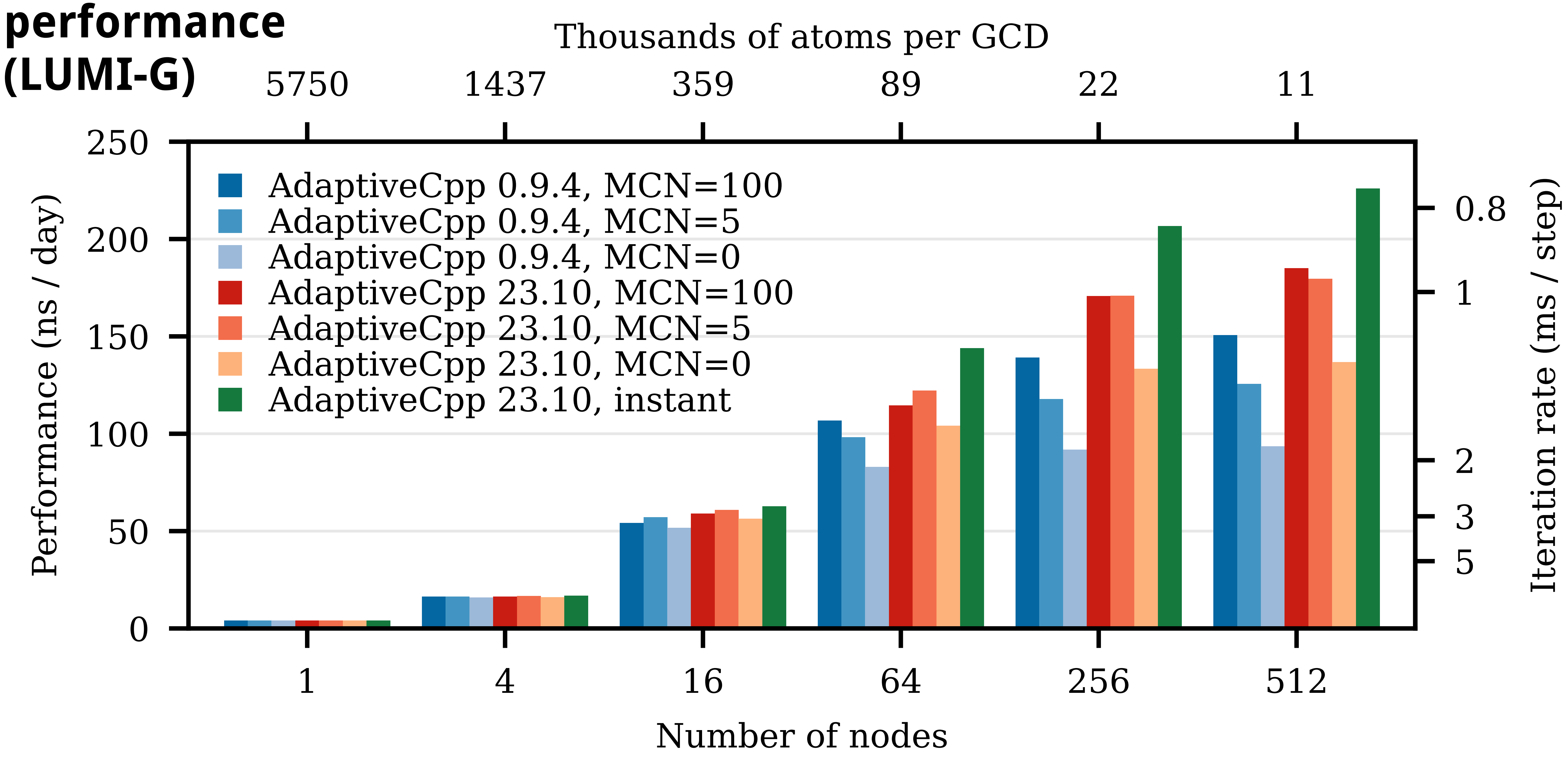


- ACPP runtime optimizations bring increasing benefits in scaling
- GROMACS 2024 outperforms HIP fork on 7-8 GCDs
- Scaling limitation with PME: lack of highly optimized distributed 3D-FFT on AMD GPUs

https://arxiv.org/pdf/2405.01420

SYCL on AMD systems: runtime optimizations

GROMACS SYCL performance on Cray EX235a (LUMI-G)

