The 11th International workshop on OpenCL and SYCL

IWOCL & SYCLcon 2023

# Comparing the Performance of SYCL Runtimes for Molecular Dynamics Applications

Andrey Alekseenko, SciLifeLab, KTH

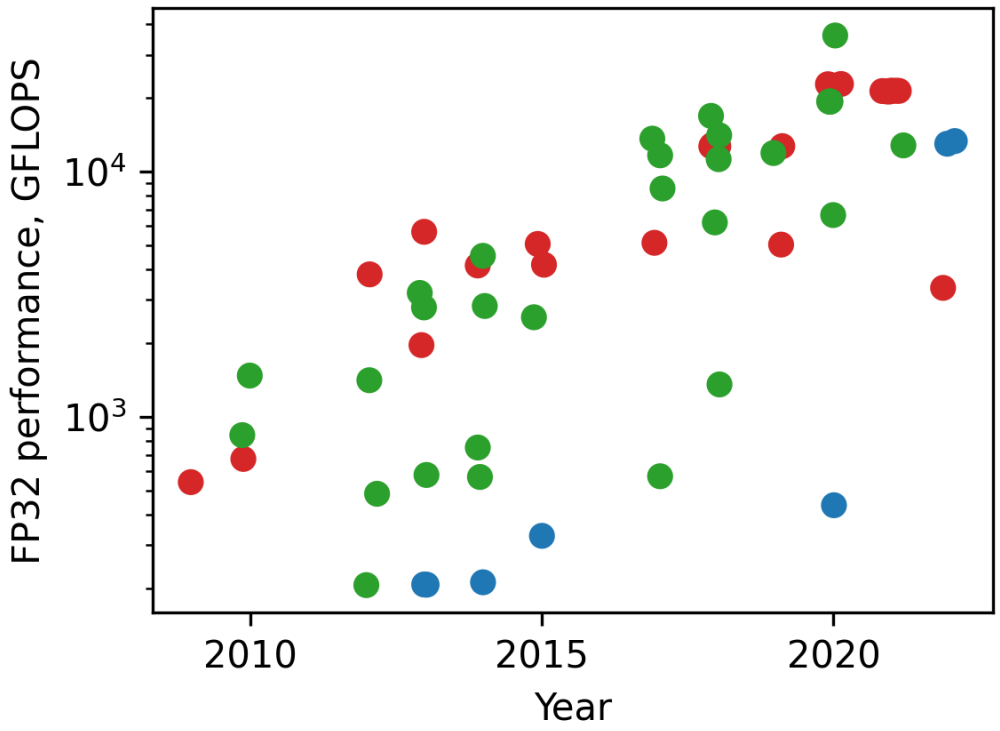
Szilárd Páll (PDC Center for High Performance Computing, KTH)

