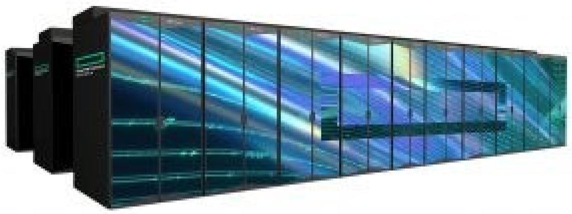
FAST. FLEXIBLE. FREE. GROMACS



SYCL™

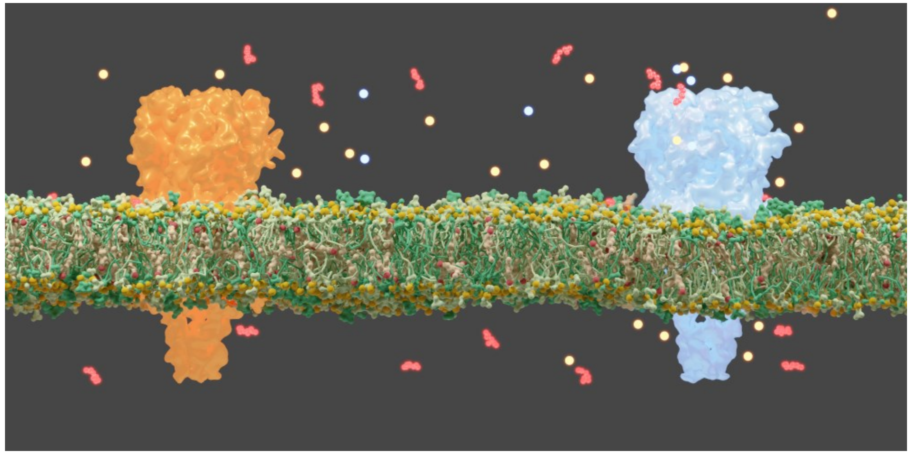
GROMACS on AMD GPU-based HPC platforms: using SYCL for performance and portability

Andrey Alekseenko, Szilárd Páll, Erik Lindahl

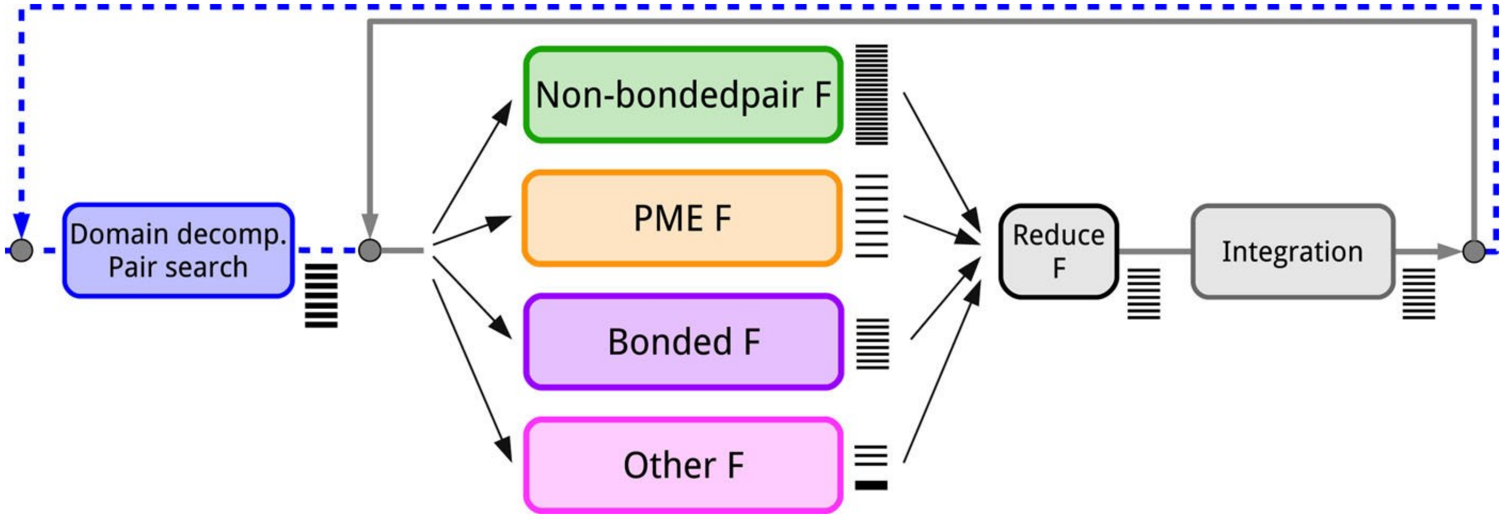
KTH Royal Institute of Technology & SciLifeLab
Stockholm, Sweden
andreyal@kth.se