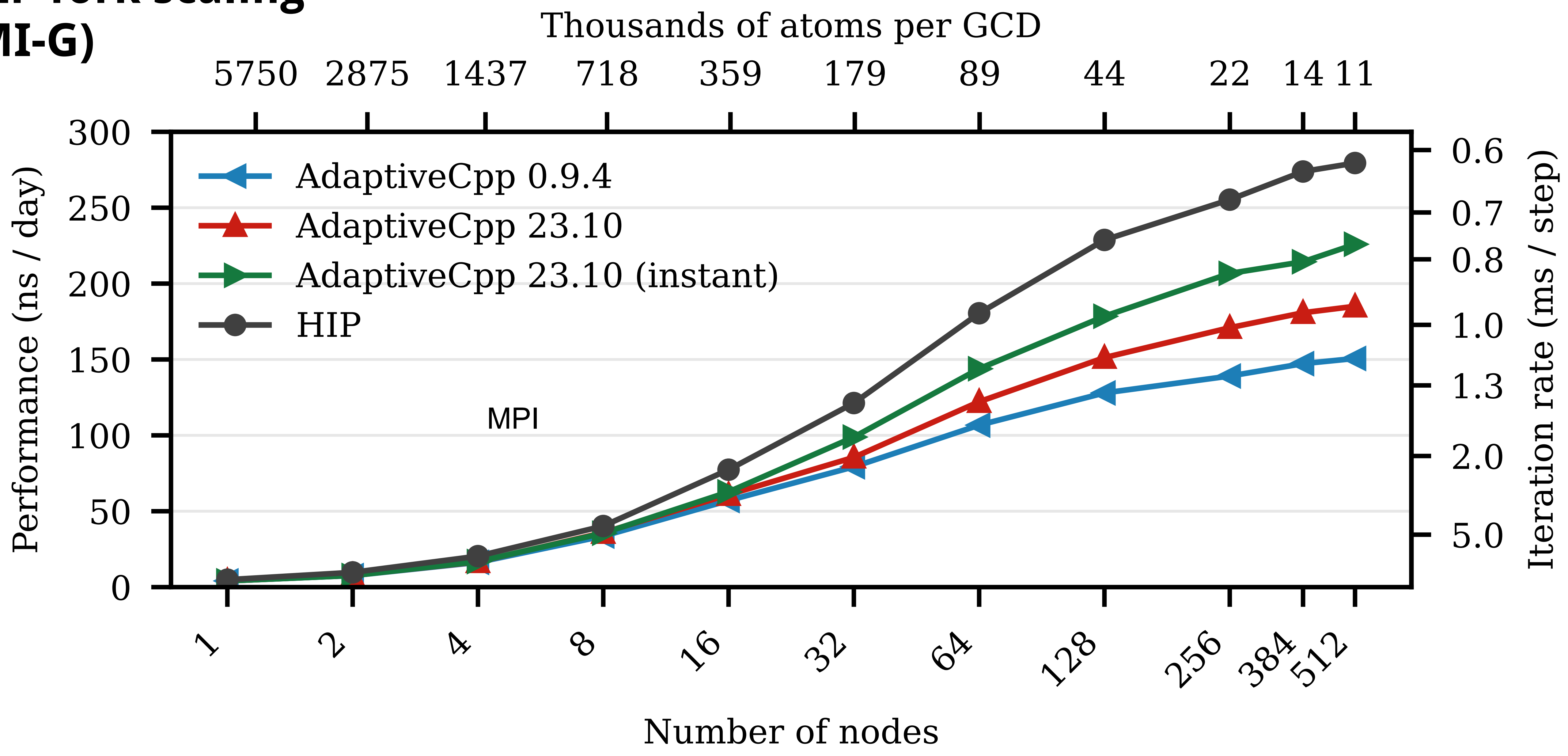


- 2023 focus: collaboration with the AdaptiveCPP team to improve runtime overheads
  - coarse grained events
  - latency optimizations to deferred execution mode
  - instant submission mode: bypass deferred execution

