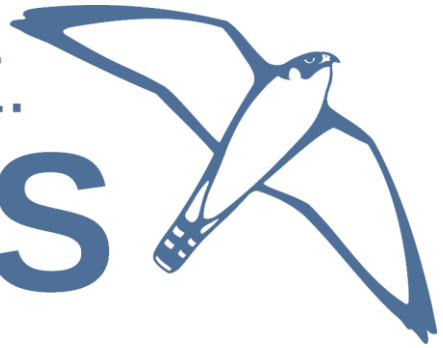
FAST. FLEXIBLE. FREE.

GROMACS



Using SYCL for Performance and Portability at Scale

PASC24
Zurich Switzerland | 3-5 June 2024

Andrey Alekseenko
KTH Royal Institute of Technology & SciLifeLab
Stockholm, Sweden

GROMACS

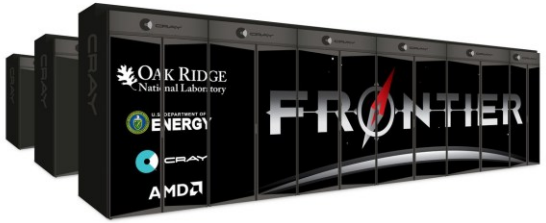
- Open-source molecular dynamics engine
  - 475k lines of C++17 code; 33 years of history
  - Code review, automated testing, sanitizers, etc
- High-performance for a wide range of modeled systems
  - From 10^4 to 10^9 particles
- … and on a wide range of platforms
  - from laptops to supercomputers
  - x86-64, ARM, POWER, RISC-V
  - AMD, Apple, Intel, and NVIDIA GPUs; Intel Xeon Phi
  - Linux, Windows, MacOS, *BSD

GROMACS

- Multi-level parallelism:
  - MPI (task- and domain decomposition)
  - OpenMP
  - CUDA / OpenCL / SYCL
  - SIMD (intrinsics)
- Efficient scaling:
  - Cache- and locality-optimized algorithms
  - Flexible offloading scheme: GPU-resident or heterogeneous
  - Direct GPU-GPU communication
    - GPU-aware MPI; *very* early GPU-initiated support (NVSHMEM)
  - Scalable distributed FFT (cuFFTMp, heFFTe)