CUG2024
Diverse Universe
PERTH AUSTRALIA

Molecular dynamics

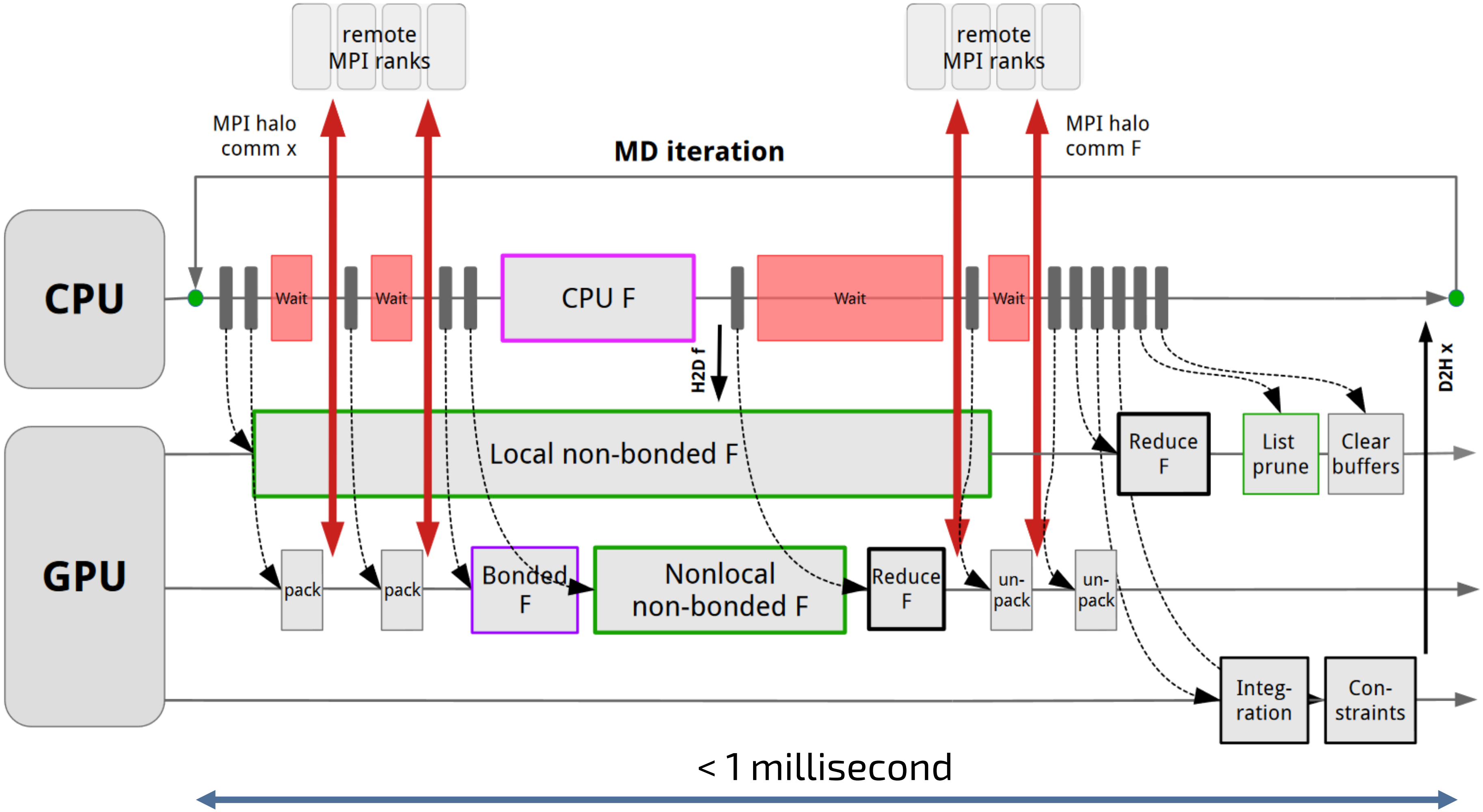


- Newton's equations of motion
- Fixed problem size (~10^4-10^6): limited hardware parallelism
- Fast iterations: timestep ~1 fs, need to reach μs to ms