April 18–20, 2023 | University of Cambridge, UK

iwocl.org

# GPU scheduling challenges

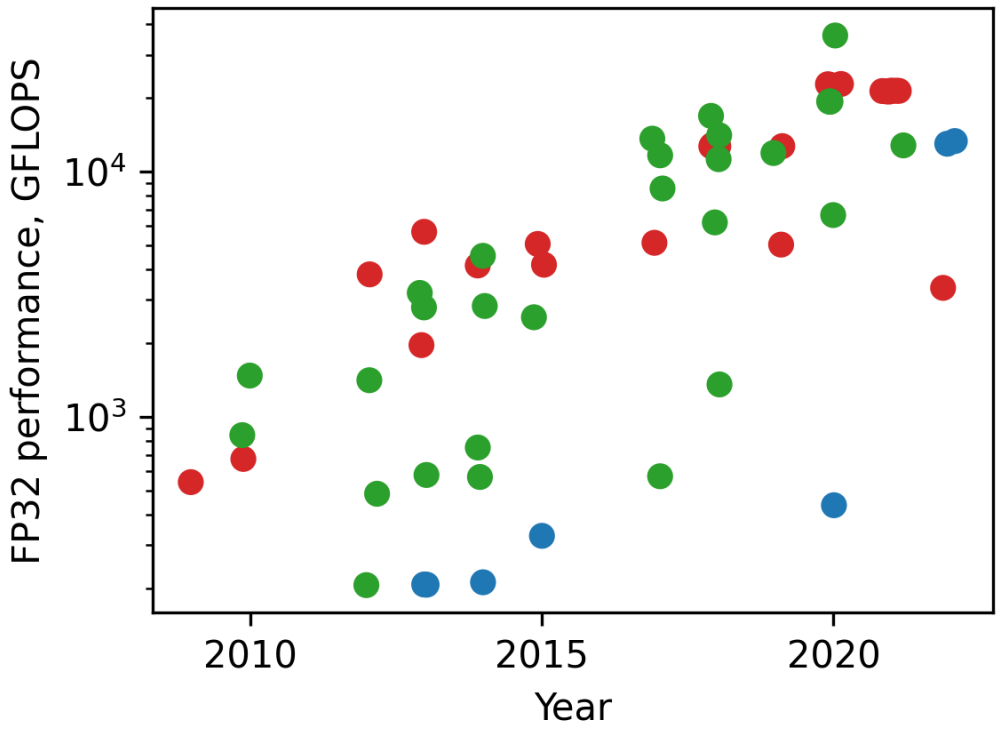
- Kernels get faster

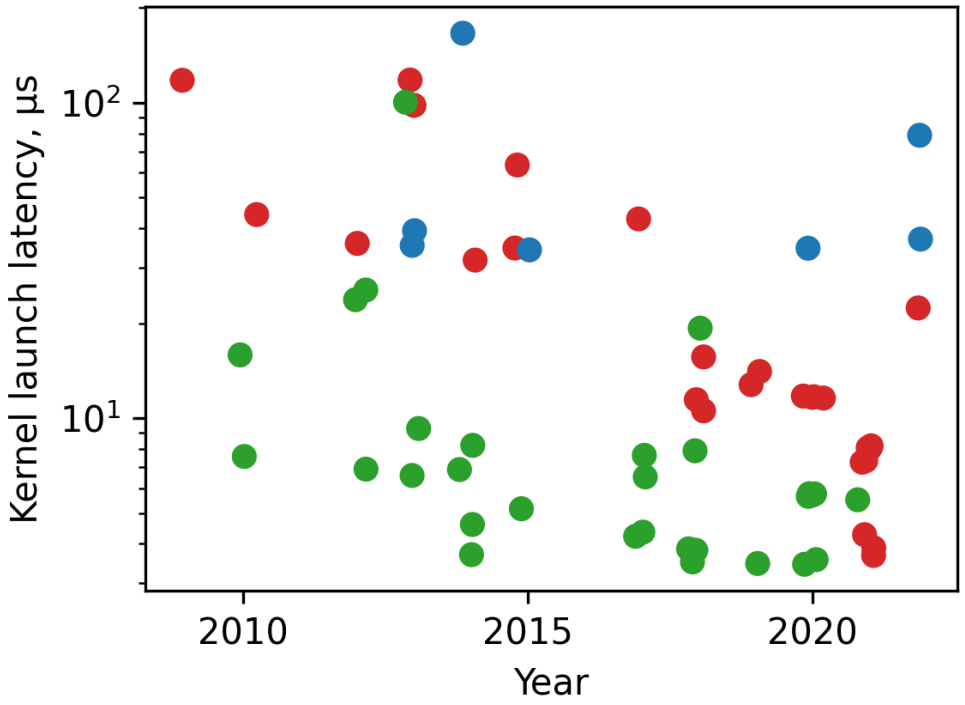



Based on https://github.com/krrishnarraj/clpeak

# GPU scheduling challenges

- Kernels get faster, latency stays the same

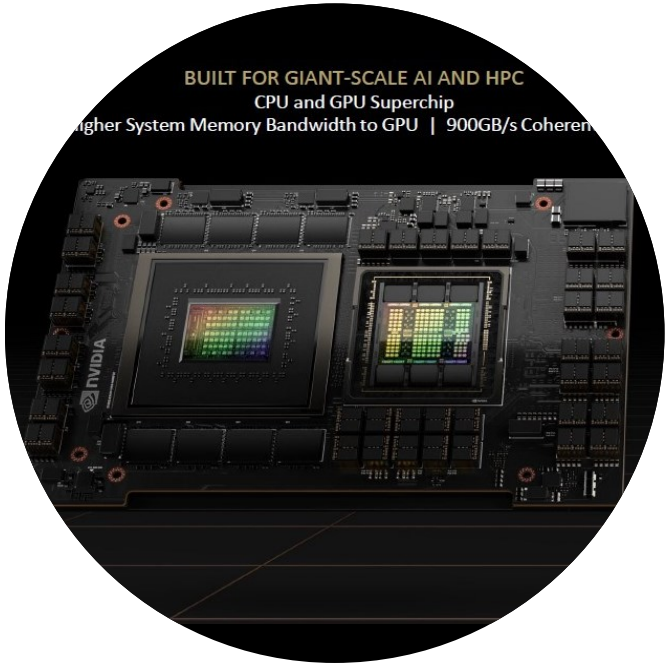


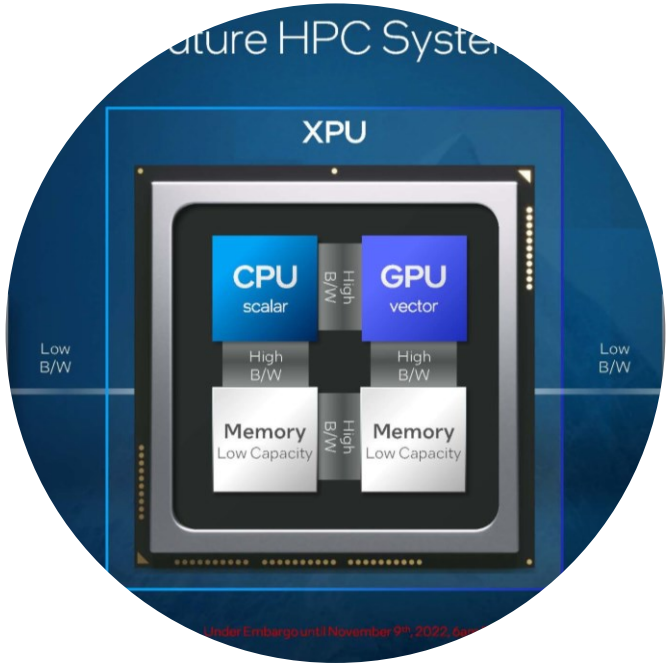



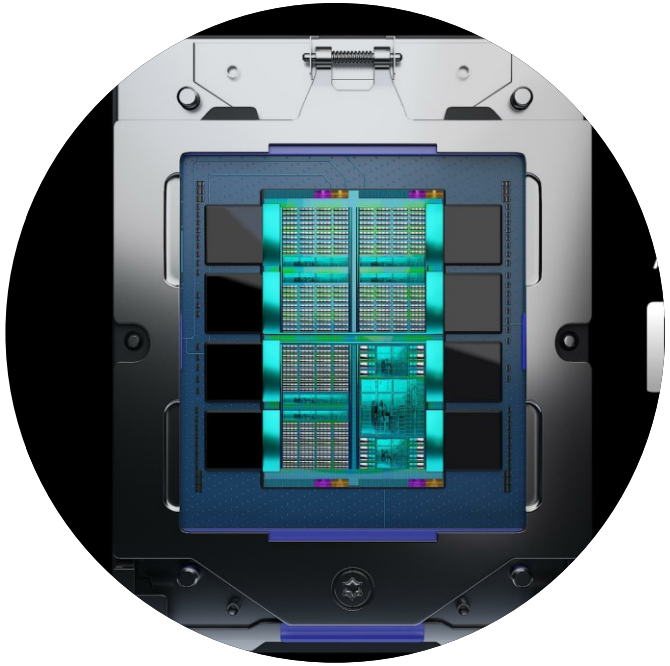

Based on https://github.com/krrishnarraj/clpeak

# GPU scheduling challenges

- Kernels get faster, latency stays the same
- HW with truly unified memory



# GPU scheduling challenges

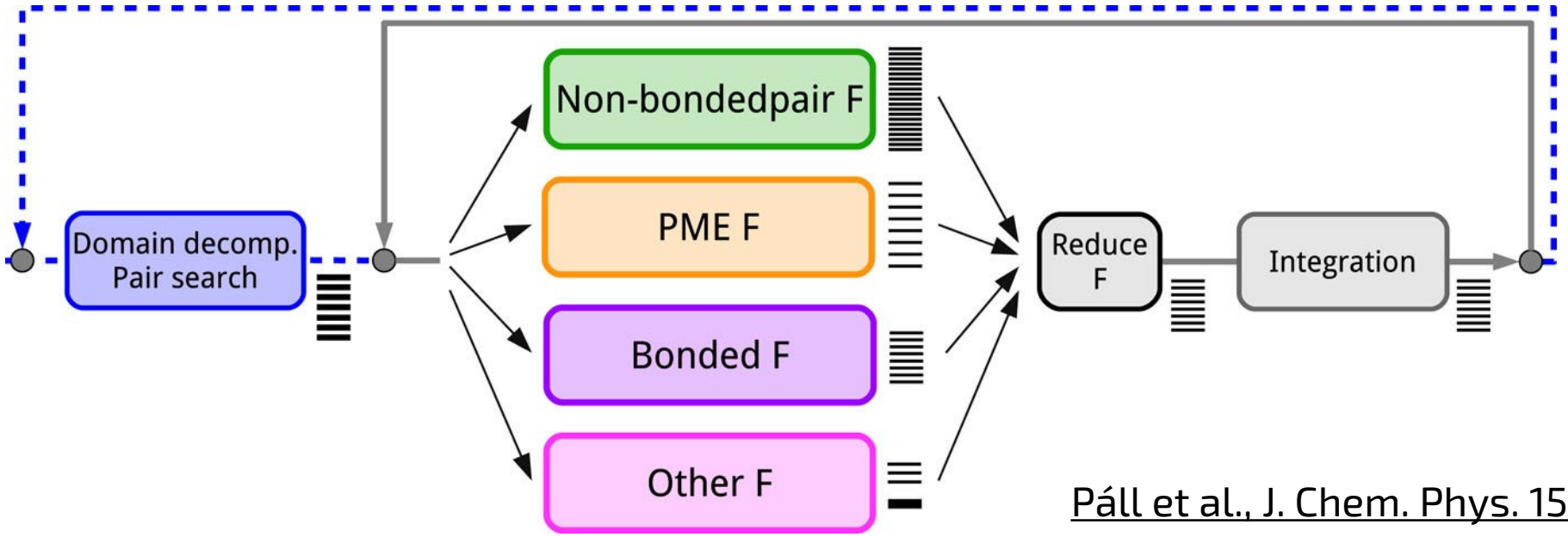
- Kernels get faster, latency stays the same
- HW with truly unified memory

- Compute gets cheaper
- Data transfer gets (much) cheaper
- Fixed cost for offloading

- CPUs can do useful work too!

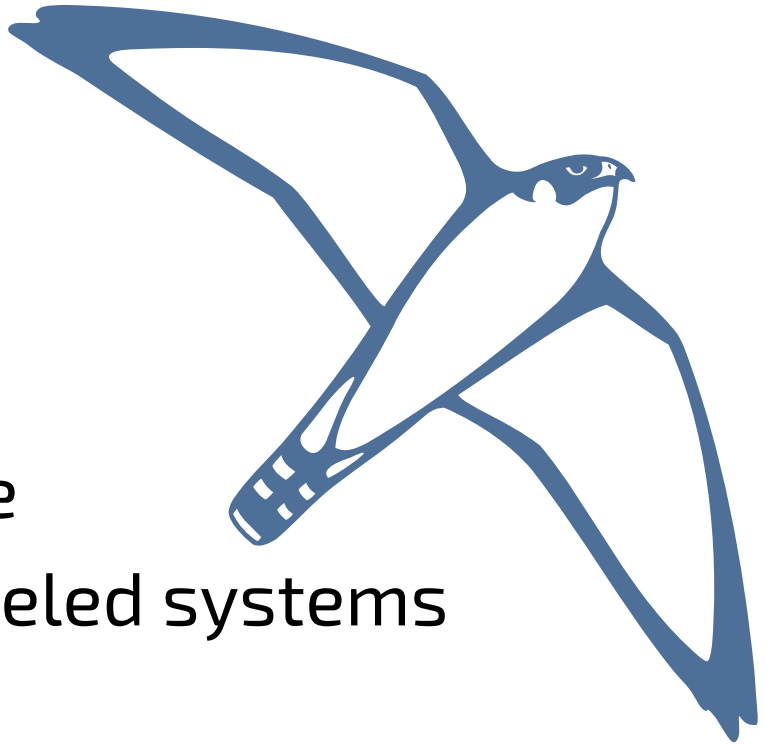
# Molecular dynamics: science at 1000 fps

- 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



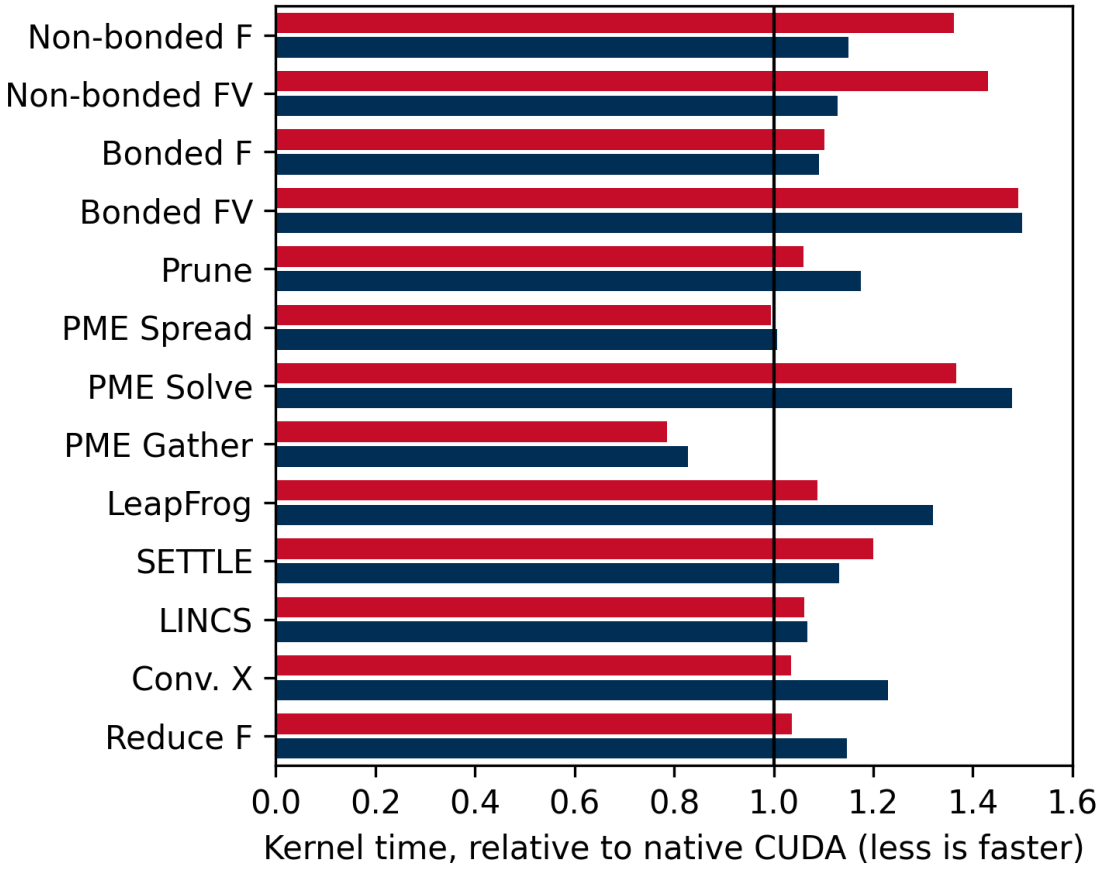
- Open-source molecular dynamics engine
- One of the most used HPC codes worldwide
- High-performance for a wide range of modeled systems
- … and on a wide range of platforms
  - CUDA: NVIDIA
  - OpenCL: AMD, Apple, Intel, NVIDIA
  - SYCL: AMD, Intel, NVIDIA

# GROMACS: GPU acceleration layer

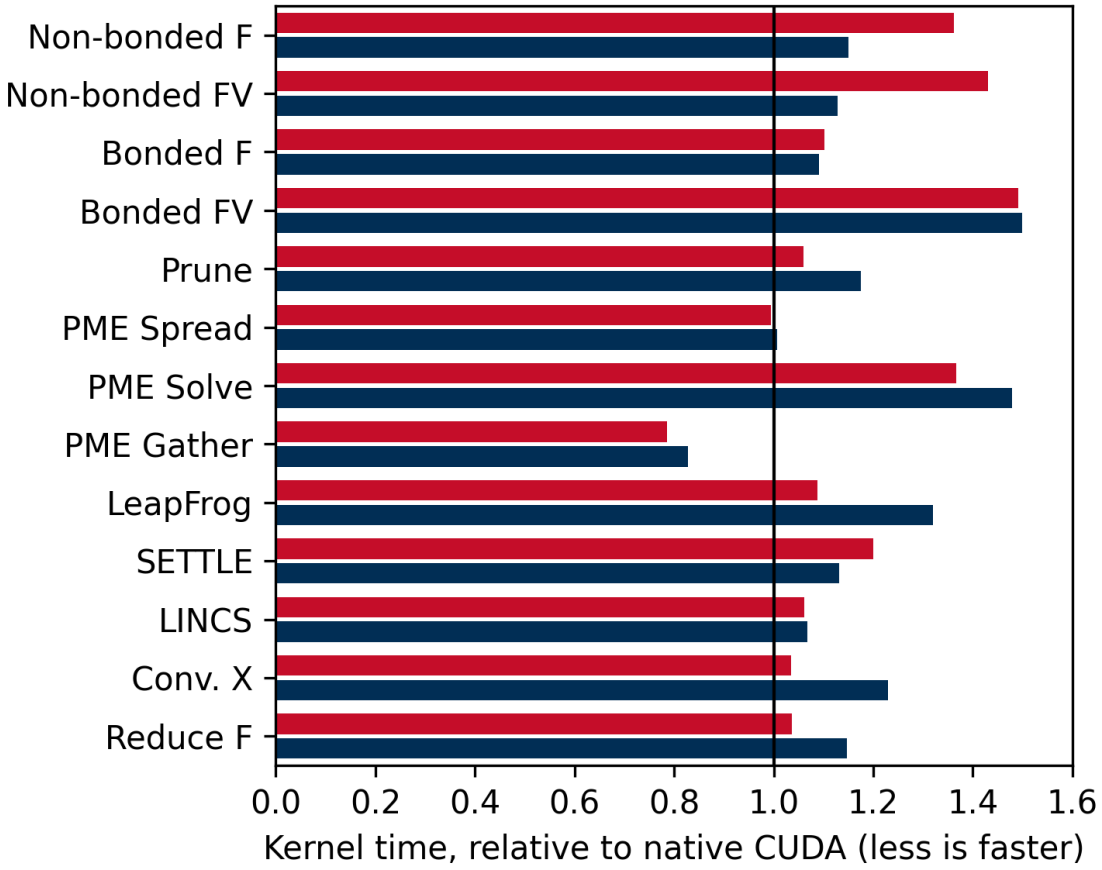
- Designed for CUDA
  - Multiple in-order queues
  - Manually-managed USM memory
  - Explicit event-based synchronization