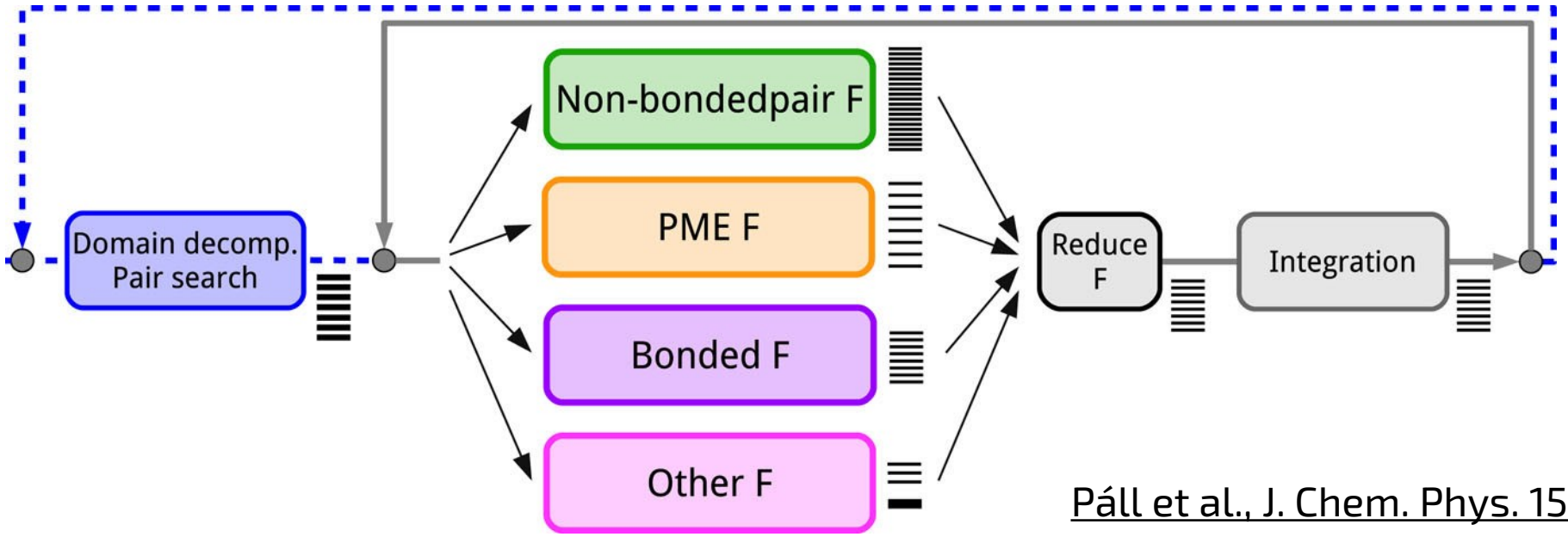
SYCL support in GROMACS 2024

- Primary targets for SYCL backend:
  - **AMD CDNA2** GPUs with **AdaptiveCpp**
  - **Intel Xe-HPC** GPUs with **oneAPI**
- Secondary targets for SYCL backend :
  - Other AMD GPUs with oneAPI and AdaptiveCpp
  - Other Intel GPUs with oneAPI
- Should work with SYCL:
  - NVIDIA GPUs with oneAPI and AdaptiveCpp
- CUDA for NVIDIA GPUs, OpenCL for Apple



Molecular dynamics: science at 1000 fps

- Iterative problem
- One step ~1 fs, need to simulate µs to ms
  - 10^9-10^{12} steps



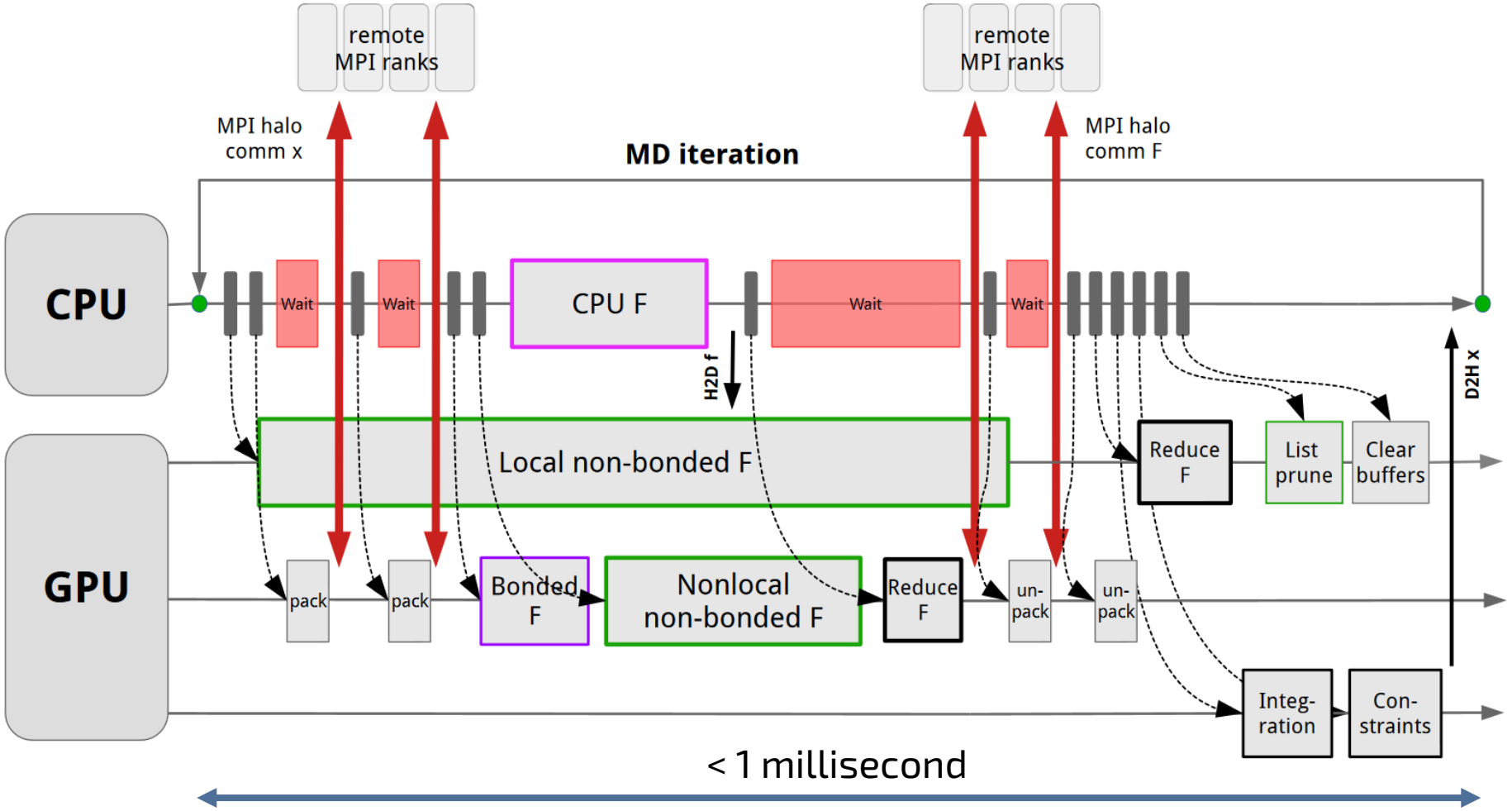
Páll et al., J. Chem. Phys. 153, 134110 (2020)

