SYCL™

FAST. FLEXIBLE. FREE.
GROMACS

SYCL for Performance and Portability: A Molecular Dynamics Case Study

Andrey Alekseenko

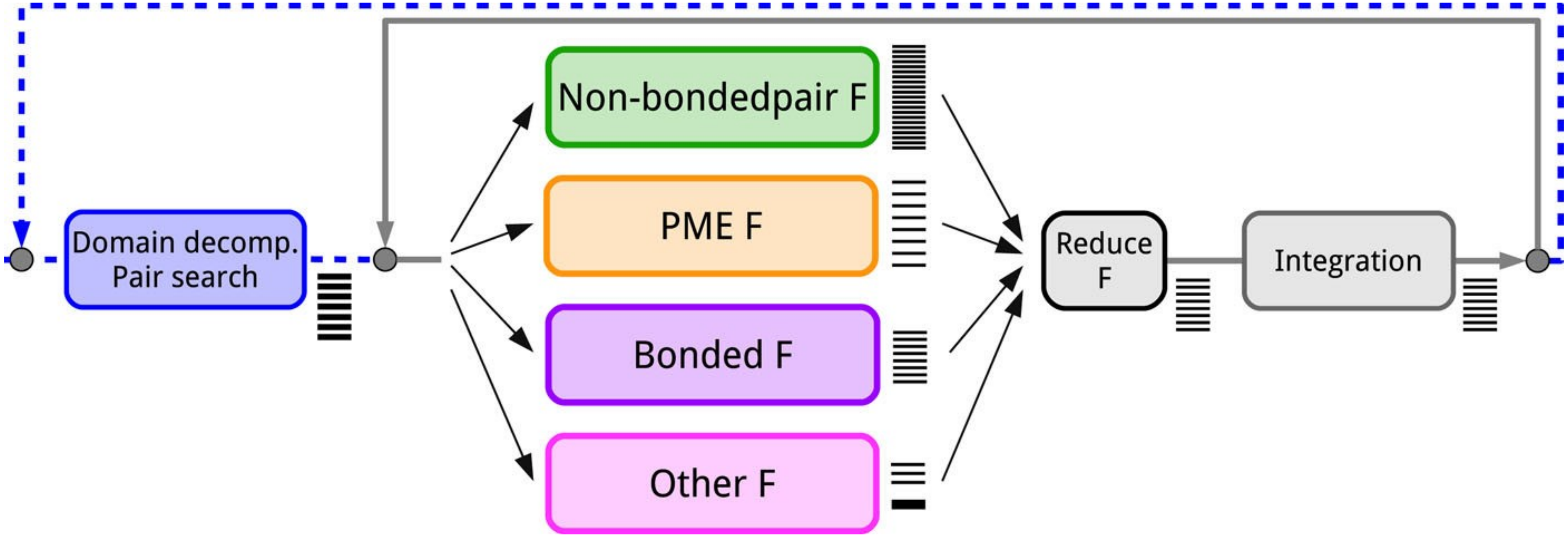
KTH Royal Institute of Technology & SciLifeLab
Stockholm, Sweden

GROMACS

- Open-source molecular dynamics engine
- High-performance for a wide range of modeled systems
  - From 10^4 to 10^9 particles
- … and on a wide range of platforms
  - x86-64, ARM, POWER, RISC-V
  - 11 SIMD backends
  - AMD, Apple, Intel, and NVIDIA GPUs; Intel Xeon Phi
  - Windows, MacOS, BSD, included in many Linux distros