https://arxiv.org/pdf/2405.01420

SYCL on AMD systems: strong scaling

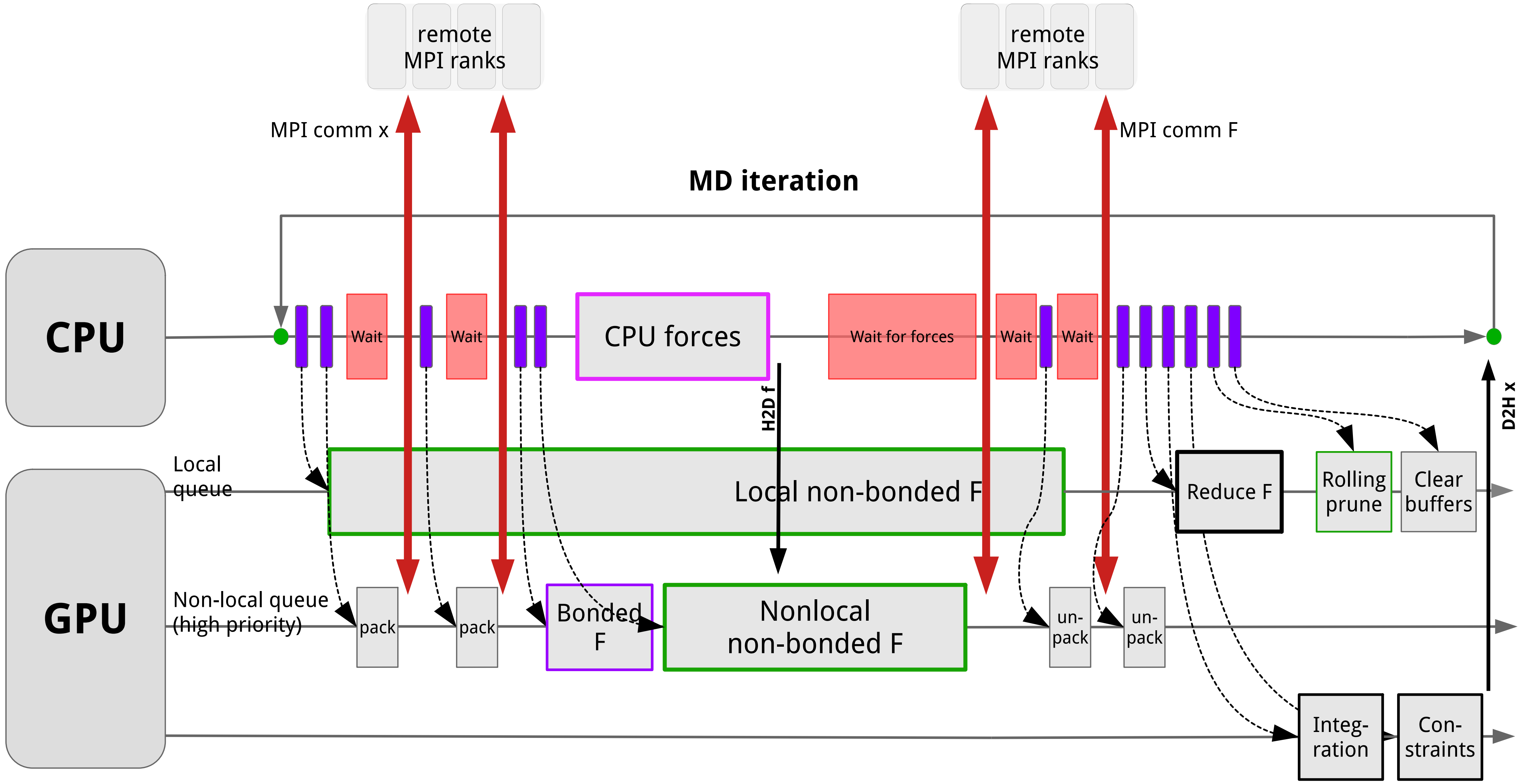
GROMACS SYCL vs HIP fork scaling on Cray EX235a (LUMI-G)



- Strong scaling of domain decomposition on up to 512 LUMI-G nodes
  - parallel efficiency with ACPP instant submission on par with HIP fork
  - absolute performance only ~15-20% from HIP fork (mainly due to compute kernels)