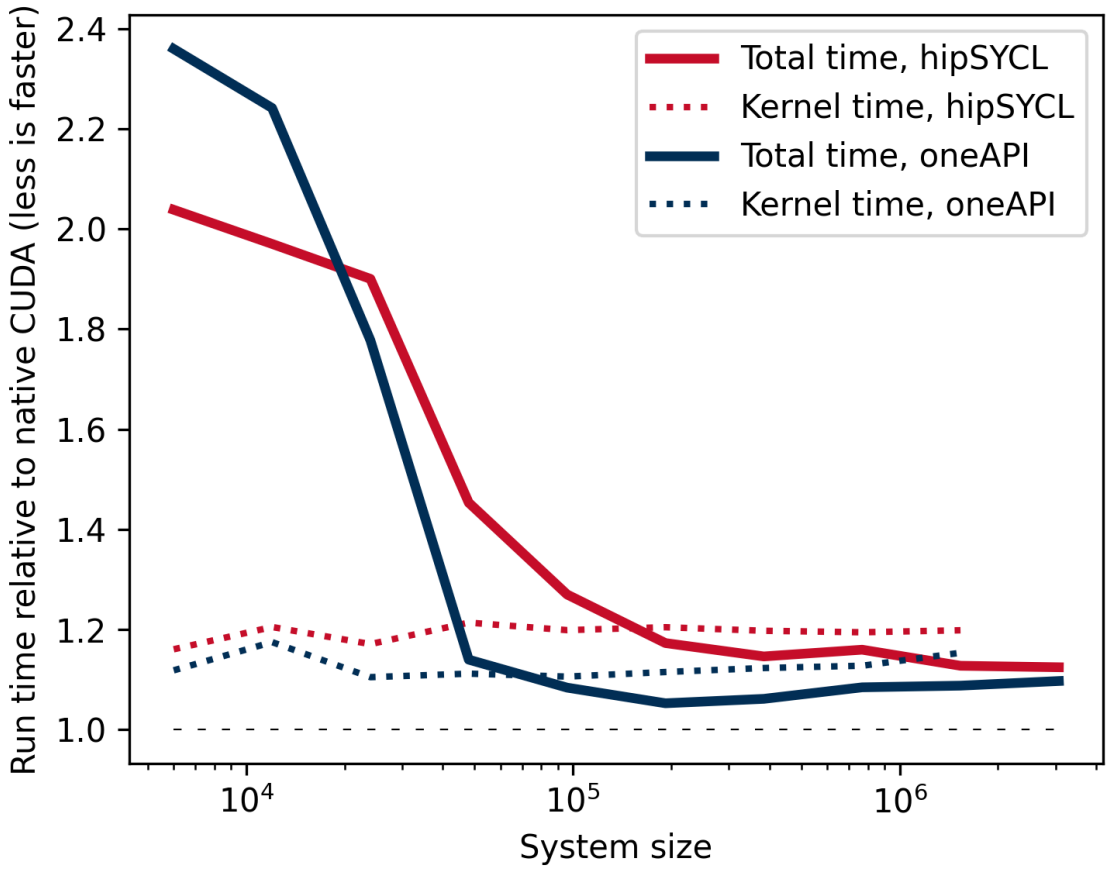
- oneAPI: Intel, AMD, NVIDIA GPUs
- hipSYCL: AMD and NVIDIA GPUs