Heterogeneous parallelization

- Minimize latency
- Minimize CPU and GPU stalls
- Minimize data exchange between host and device
  - And between nodes

- Optimal offloading scheme depends on simulated system
  - And on available hardware

Molecular dynamics: example schedule



Alekseenko *et al.*, arXiv:2405.01420

GROMACS: GPU acceleration architecture

- Designed for CUDA
  - Multiple in-order queues
    - with different priorities
  - Manually-managed memory allocation and movement
  - Explicit event-based synchronization
    - Both between device streams and between host and device

- Abstraction layer added for OpenCL
  - Device and memory management, synchronization

SYCL Buffers or USM

- Buffers prevent bugs and improve performance
  - At least in theory
- GROMACS is built around in-order queues
  - Additional divergence between backends when using buffers
- Solution: use in-order queues and USM
- Bonus:
  - Accessors are hard to optimize for compiler
  - Easier interop with native libraries
    - USM pointers *are* native pointers

Synchronization events

- We use barriers in CUDA and OpenCL
- SYCL does not have them

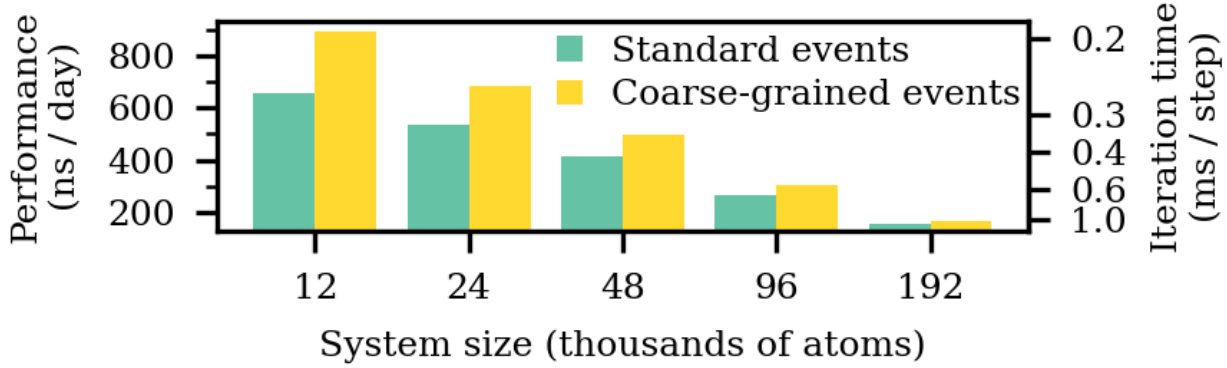
- Event can be recorded far from the last submission
  - Not easy to tell which operation should be used for synchronization

- Solution: Use extensions to mark events:
  - oneAPI: `sycl_ext_oneapi_enqueue_barrier`
  - AdaptiveCpp: `hipSYCL_enqueue_custom_operation`