Molecular dynamics: science at 1000 fps

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



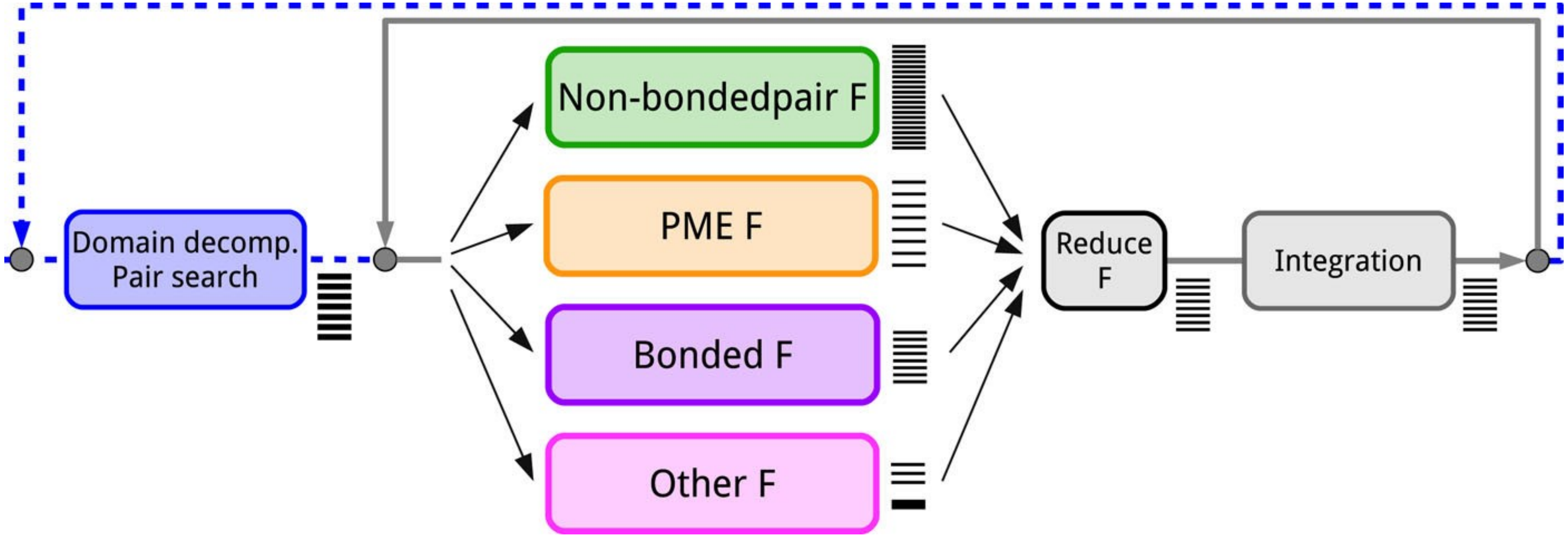
Páll et al., J. Chem. Phys. 153, 134110 (2020)

GROMACS

- Main language: C++17
  - 468k lines of C++ code
  - With a bit of legacy (first release: 1991)
- MPI for inter-node parallelism
- OpenMP for multithreading
- SIMD for low-latency operations on CPU
- GPU offload for high-throughput operations
  - CUDA: NVIDIA GPUs
  - OpenCL: AMD, Apple, Intel, NVIDIA GPUs
  - SYCL: AMD, Intel, NVIDIA GPUs