# Performance: NVIDIA V100

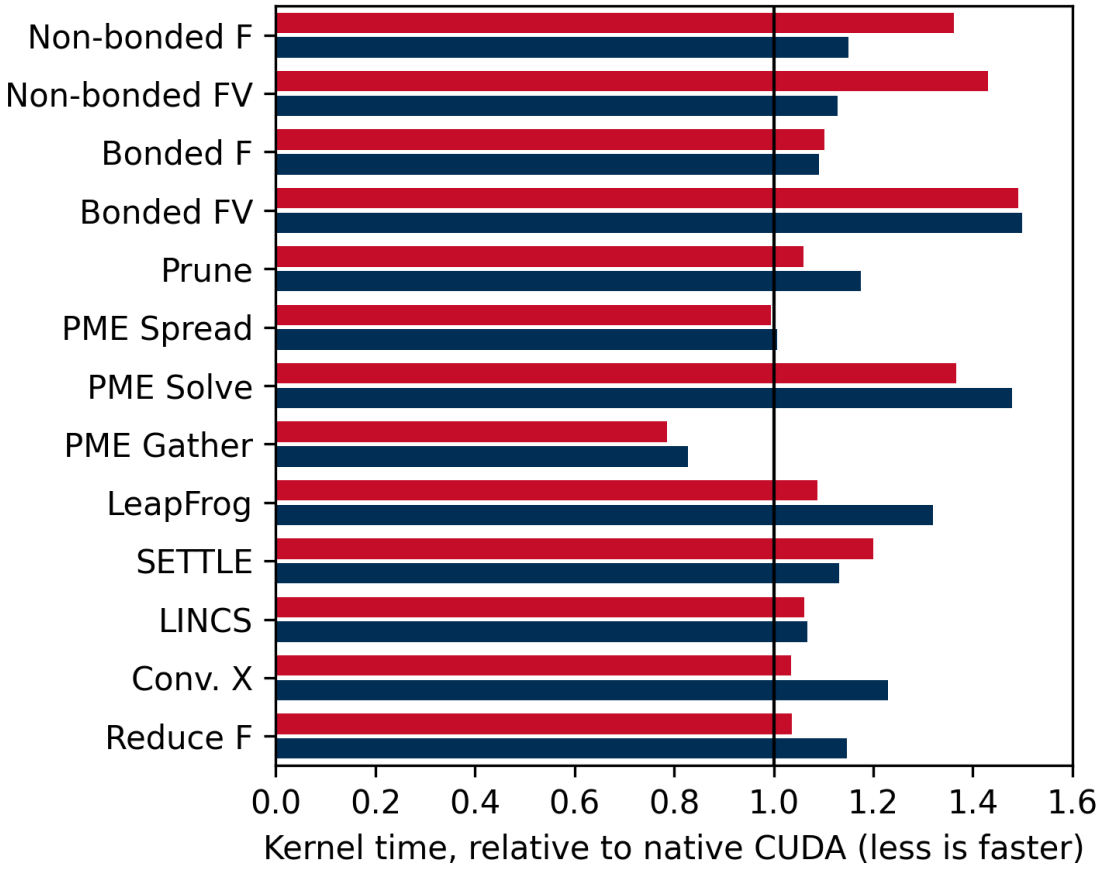


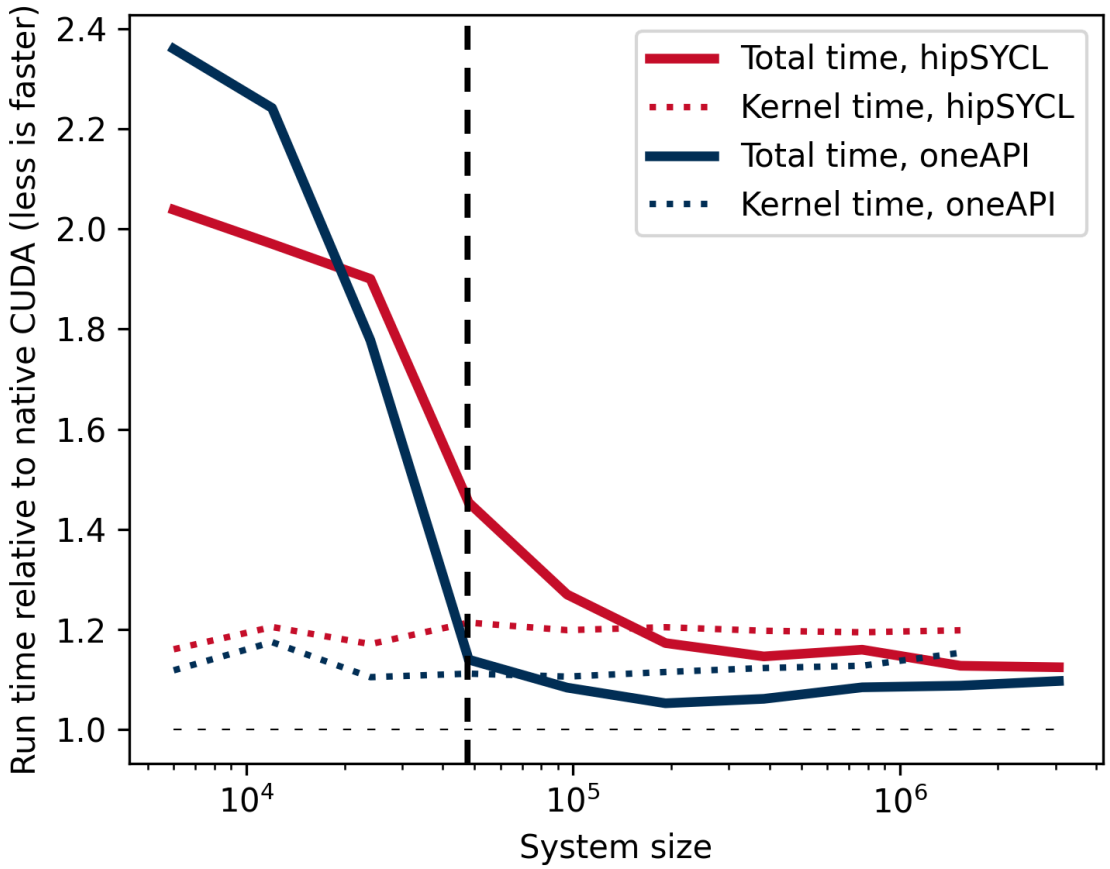
# Performance: NVIDIA V100



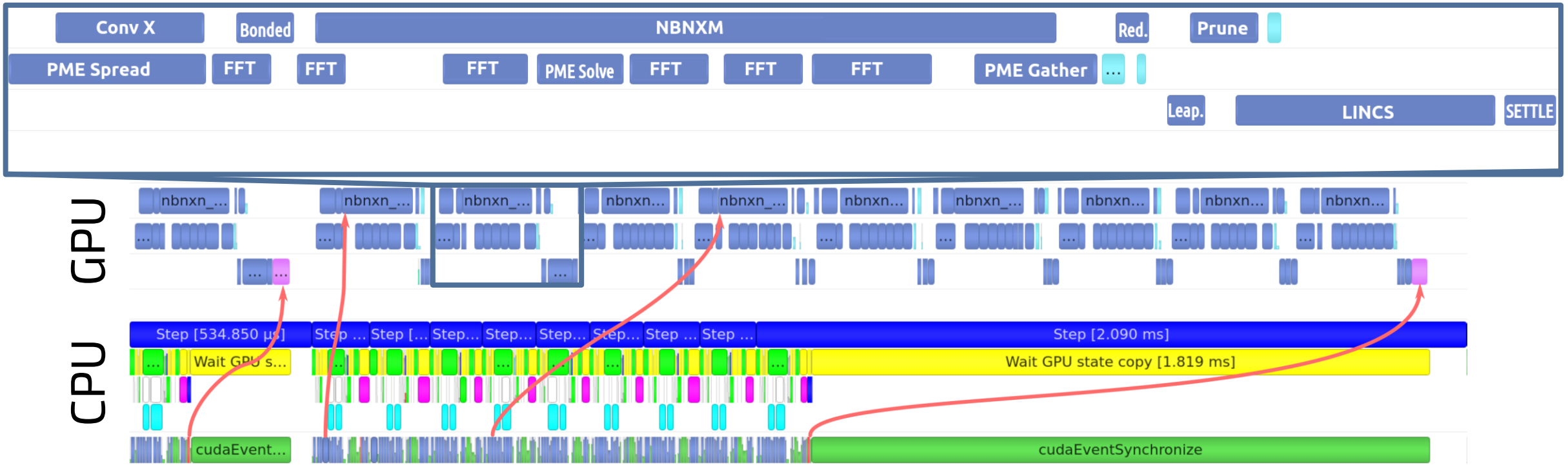
# Performance: NVIDIA V100



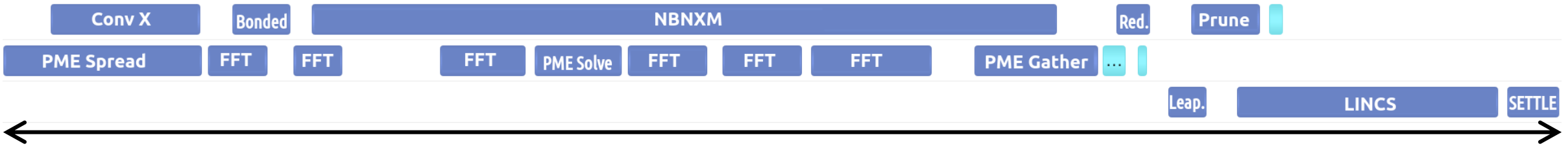



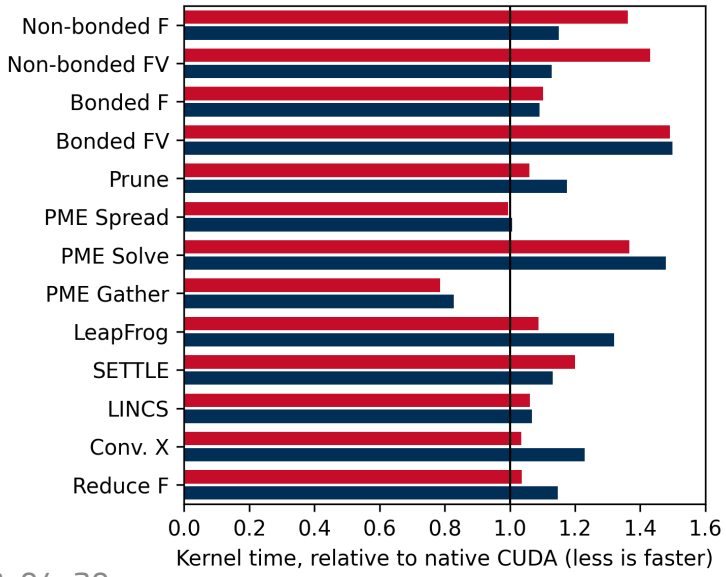
# GPU-resident run: CUDA

# GPU-resident run: CUDA

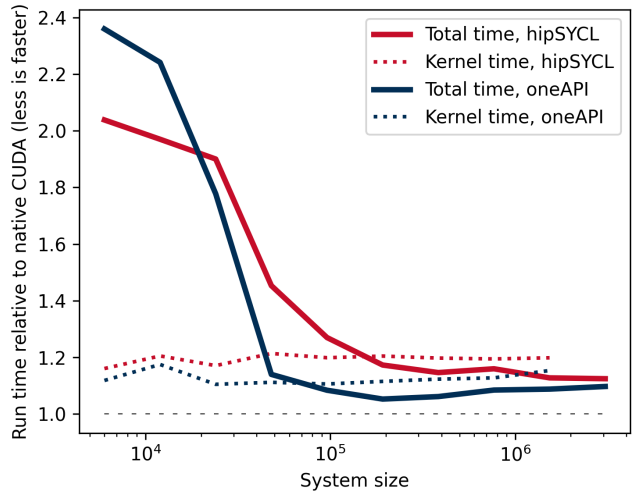


# Performance: GPU work



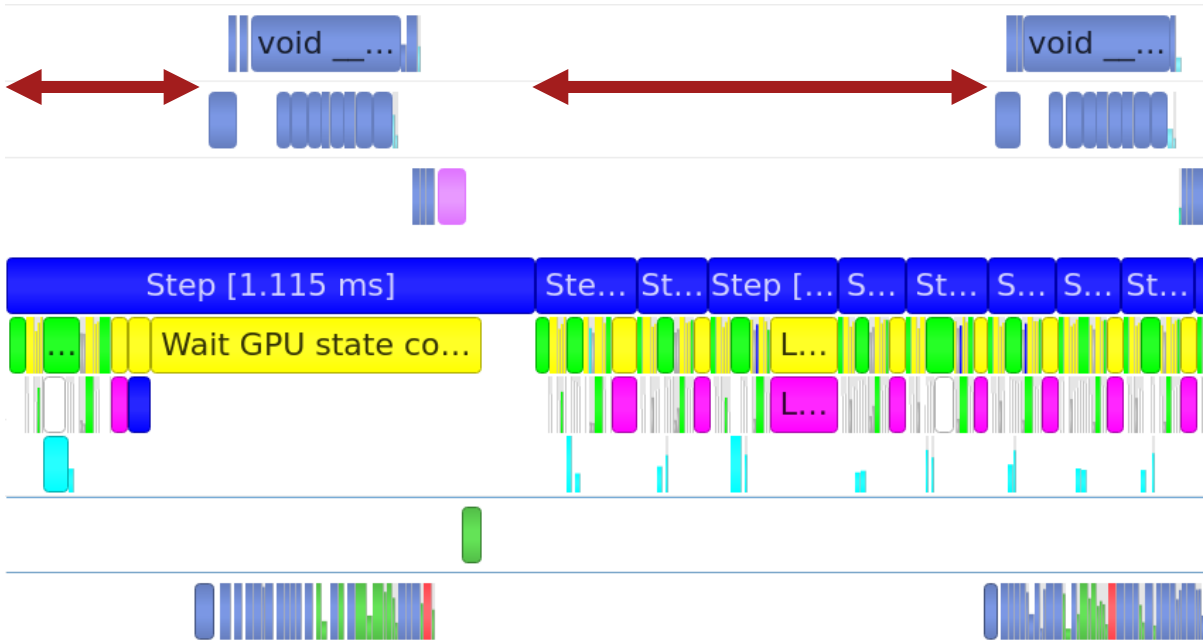

| | |
|---|---|
| CUDA native | ~400 µs |
| hipSYCL | ~440 µs |
| oneAPI | ~440 µs |