Synchronization events: runtime overhead

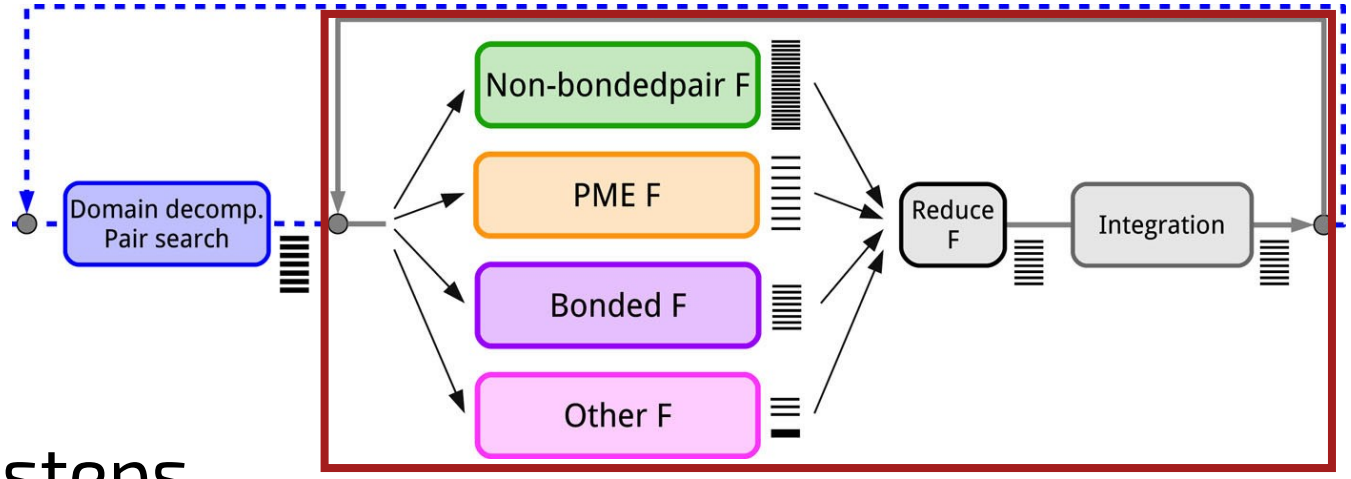
sycl::event sycl::queue::submit(…) ⇒ (cuda|hip)Event(Create|Record|Destroy)

HIPSYCL_EXT_COARSE_GRAINED_EVENTS:

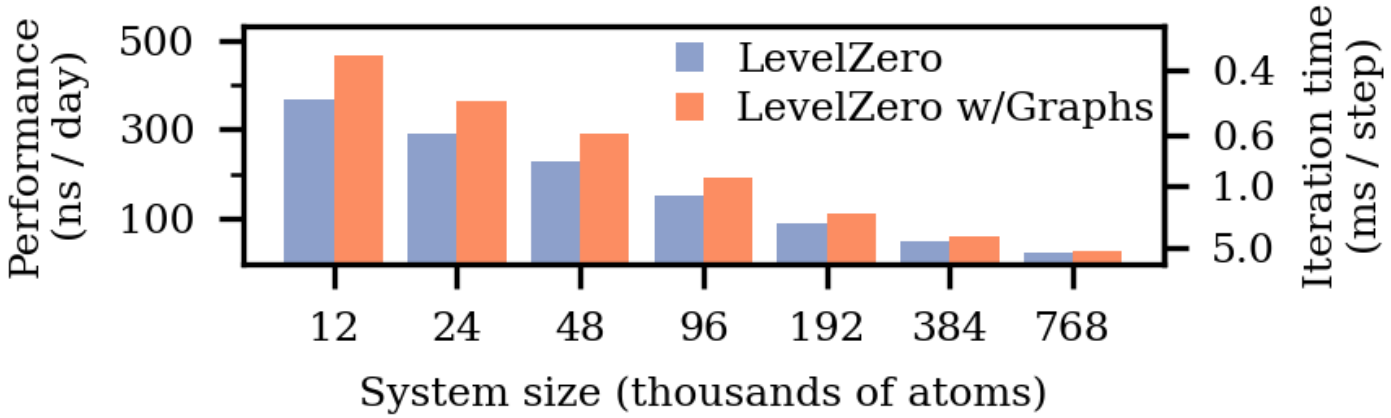


GROMACS 2024.0, AdaptiveCpp 23.10 + ROCm 5.3.3, AMD MI250X

Synchronization events: SYCL Graphs

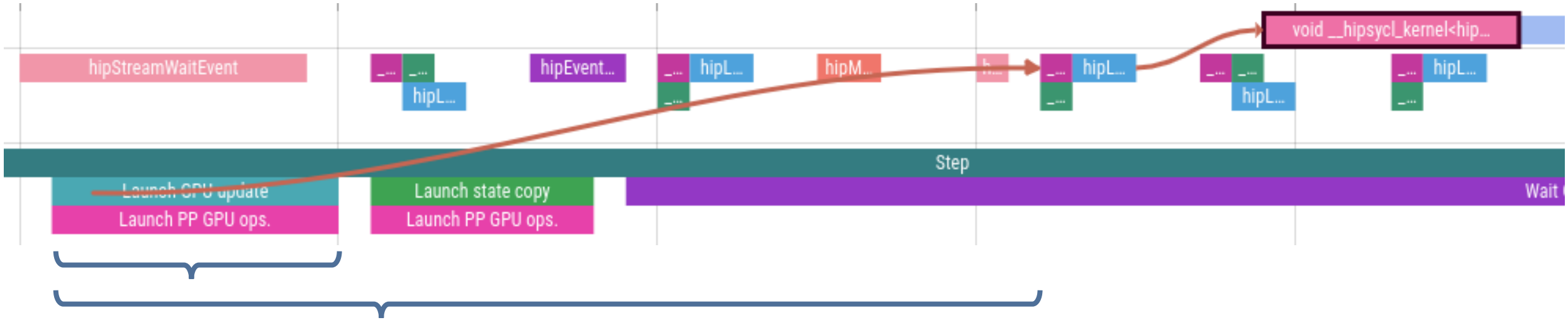


- Repetitive steps
- ~10 operations in each
- “Record-replay” mode
- oneAPI Graph extension
  - Still very WIP



GROMACS 2025-dev, oneAPI nightly 20240531, Intel Arc A770

Runtime asynchronicity



8µs	72µs	MCN=100
15µs	30µs	MCN=0
40µs	14µs	Instant

Max. Cached Nodes (MCN) controls how eagerly worker threads start submitting tasks to GPU.

“Instant” mode behaves like native CUDA/HIP, without extra threads.

GROMACS 2024.beta, ROCm 5.3.3, AdaptiveCpp 23.10.0, Dardel GPU. **Qualitative** data for illustration only.