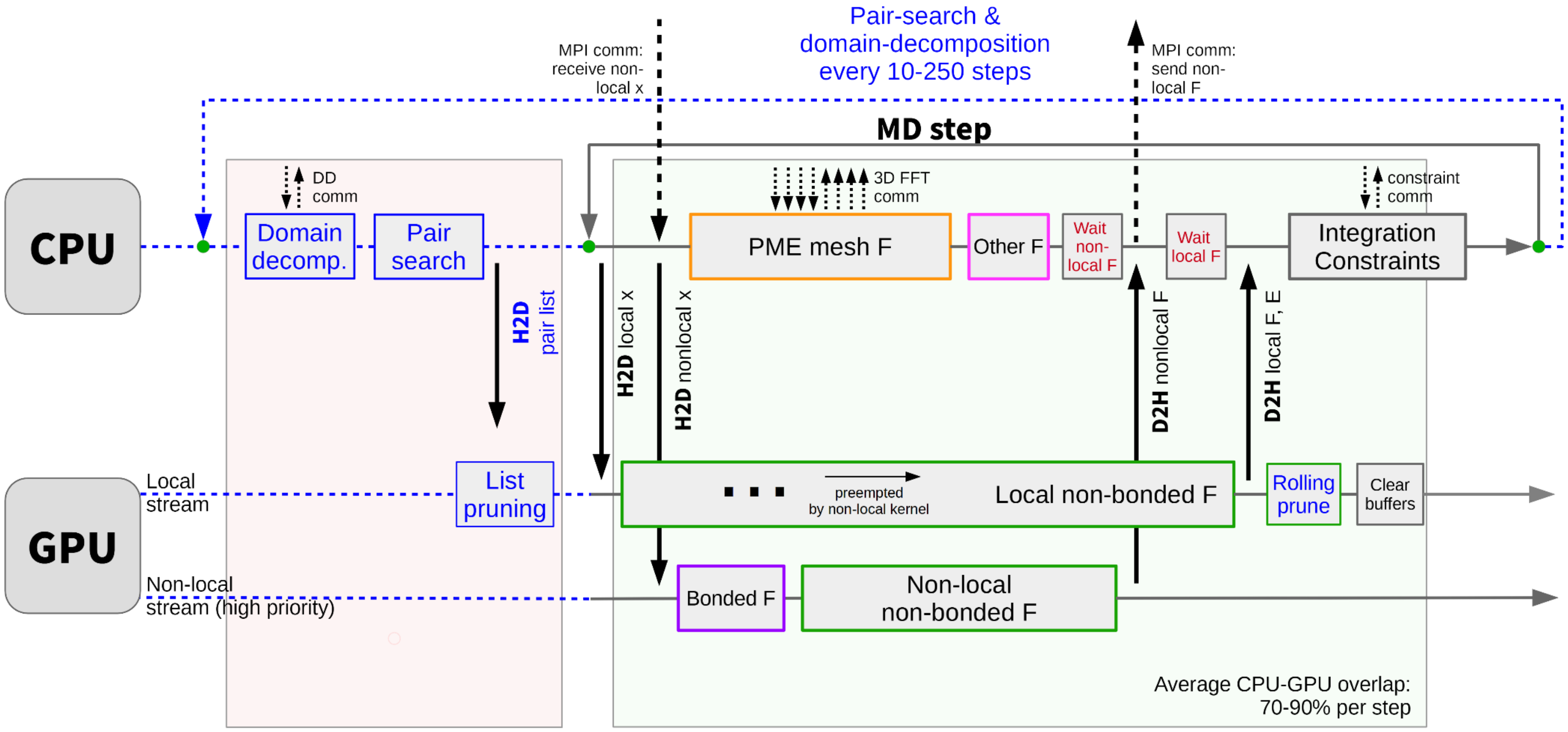
Molecular dynamics: science at 1000 fps

- 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)

Molecular dynamics: science at 1000 fps



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

GROMACS: GPU acceleration evolution

- Designed for CUDA (~2010)
  - Multiple in-order queues
  - Manually-managed USM memory
  - Explicit event-based synchronization
- OpenCL added for AMD GPUs (~2014)
  - With **compatibility layer** to reduce host code duplication
  - Intel GPU support added in ~2018
- SYCL added for Intel GPUs via oneAPI (~2021)
  - Reusing the same compatibility layer
  - Later to hipSYCL/AdaptiveCpp and NVIDIA and AMD GPUs (~2022)

Supported hardware

- Primary targets:
  - AMD CDNA2 GPUs with AdaptiveCpp/hipSYCL
  - Intel Xe-HPC GPUs with oneAPI
- Secondary targets:
  - Other AMD GPUs with oneAPI and AdaptiveCpp
  - Other Intel GPUs with oneAPI
- Should work:
  - NVIDIA GPUs with oneAPI and AdaptiveCpp

Shoehorning SYCL into legacy code

- Prior to SYCL2020: only buffers/accessors
  - Wrapper to use buffers *à la* USM
- With SYCL2020: USM-only
  - Identical code logic with CUDA/OpenCL
  - Accessors lead to register spills
  - Have the data copies and multiple streams anyway
- Only difference: need context for pinned host allocations

Shoehorning SYCL into legacy code

CUDA: Barrier-like events; no such thing in SYCL standard

1. Refactor code to use returned events
2. Use extensions
   - ext_oneapi_submit_barrier, HIPSYCL_EXT_ENQUEUE_CUSTOM_OPERATION
   - Better mapping to underlying CUDA/HIP model
   - Reduced launch cost
     - AdaptiveCpp: HIPSYCL_EXT_COARSE_GRAINED_EVENTS
     - oneAPI: waiting for https://github.com/KhronosGroup/SYCL-Docs/issues/404
3. Track last returned event in our queue wrapper
   - Might be needed for oneAPI's Immediate command lists / SYCL Graphs