# Performance: CUDA vs. hipSYCL

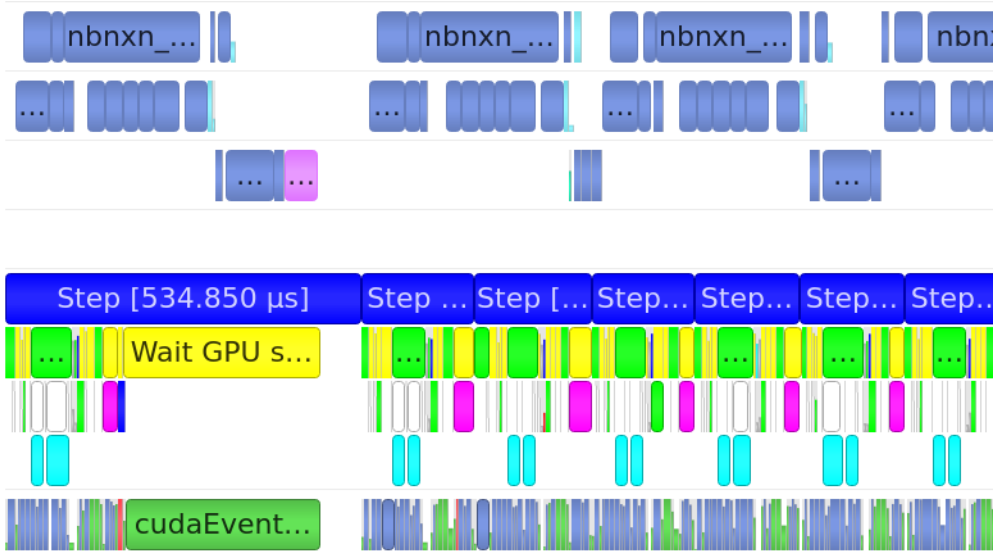
**CUDA: 3.9 ms per 10 steps**


**hipSYCL: 6.1 ms per 10 steps**



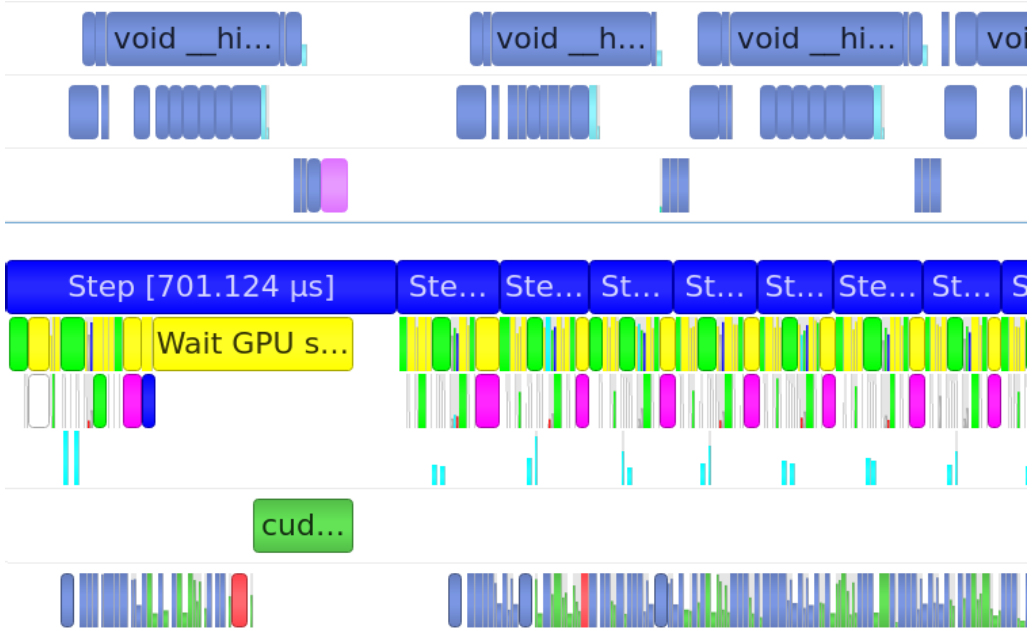
# Performance: CUDA vs. hipSYCL

**CUDA: 3.9 ms per 10 steps**

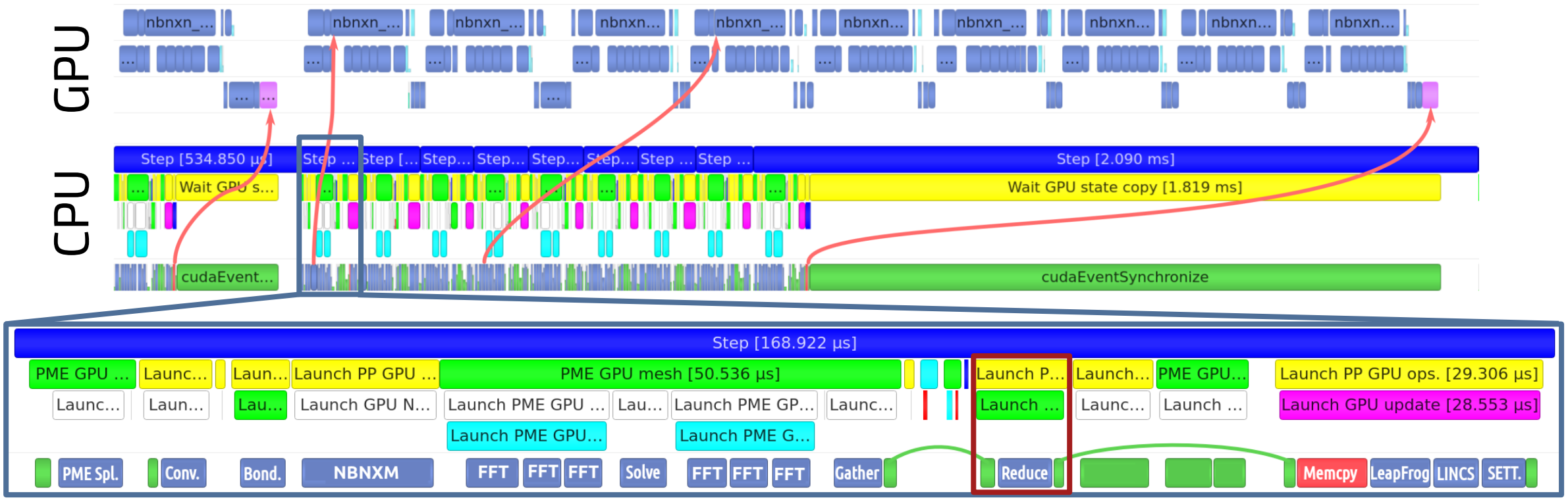



**hipSYCL: ~~6.1~~ 5.0 ms per 10 steps**

HIPSYCL_RT_MAX_CACHED_NODES=1

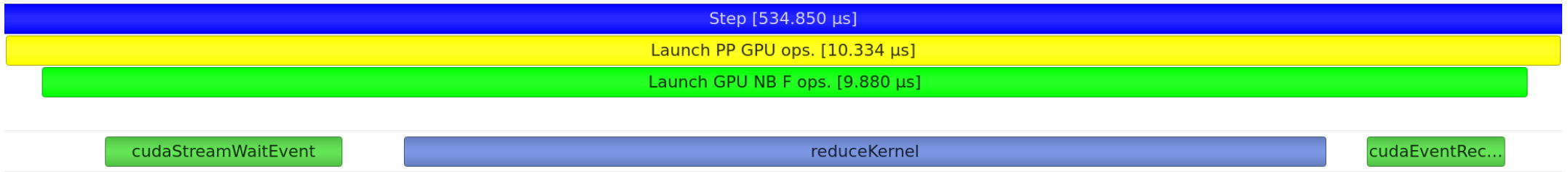


# GPU-resident run: CUDA

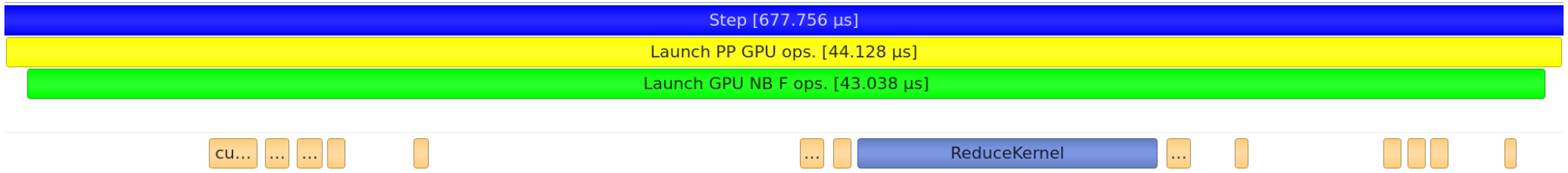


# Performance: CUDA vs. oneAPI

## CUDA: 10.3 µs

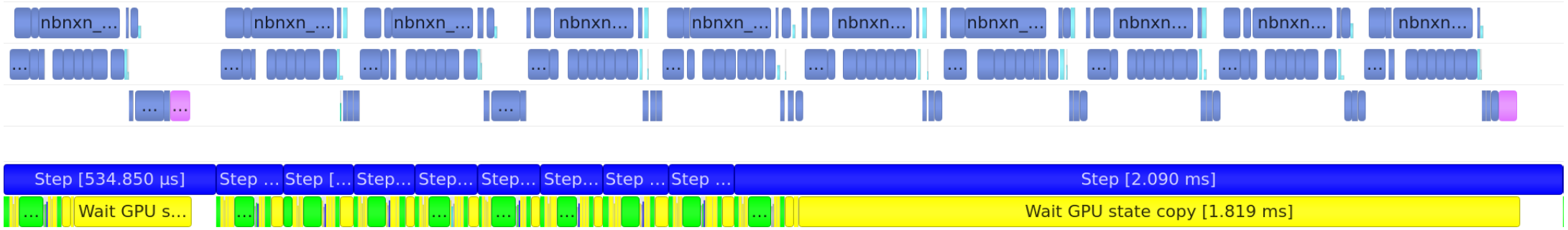



## oneAPI: 44.1 µs

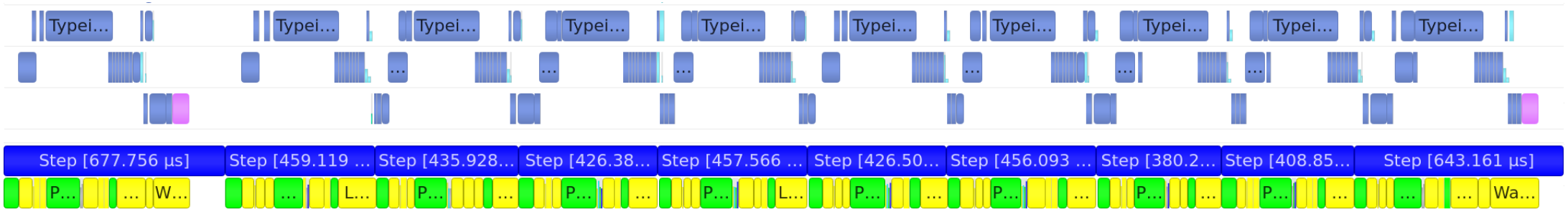


# Performance: CUDA vs. oneAPI

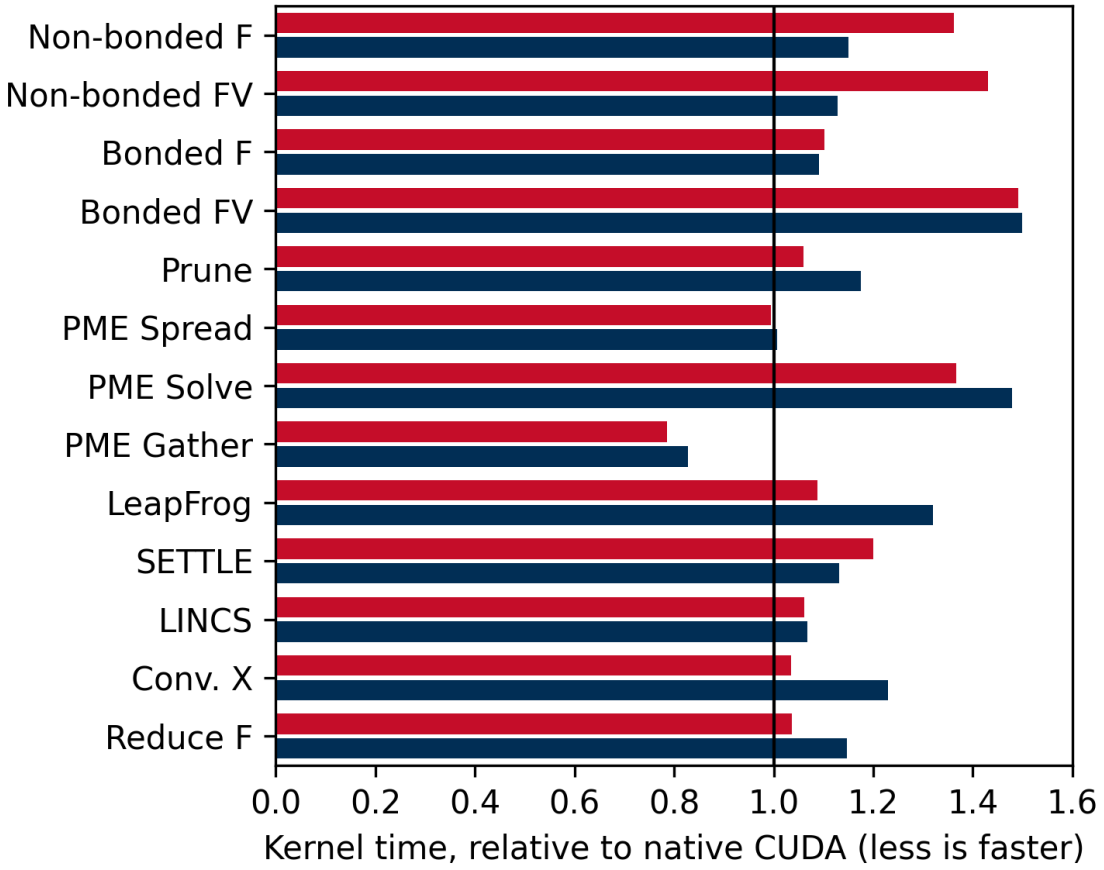
## CUDA: 3.9 ms per 10 steps, 1.8 ms waiting