Native libraries

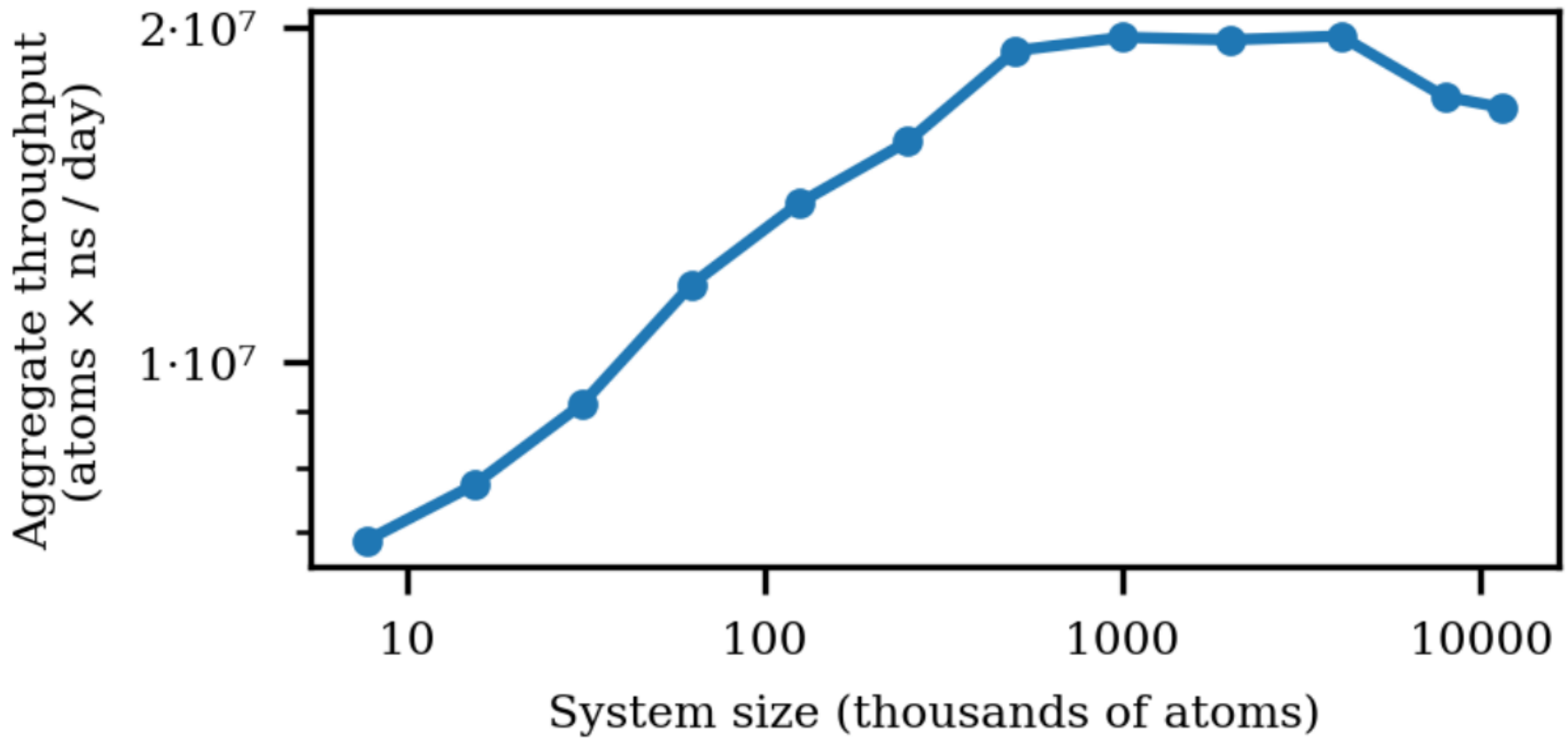
- External libraries: GPU-aware MPI, FFT
- Native SYCL libraries only available for Intel hardware:
  - IntelMPI; oneMKL, BBFFT
- With USM, native ones work fine via interop!
  - USM pointer *is* native pointer
  - Submitting operations/synchronizing with them is more complicated
    - AdaptiveCpp has dedicated API for such cases
    - oneAPI does not, but the implementation allows it
  - OpenMPI, MPICH; cuFFT, rocFFT, vkFFT, HeFFTe
- Required effort:
  - CMake and thin wrapper/launcher

Code portability in practice

Vendor-specific code:

- Sub-group-size-dependent algorithms
- Overload some functions to use architecture-specific intrinsics
- FFT invocation
- Extensions for synchronization events and runtime hints
- A lot of CMake and user-facing documentation