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

Molecular dynamics: example schedule



GROMACS

- Open-source molecular dynamics engine
  - 470k 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)

GPU support in GROMACS 2020

	NVIDIA CUDA	OpenCL
Non-bonded offload	√	√
PME offload	√	√
Update offload	√	X
Bonded offload	√	X
Direct GPU-GPU comm	√	X
PME Decomposition	X	X
Hardware support	NVIDIA	NVIDIA, AMD, Intel

Simple abstraction layer for resource management and synchronization

SYCL

- Open standard
- High-level
- Kernel-based
- Pure C++17

```
#include <sycl/sycl.hpp>

// Create a queue (stream)
sycl::queue queue{{sycl::property::queue::in_order()}};
// Allocate GPU memory
float* Ad = sycl::malloc_device<float>(n, queue);
// Copy the data from CPU to GPU
queue.copy<float>(Ah, Ad, n);
// Submit a kernel into a queue; cgh is a helper object
queue.submit([&](sycl::handler &cgh) {
 cgh.parallel_for<class Kernel>(sycl::range<1>{n},
   [=](sycl::id<1> i) {
     Cd[i] = Ad[i] + Bd[i];
 });
}).wait();
```

- No support from AMD or HPE/Cray

Why SYCL for AMD GPUs

- Open, vendor-independent **standard**
- Intel GPU support
- Two relevant implementations
  - AdaptiveCpp (previously known as hipSYCL)
  - Intel oneAPI DPC++
  - Both open-source and support all three vendors
- Built upon the HIP toolchain:
  - Day-one hardware support
  - Can use native code inline (up to whole kernel)
  - Vendor tools and native libraries just work
    - oneMKL aims to be a portable library/wrapper for FFT, BLAS, etc.

Hardware 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

GROMACS 2024

- SYCL nearly on par with CUDA in feature-support
  - Forces and update offload, GPU-aware MPI
- Already used for large-scale runs on LUMI
- FAST?FLEXIBLE. FREE. GROMACS
- Two versions of AdaptiveCpp runtime compared
  - 0.9.4 (February 2023)
  - 23.10.0 (October 2023)
- Intel oneAPI also works, but slower for now

GROMACS HIP

- Independent fork by AMD and Stream HPC
  - Based on 2022.beta2 (November 2021)
  - HIPified CUDA version with many optimizations
- A lot of divergence from mainline GROMACS:
  - Conditional kernel fusion to avoid memsets
  - Pair list sorting to improve kernel scheduling
  - Hardware-specific intrinsics (unsafeAtomicAdd, warp_move_dpp)
    - Mostly ported to SYCL with preprocessor guards where appropriate
  - Compiler workarounds (moving code around, casting pointers to different type and back, etc)
  - No pull-down of any correctness fixes or scheduling improvements

Portability vs. device-specific optimizations

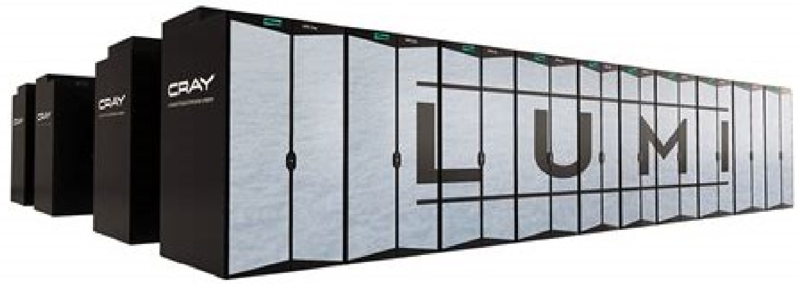
```
float fraction     = normalized - index;
// For some reason the compiler does not generate v_fract_f32 for normalized - floorf(normalized)
float fraction     = __builtin_amdgcn_fractf(normalized);
```

```
    float         inv_r6, c6, c12;
    float         inv_r6;
    float2        c6c12;
#    endif
#    ifdef LJ_COMB_LB
    float         sigma, epsilon;
@@ -461,21 +462,20 @@ __launch_bounds__(THREADS_PER_BLOCK, MIN_BLOCKS_PER_MP)
                    /* LJ 6*C6 and 12*C12 */
                    typei = atib[i * c_clSize + tidxi];
#        ifdef __gfx1030__
                    fetch_nbfp_c6_c12(c6, c12, nbparam, ntypes * typei + typej);
                    c6c12 = fetch_nbfp_c6_c12(nbparam, ntypes * typei + typej);
#        else
                    fetch_nbfp_c6_c12(c6, c12, nbparam, __mul24(ntypes, typei) + typej);
                    c6c12 = fetch_nbfp_c6_c12(nbparam, __mul24(ntypes, typei) + typej);
#        endif
```