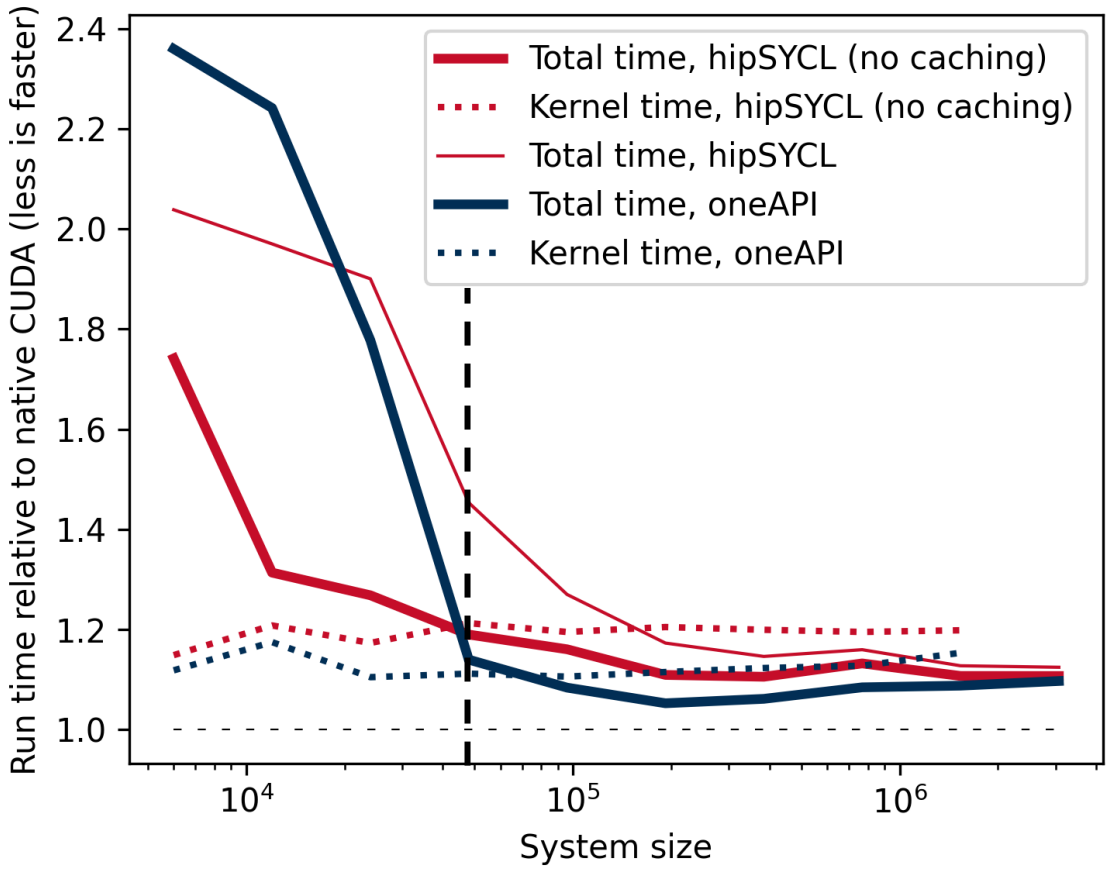



## oneAPI: 4.8 ms per 10 steps, 0.1 ms waiting

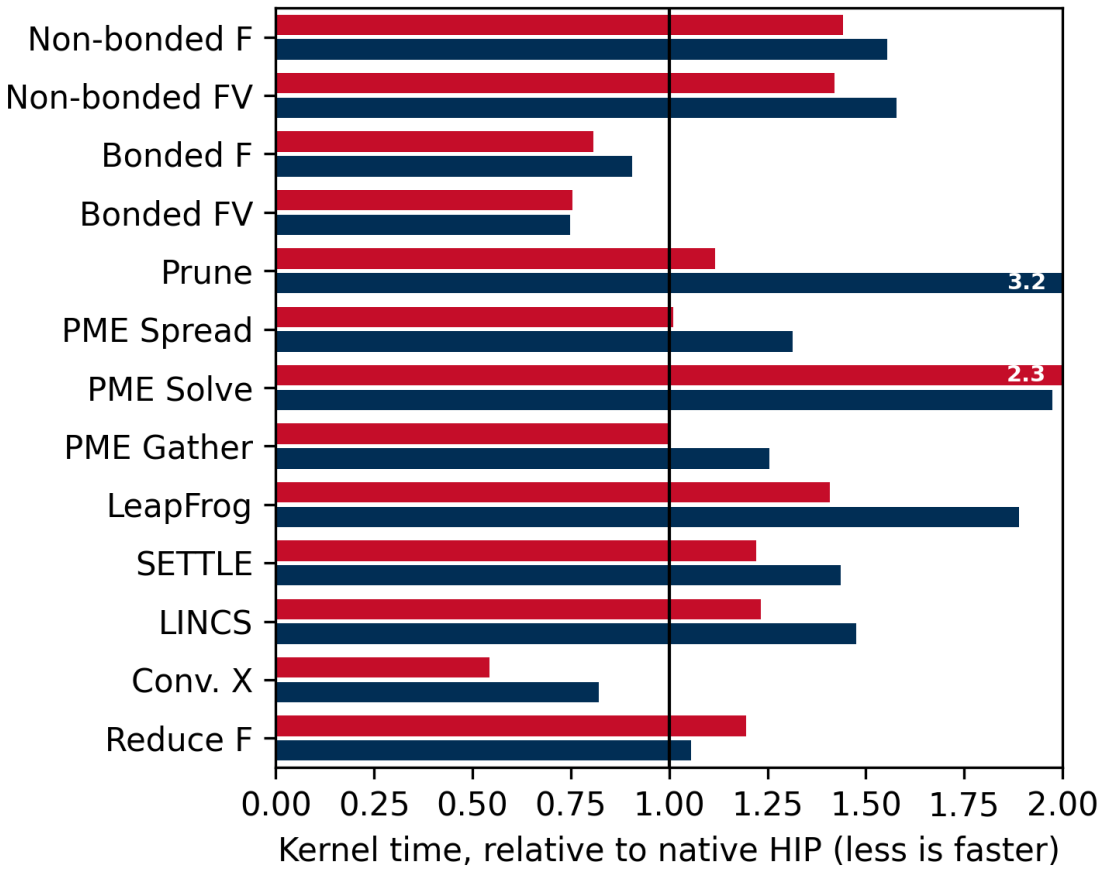


# Performance: NVIDIA V100

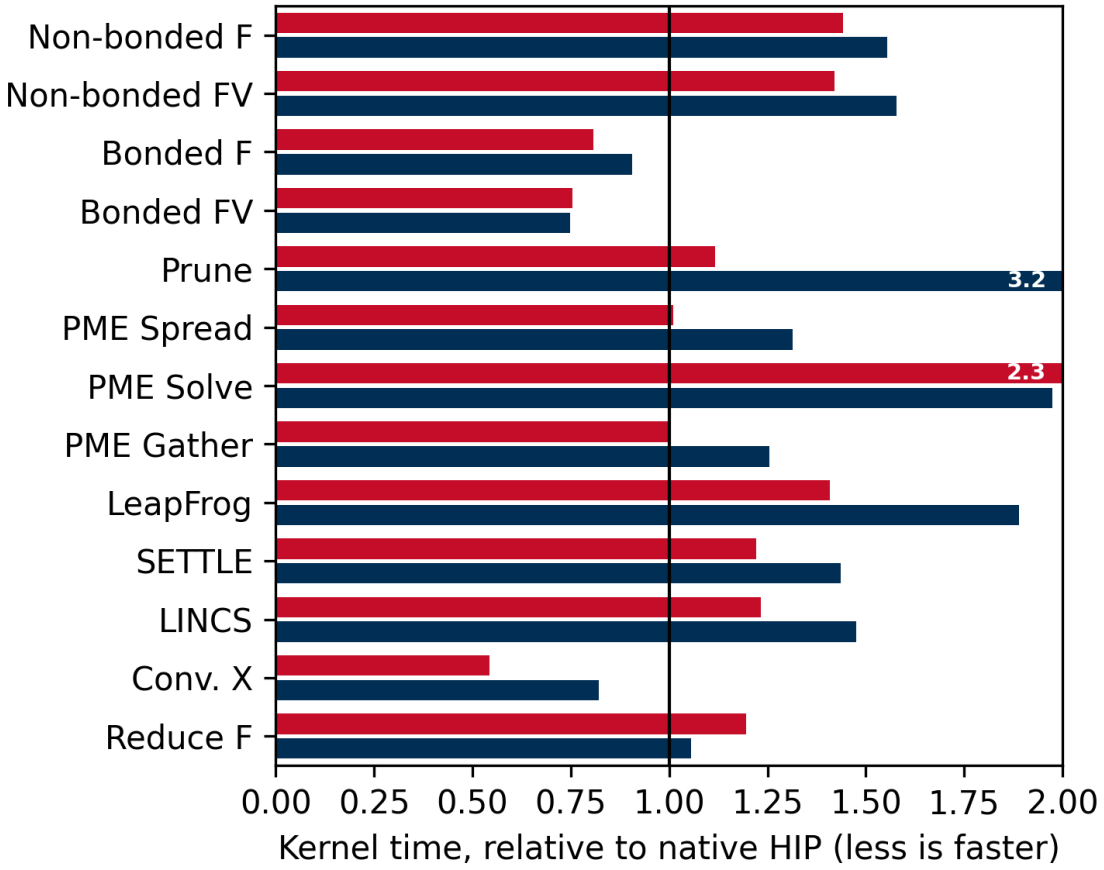


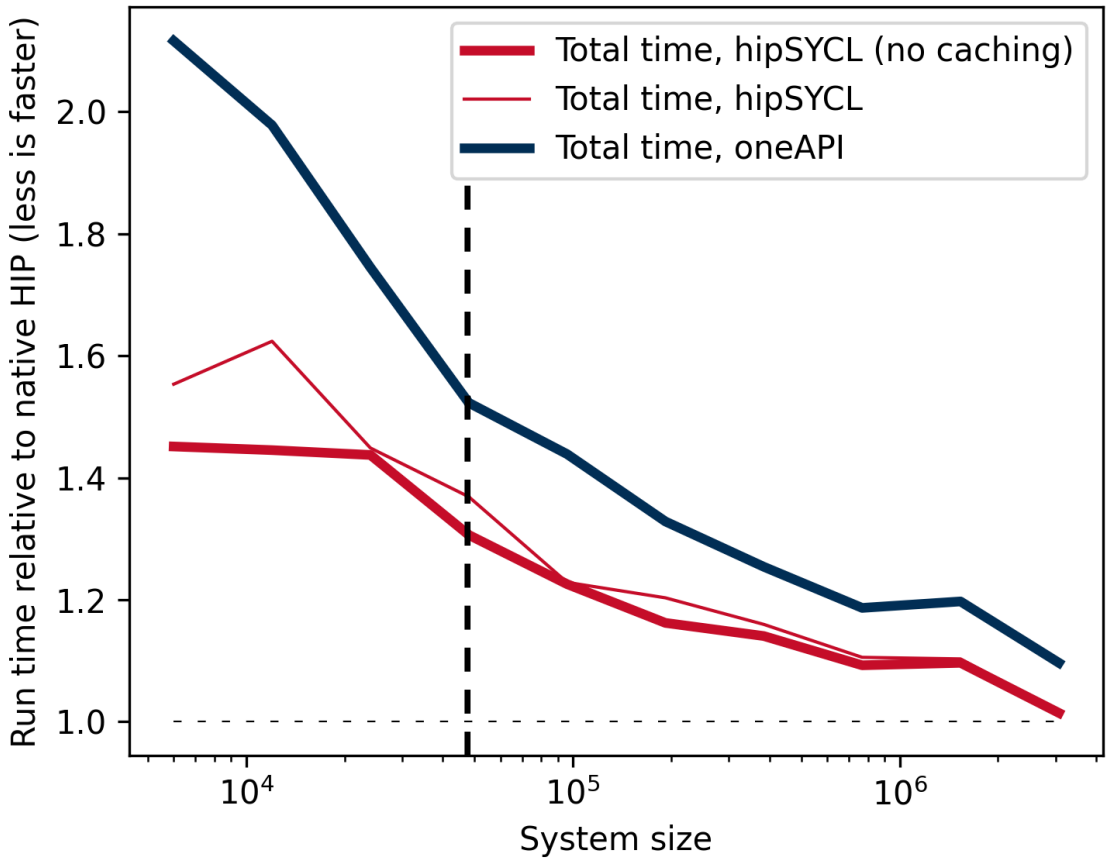



# Performance: AMD MI250X

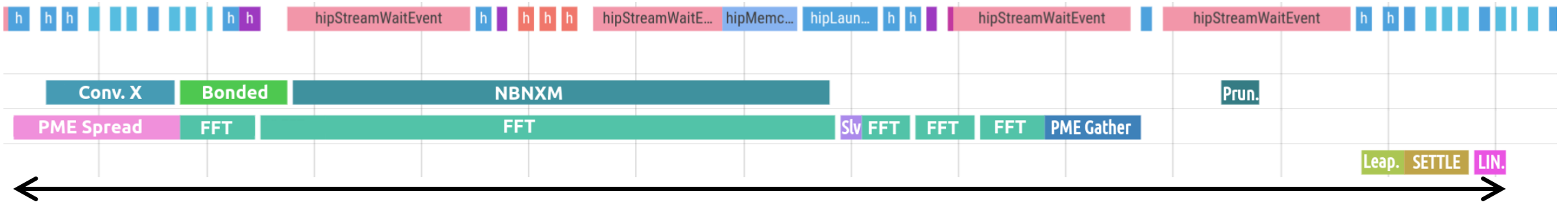


# Performance: AMD MI250X



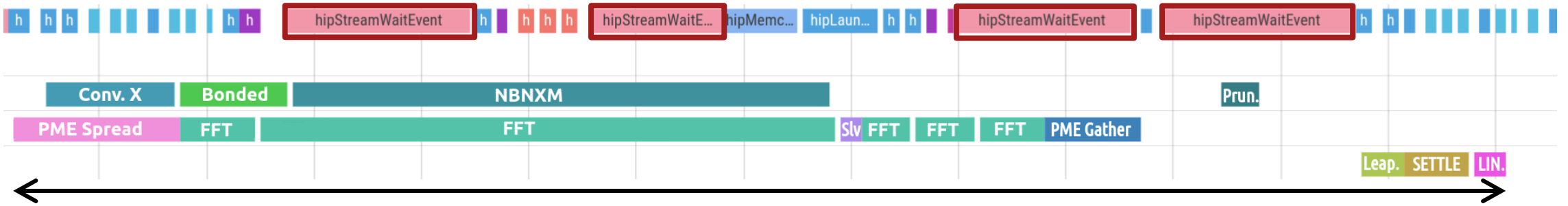