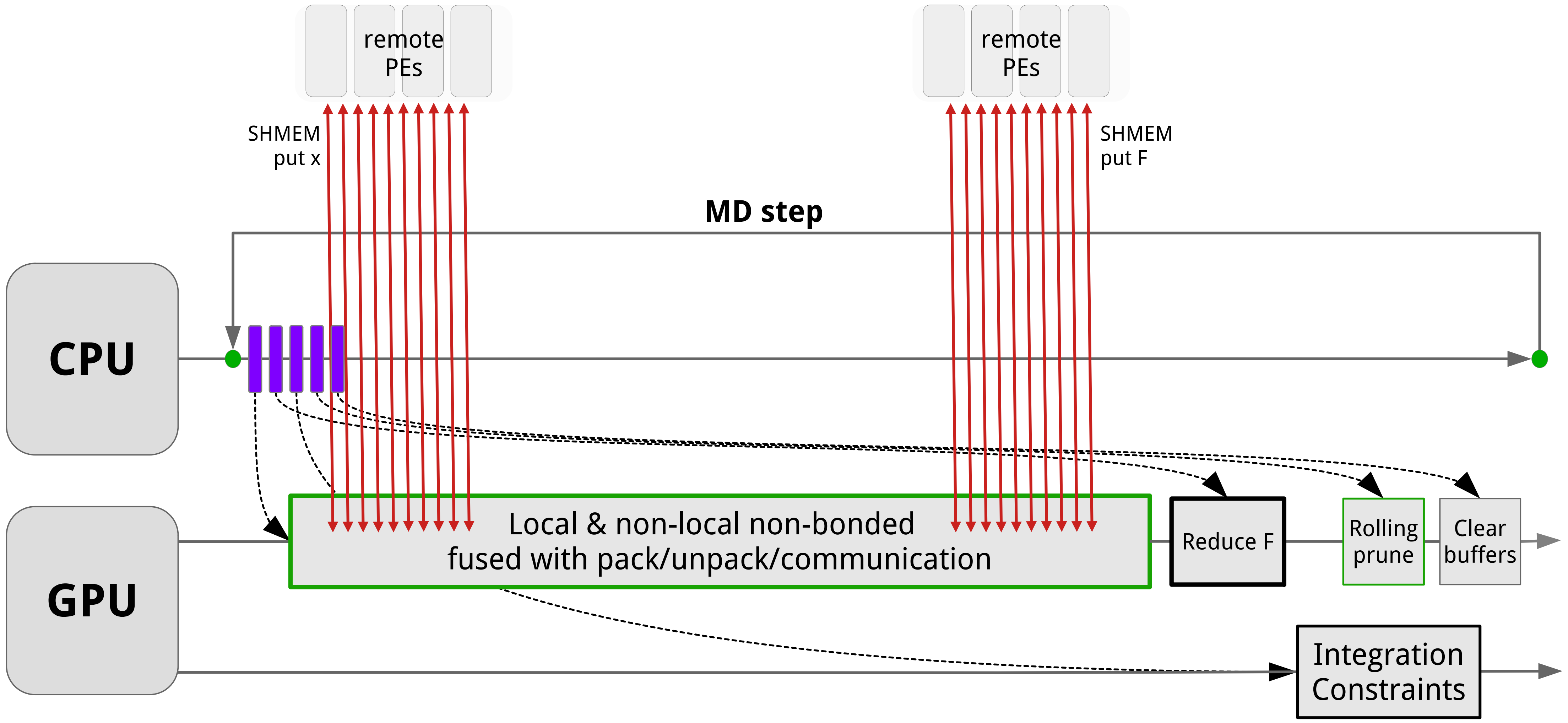
https://arxiv.org/pdf/2405.01420

Strong scaling limitation: CPU-initiated communication



- MPI not sufficiently GPU-aware
- Multiple syncs on critical path
  - adds latency overheads to the critical path
  - prevents scheduling ahead-of-time (and hiding launch overhead)