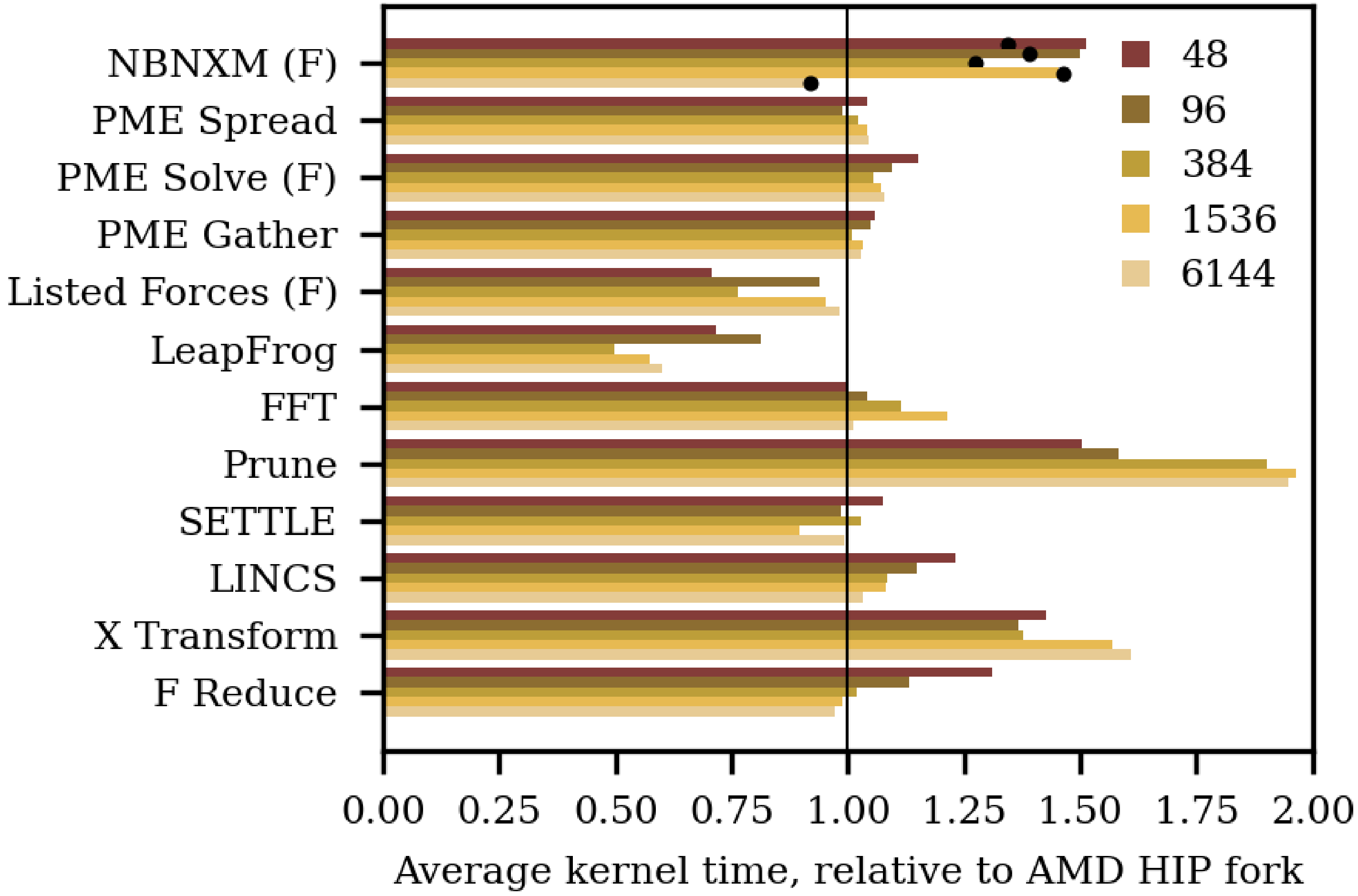
https://github.com/ROCm/Gromacs/

Cray EX235a

- AMD EPYC 7A53 64-core CPU
  - 8 cores per CCX, 8 CCX per socket; some sites have reserved cores
- 4x AMD Instinct MI250X
  - 2 GCD per board, 8 GCDs per node
- 4x Cray Slingshot-11 NICs
  - connected to MI250X