# Performance: GPU work



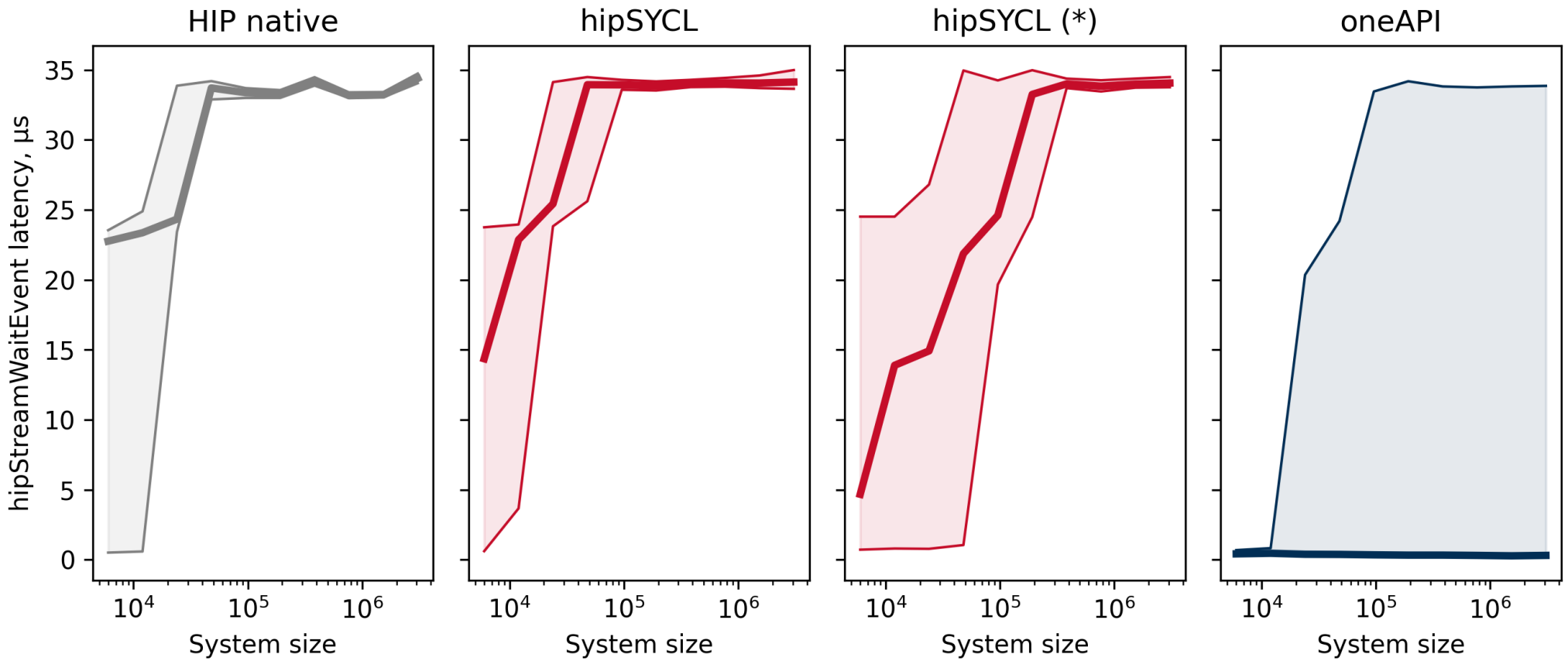

| | |
|---|---|
| HIP native | ~270 µs |
| hipSYCL | ~330 µs |
| hipSYCL (no caching) | ~380 µs |
| oneAPI | ~480 µs |

# Performance: GPU work



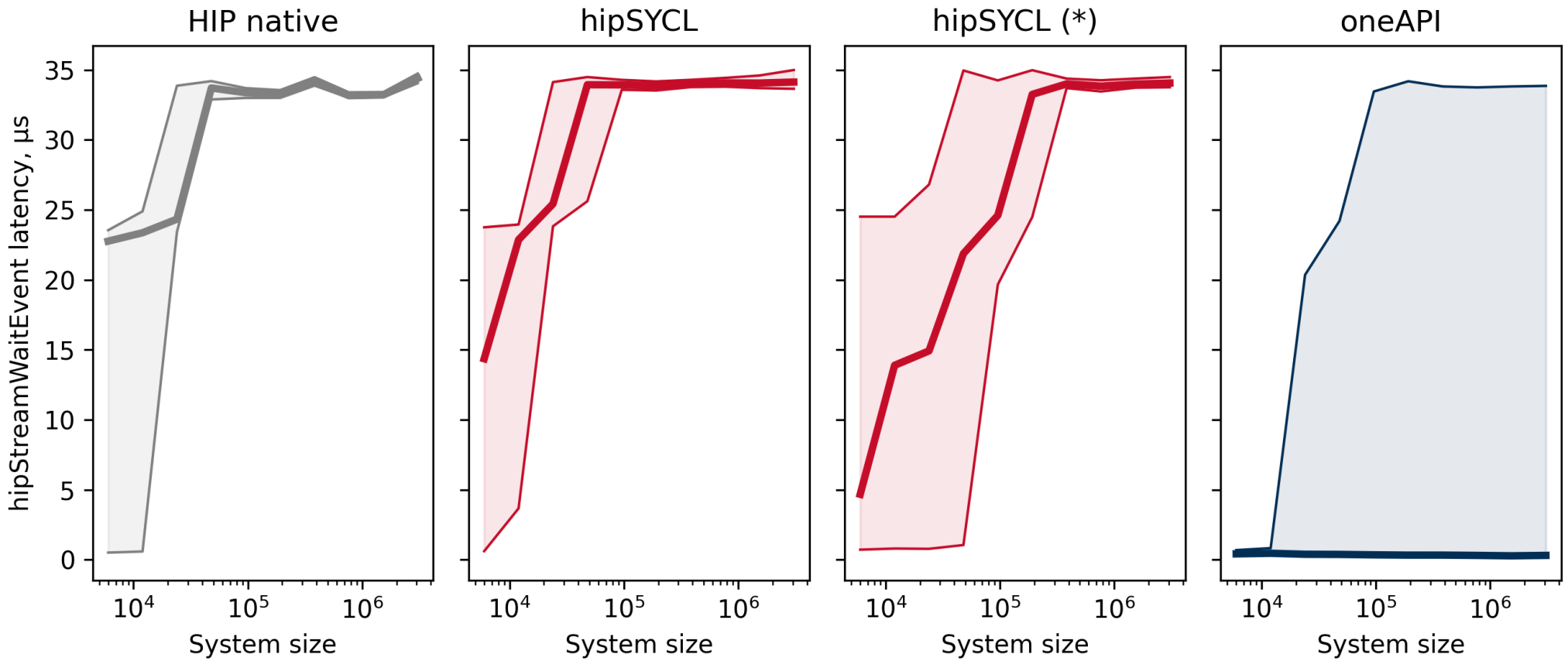

| | |
|---|---|
| HIP native | ~270 µs |
| hipSYCL | ~330 µs |
| hipSYCL (no caching) | ~380 µs |
| oneAPI | ~480 µs |

# API latency




It gets worse the more concurrent work we have on the GPU!