Shoehorning SYCL into legacy code

- External libraries: FFT, GPU-aware MPI
- Native SYCL libraries only available for Intel hardware:
  - IntelMPI; oneMKL, BBFFT
- With USM, native ones work fine via interop!
  - OpenMPI, MPICH; cuFFT, rocFFT, heFFTe, vkFFT
- Required effort: CMake and thin wrapper/launcher
  - A lot duplication for FFT

Performance-portability challenges

- It's all LLVM, so not far from native, but
- Performance tuning with software stack in flux
- Kernel performance is *usually* not a huge problem
  - AoS→SoA to improve register allocation on Xe-HPC
    - 2.4x performance improvement for some kernel flavors
    - No impact on CDNA (gfx908)
  - On CDNA2 with ROCm 5.3: ~10% performance loss with SoA
  - ROCm 5.5: further ~10% loss
  - SoA→AoS resolves most (not all) regressions with ROCm 5.5

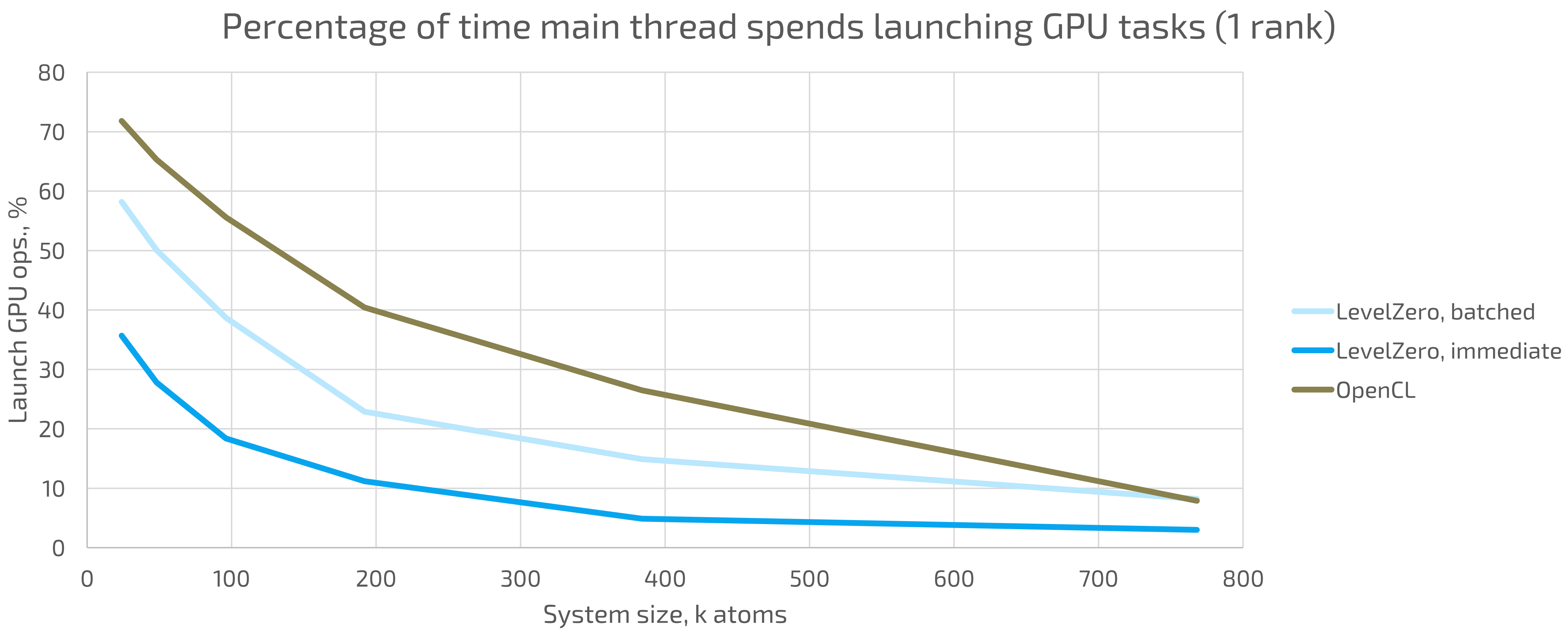
Launch latency challenges

- MD is often about strong scaling

- Compute and data transfers get faster
- Latency stays ~the same

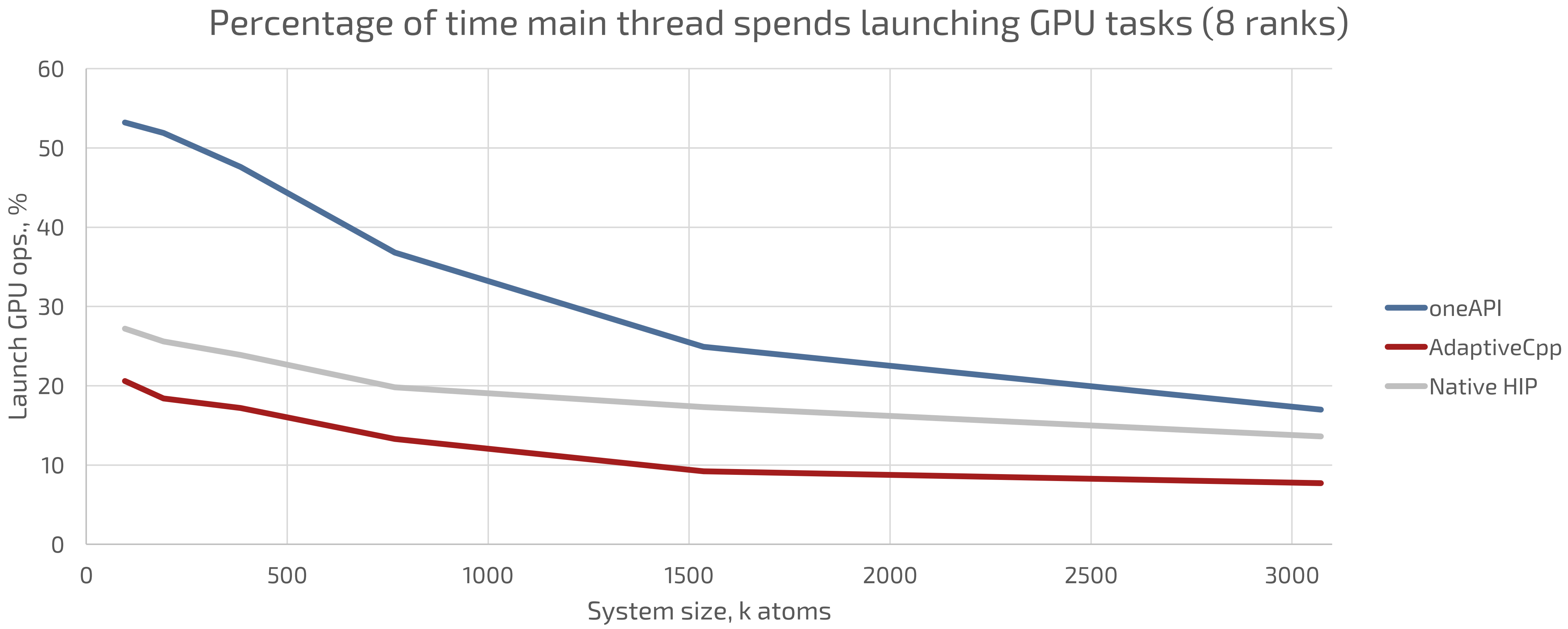
- Communications often on the critical path
- CPU can do useful work too