Kernel performance (single GCD)



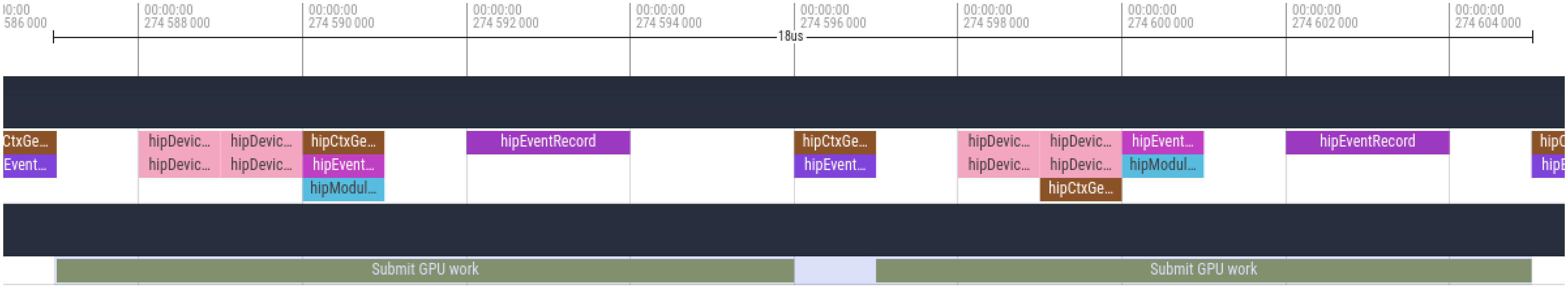
- Many SYCL kernels are close to HIP
  - Some are faster!
- NBNXM is the largest
  - Pair list sorting (work-in-progress)
  - Compiler codegen optimizations
  - Better parameter tuning
- Seen 20% perf. difference between ROCm 5.3-5.6 for the same kernel

Performance/maintainability balance!

GROMACS 2024.0, ROCm 5.3.3, Dardel GPU, Grappa PME

Runtime performance: events

API spec: sycl::event sycl::queue::submit(...)
=> (cuda|hip)Event(Create|Record|Destroy)



Solution: **HIPSYCL_EXT_COARSE_GRAINED_EVENTS**

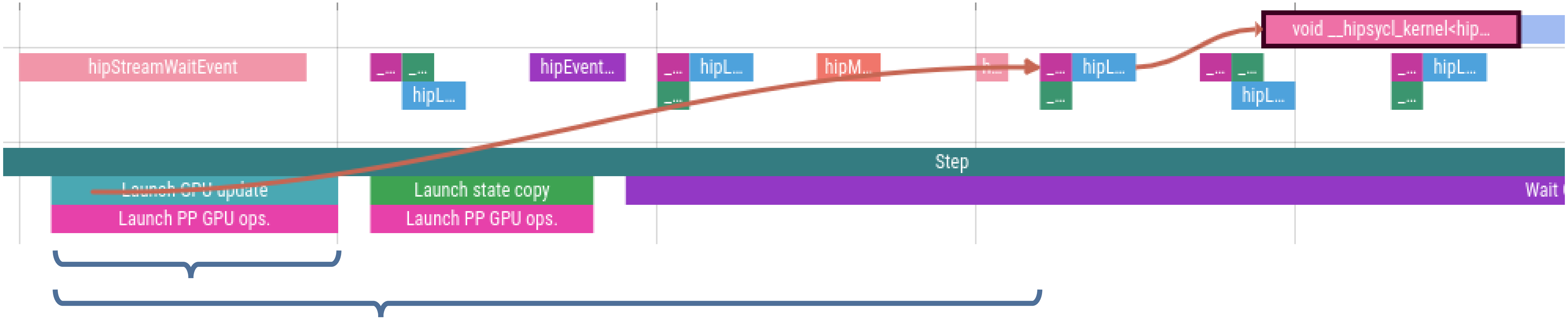
Note: the figure is with oneAPI, which still has this problem

Runtime performance: asynchronicity

- AdaptiveCpp calls HIP API asynchronously from separate thread
  - Optimized in AdaptiveCpp 23.10.0, but still needs 1 CPU
  - Only 8C/16T per GCD, minus reserved cores
  - Minus one more CPU for AMD HSA worker thread

- Does this asynchronous submission introduce latency?
  - Yes
- Do we even want asynchronous submissions?
  - No (?)