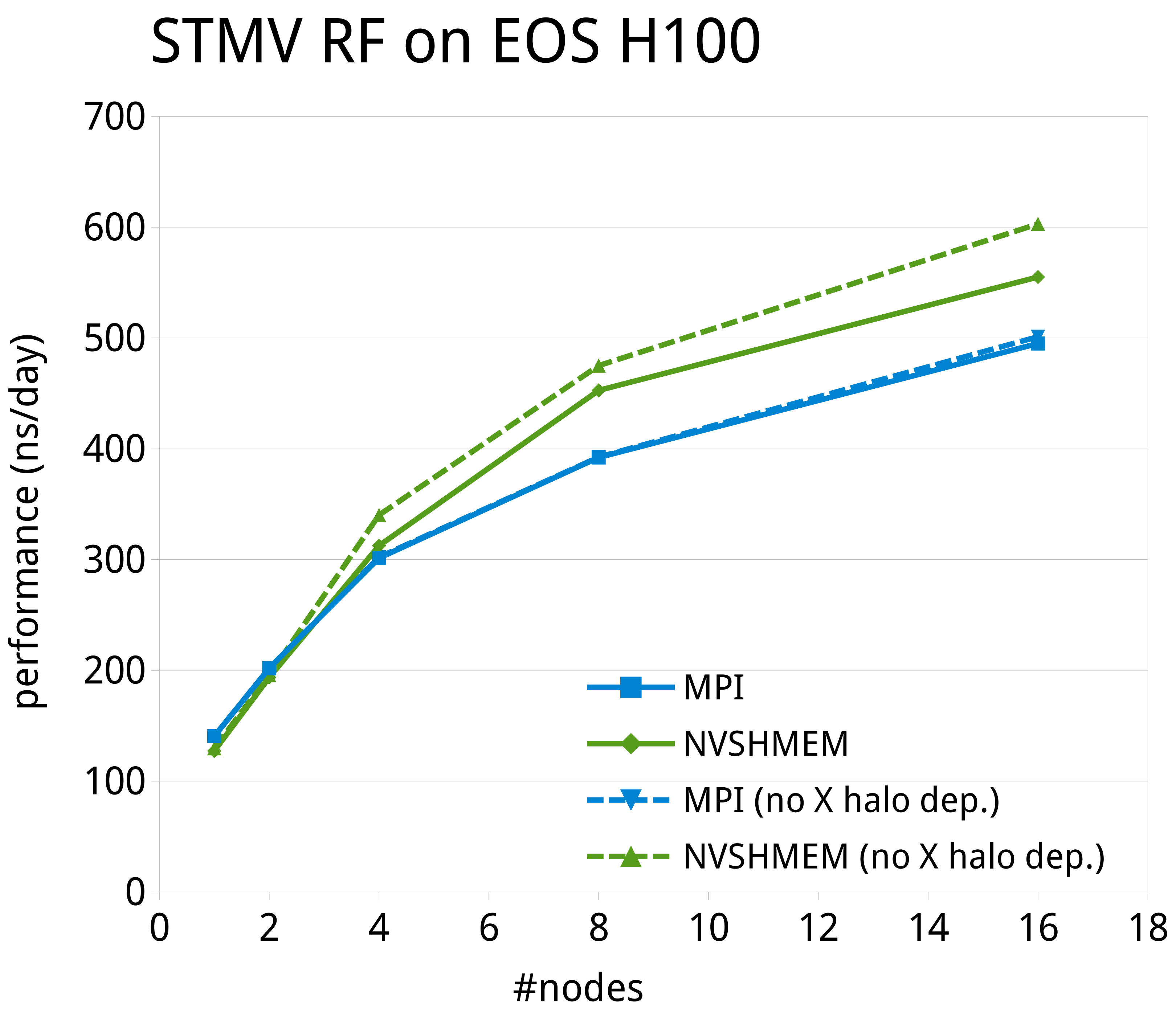
GPU-initiated communication: long-term



- Fine-grained GPU-initiated communication: **NVSHMEM** (MPI one-sided)
  - reduce latency:
    - avoid CPU-initiated round-trip/wait
    - fuse kernels: avoid launch latencies
  - make use of the GPU hardware latency hiding abilities

GPU initiated communication: preliminary performance

- NVSHMEM prototype shows promising performance
- current halo-exchange algorithm limiting: volume optimized indirect comm
- algorithmic changes needed:
  - switch to direct communication
  - increase communication concurrency
  - estimated performance impact shows improvements with NVSHMEM



Acknowledgments

GROMACS

Andrey Alekseenko

Berk Hess

Erik Lindahl

Magnus Lundborg

Mark Abraham (Intel)

Paul Bauer (AMD)

Alan Gray (NVIDIA)

Ania Brown (NVIDIA)

Gaurav Garg (NVIDIA)

Mahesh Doijade (NVIDIA)

AdaptiveCPP (formerly hipSYCL)

Aksel Alpay

HW / code contribution

Funding

We are hiring!

Researcher in High Performance Computing

at PDC, KTH Royal Institute of Technology

to work on GROMACS and Neko

https://www.kth.se/lediga-jobb/712497?l=en