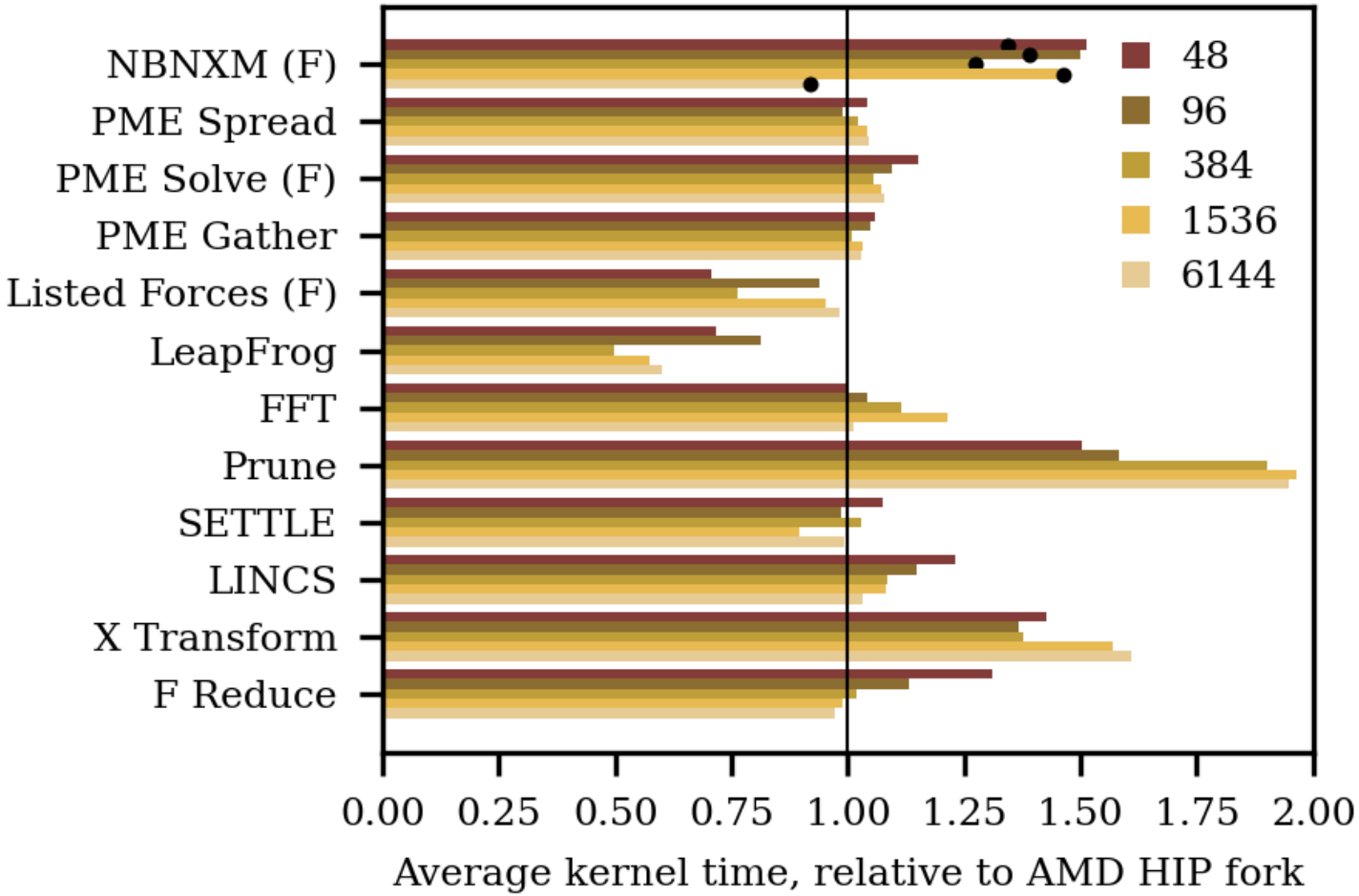
Ensemble performance on Intel PVC1550



GROMACS 2023, SuperMUC NG Phase 2 (8 GPU tiles); no coupling

Adapted from Dobrev, Pribec, Mathias, P25 - GPU Benchmarking on Fully Occupied Accelerated Cluster Nodes […]