Runtime performance



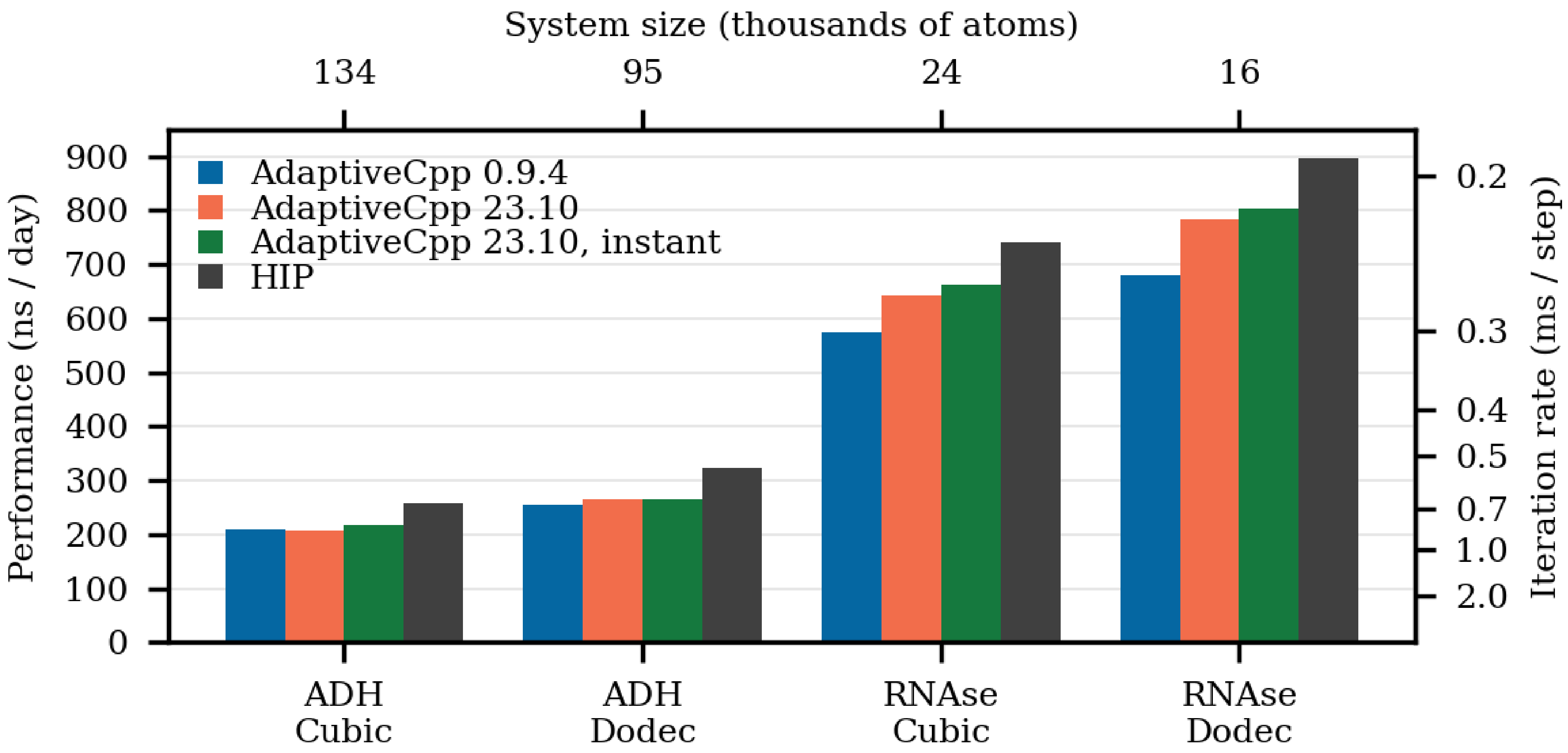
8µs	72µs	MCN=100
15µs	30µs	MCN=0
40µs	14µs	Instant