Launch overheads



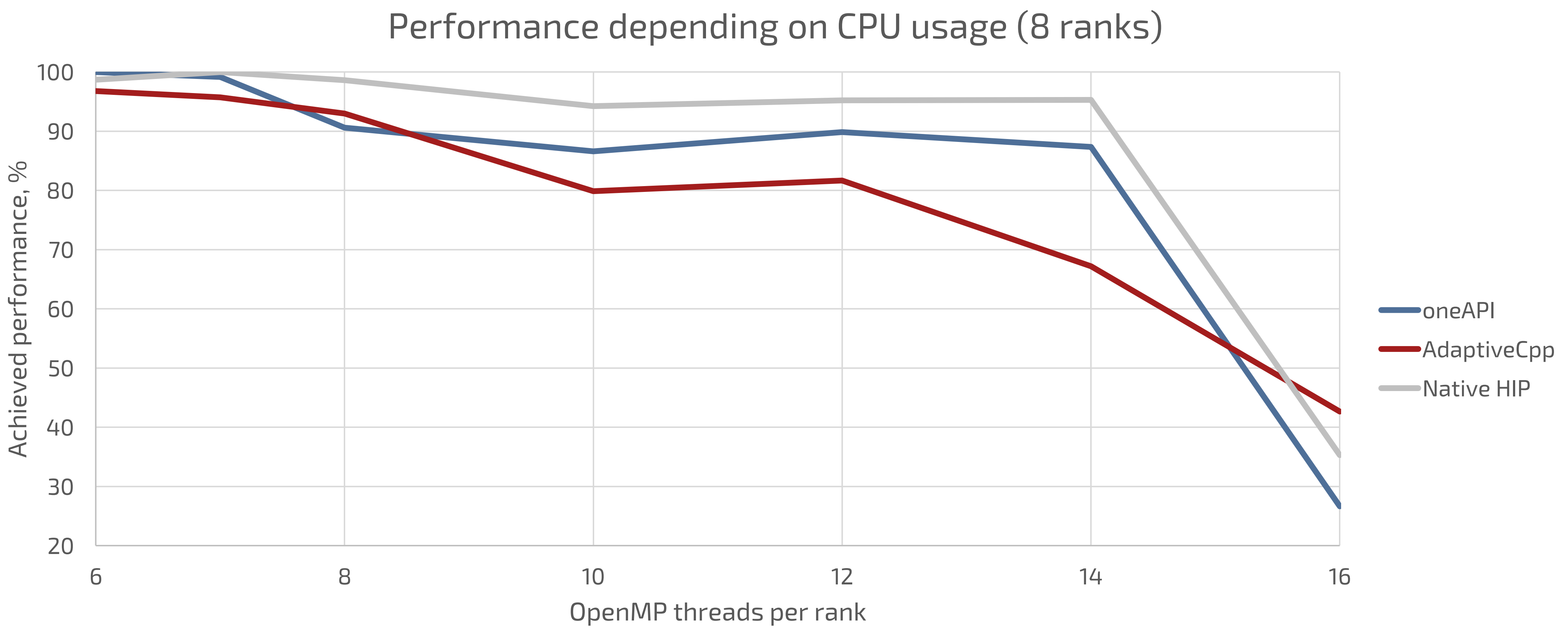
Intel Xeon 8480+, 1xPVC 1100, oneAPI 2023.2, PME grappa box, -ntomp 6 -bonded gpu

Launch overheads



AMD Epyc 7A53, 8xMI250X, ROCm 5.3.3, RF grappa box, -ntomp 6 -bonded gpu

Launch overheads



AMD Epyc 7A53, 1xMI250X, ROCm 5.3.3, 384k RF grappa box, -bonded cpu

Launch overheads

- Know all your runtimes!

- Runtime optimizations to avoid extra API calls
- Runtime optimizations in general
- Using extensions to give hint to the runtime
- Changing scheduling code to minimize GPU APIs on critical path
- JITting all relevant kernel flavors
- Submitting all at once: SYCL Graphs

Performance-portability challenges

- Consistent maintenance with software stack in flux
- New developments of both AdaptiveCpp and oneAPI
  - Supported standard features, deprecations, defaults, environment variables…
  - But at least it'
- Especially when relying on extensions
  - E.g.: oneAPI queue priority extension: just what we need!
    - Supported only with LevelZero backend, not hardware support
- CMake / build infrastructure
- User-facing documentation

Performance-portability benefits

SYCL vs. vendor-specific APIs:

- Reduced code duplication
  - Most code is not device-specific
- Easier feature enablement
- Different sanitizers and profilers

Performance-portability benefits

Multiple runtimes:

- Keeping the code standard compliant
  - Future-proofing
  - Finding runtime bugs
- Having the choice when thing don’t work as planned
- Stress-testing the scheduling code
  - Dealing with inefficient is never fun, but it has upsides
  - Found synchronization bugs in both SYCL and CUDA backends

Looking into the future

- Continuing collaboration with oneAPI and AdaptiveCpp teams
- SYCL Graphs
- Refactoring for shared memory architectures
- GPU-initiated MPI
- Stream-aware MPI

- Performance-portability with SYCL does not come for free, but it works, production-ready, and the benefits are worth it!

Acknowledgements

- Intel Corporation
- AMD for sharing their HIP port
- Mark Abraham, Heinrich Bockhorst, Roland Schulz (Intel)
- Aksel Alpay (Heidelberg University Computing Centre)
- GROMACS dev team, especially Szilárd Páll

- **Get in touch: andreyal@kth.se**