Kernel performance, vs. native HIP

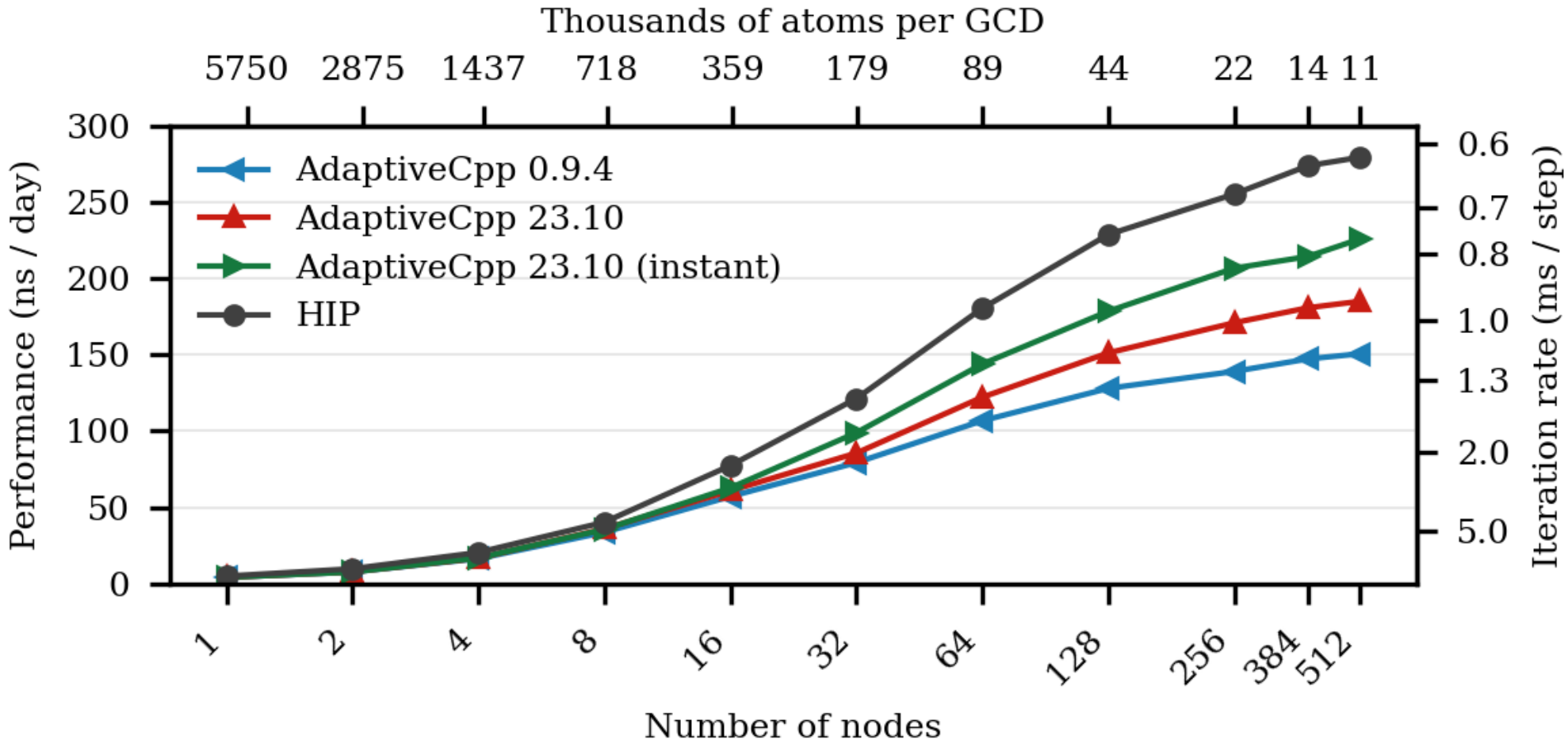


- Many SYCL kernels are close to HIP
  - Some are faster!
- NBNXM is the largest
  - Pair list sorting (work-in-progress)
  - Compiler codegen optimizations
  - Better parameter tuning
- Seen 20% perf. difference between ROCm 5.3-5.6 for the same kernel

Performance/maintainability balance!

GROMACS 2024.0, AdaptiveCpp 23.10 + ROCm 5.3.3; AMD HIP fork based on 2022.beta2

Multi-node scaling on LUMI-G



GROMACS 2024.1 / GROMACS HIP, ROCm 5.4.6, LUMI-G (8xAMD MI250X per node), Grappa RF (46M atoms)

Results

- SYCL is a feature-complete GPU backend in GROMACS
- (Almost) same code running on AMD, Intel, and NVIDIA GPUs
  - Minimal device-specific optimizations
  - Performance typically within 20% of **highly-optimized** native code
  - Downside: any code change requires extensive testing
    - Any change is the whole underlying software stack too!

- Production ready:
  - Used on LUMI today, scaling across multiple GPUs

Next challenges

- GPU-initiated communications
  - SHMEM, MPI RMA, Stream-aware MPI?
- 3D FFT strong scaling
  - HeFFTe does not scales as well as cuFFTMp
- Unified memory architectures
  - AMD MI300A, NVIDIA Grace Hopper
- AdaptiveCpp SSCP mode support
- Make the installation less of a pain
  - Spack?

Acknowledgements

- Aksel Alpay (Heidelberg University)
- Szilárd Páll, Ragnar Sundblad (KTH PDC)
- Maciej Szpindler (LUMI)
- Mark Abraham, Heinrich Bockhorst, Roland Schulz (Intel)
- Julio Maia, Bálint Soproni, Samuel Antao (AMD & StreamHPC)
- And the whole GROMACS community!

Learn more

- https://www.gromacs.org/
- Páll *et al.*, J. Chem. Phys. 153, 134110 (2020)
- Alekseenko *et al.*, arXiv:2405.01420

- **If you have questions: andreyal@kth.se**