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

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

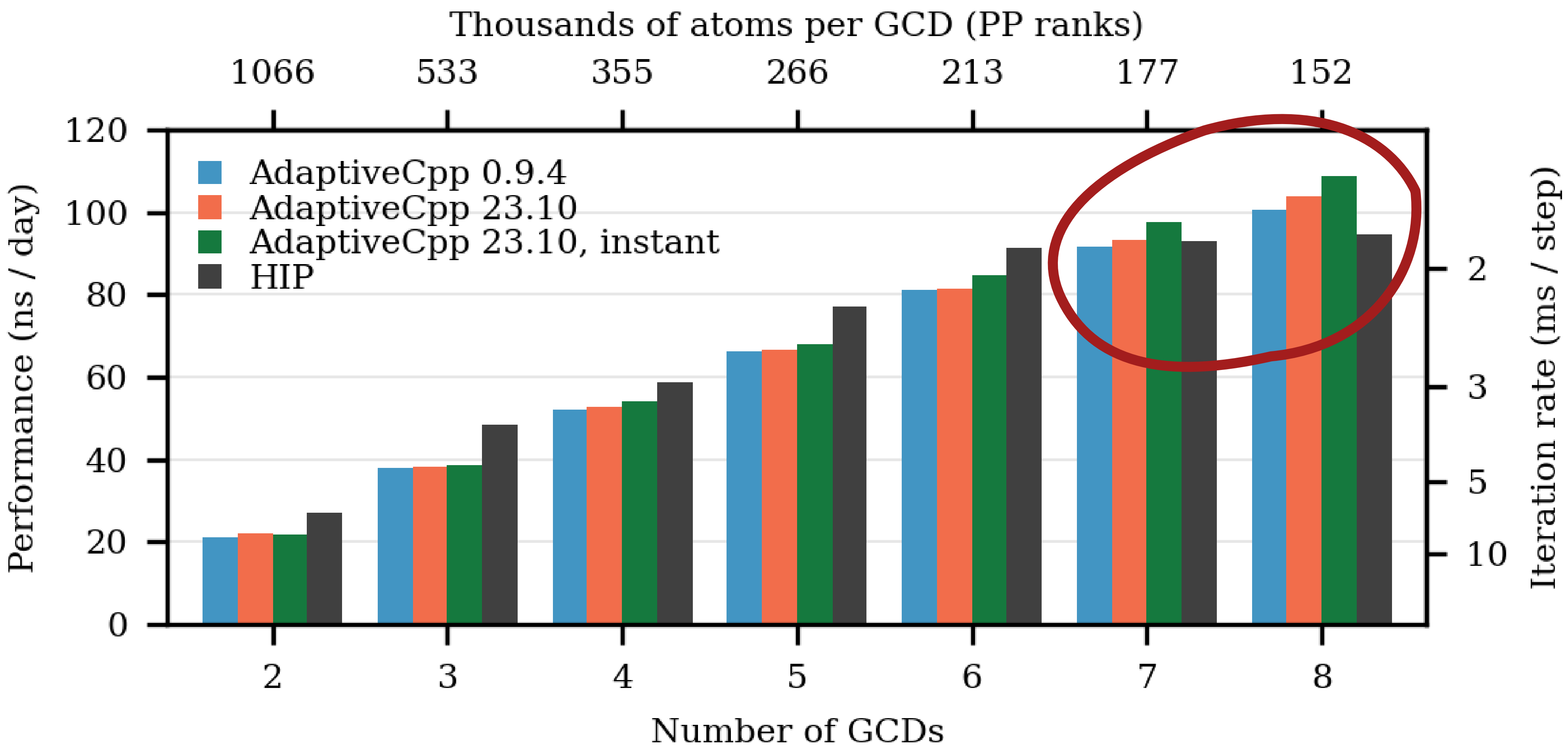
GROMACS 2024.beta, ROCm 5.3.3, AdaptiveCpp 23.10.0, Dardel GPU. **Qualitative** data for illustration only.

Single-GCD performance



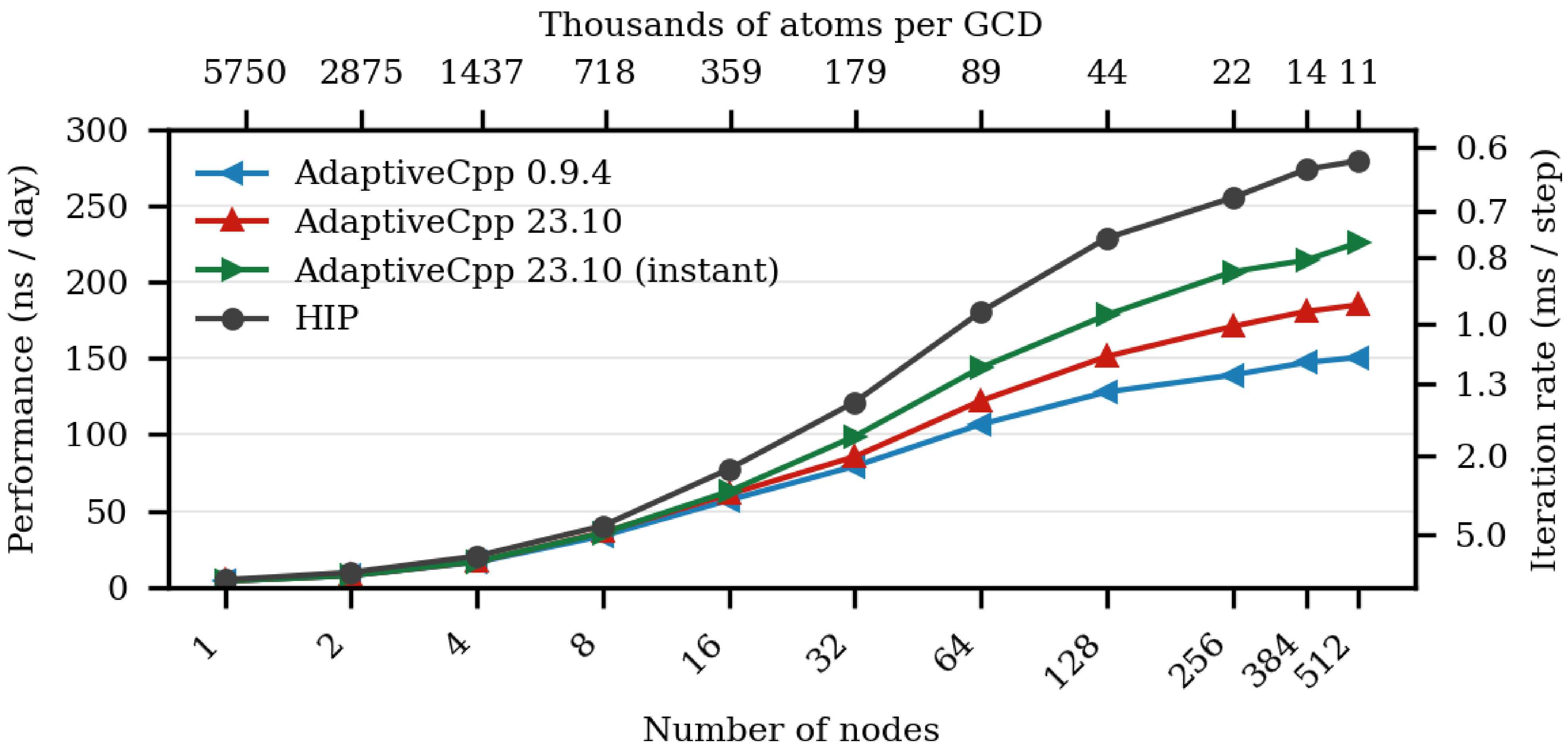
GROMACS 2024.0 / GROMACS HIP, ROCm 5.3.3, Dardel GPU

Single-node performance



GROMACS 2024.0 / GROMACS HIP, ROCm 5.3.3, Dardel GPU, STMV (1M atoms)

Multi-node scaling



GROMACS 2024.1 / GROMACS HIP, ROCm 5.4.6, LUMI-G, Grappa RF (46M atoms)

Conclusions

- Portability:
  - Same code running on AMD, Intel, and NVIDIA GPUs
    - Minimal device-specific optimizations
- Performance:
  - Kernel performance is lower, but does not have to be
    - Complete native kernels can be dropped in
    - We know where the performance is lost
      - Need to find the balance between maintainability and performance
      - But easier maintenance also leads to performance improvements
  - SYCL runtime required initial effort, but now works well
    - Other projects can benefit!

Conclusions

- Can SYCL be used on exascale-class AMD GPU-based systems?
  - **Yes!** Even for challenging cases like MD!

- Next challenges:
  - GPU-initiated communications
    - SHMEM, MPI RMA, Stream-aware MPI?
  - 3D FFT strong scaling
  - Optimizations for unified memory architectures
  - HIP port scales worse than CUDA, many runtime issues
  - Make the installation less of a pain

Acknowledgements

- Aksel Alpay
- Maciej Szpindler
- Ragnar Sundblad
- Julio Maia, Bálint Soproni, Samuel Antao
- And the whole GROMACS community!

intel®
Swedish Foundation for Strategic Research
SeRC Swedish e-Science Research Centre
NAISS National Academic Infrastructure for Supercomputing in Sweden
bioexcel