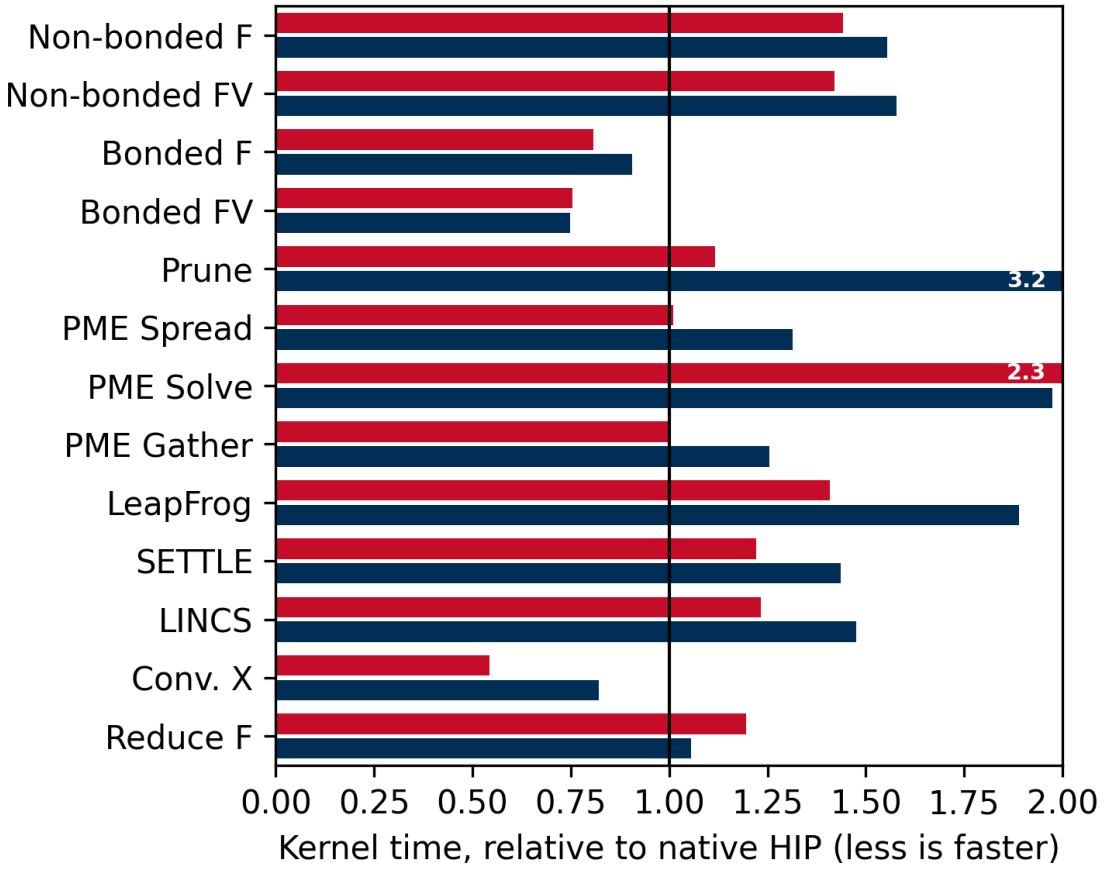
# API latency

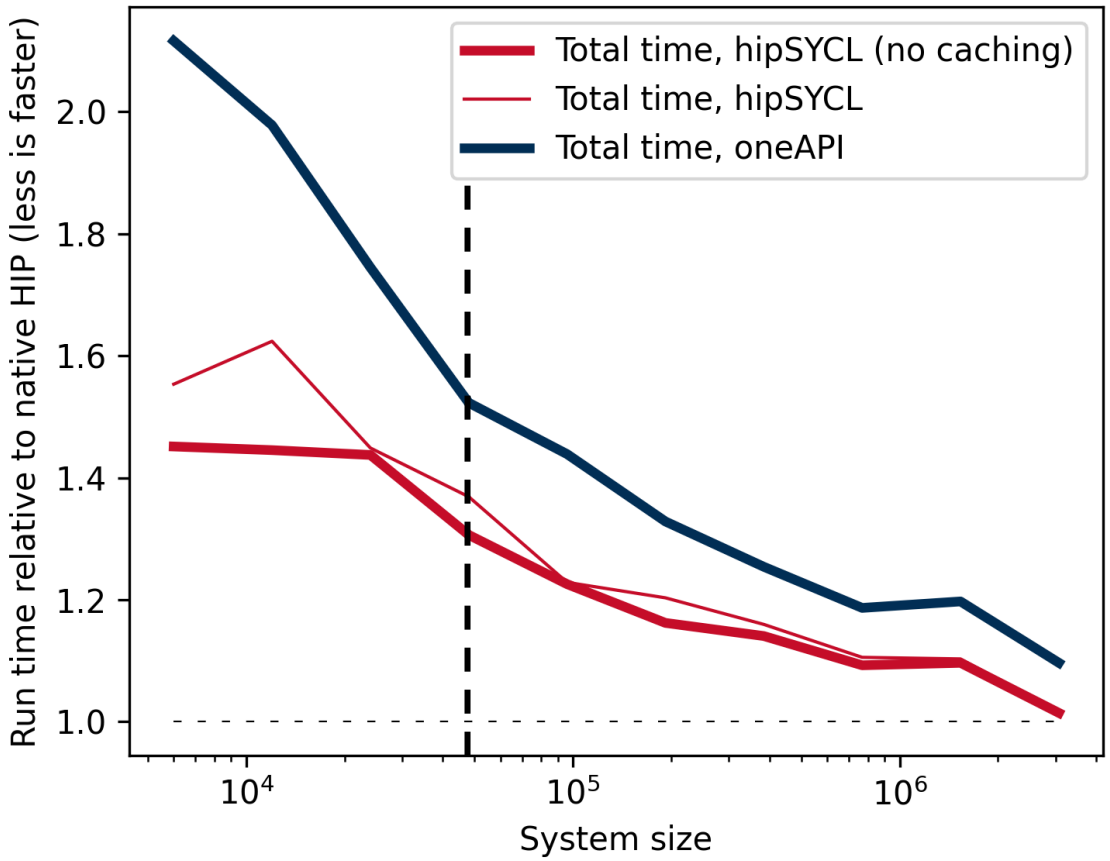



It gets worse the faster we are scheduling new work!

On CUDA, <2 µs

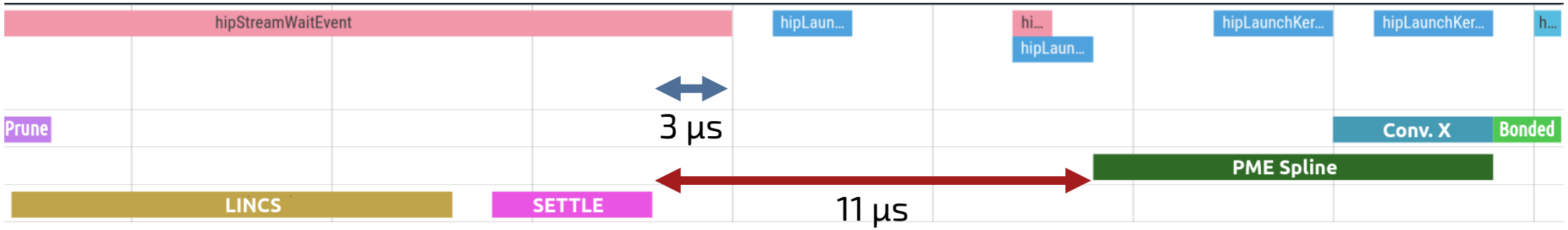
# Performance: AMD MI250X



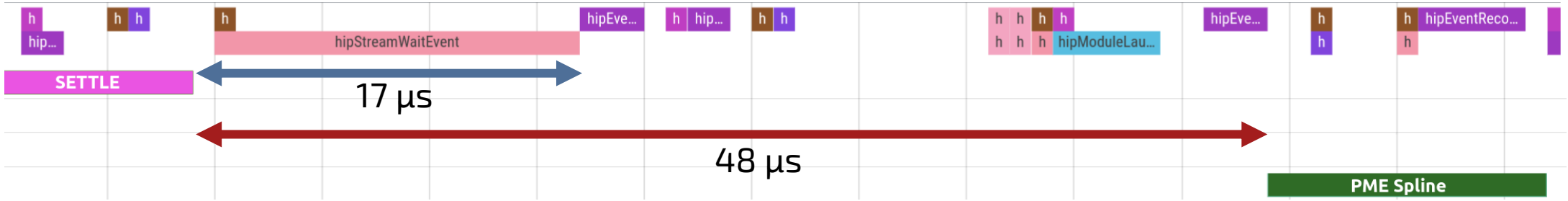



# Performance: HIP vs. oneAPI

## Native HIP




## oneAPI



# Solutions / workarounds

- Minimize extra API calls
  - Coarse-grained events (hipSYCL extension)
  - Discard queue events (oneAPI extension, too restrictive)
  - Event reuse (done in hipSYCL)
- Avoid DAG caching
  - HIPSYCL_RT_MAX_CACHED_NODES
  - HIPSYCL_RT_SCHEDULER=direct
  - Priorities?
- Submitting all at once: CUDA/HIP Graphs?
  - See talk by Ewan Crawford after lunch!
  - Efficient interaction with host tasks?

# Conclusions

- Kernel performance is not a (huge) problem
  - It's all LLVM
  - There is interop
- Scheduling can be the bottleneck
  - In many ways!
- Both oneAPI and hipSYCL add overhead
  - And sometimes, it's the underlying platform
    - Which makes it hard to narrow down
  - Applications can avoid some of it
  - The rest should be dealt in runtimes and standard extensions

# Acknowledgements

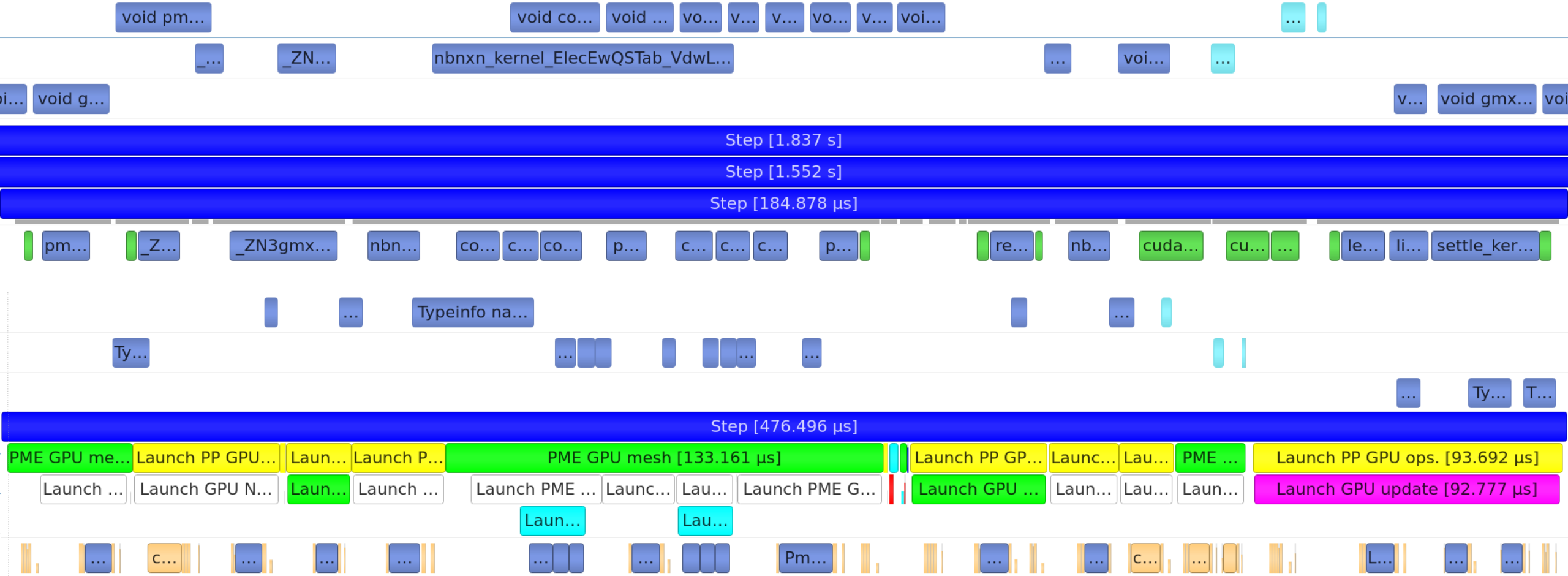

- Intel Corporation
- AMD for sharing their HIP port
- Mark Abraham, Heinrich Bockhorst, Roland Schulz (Intel)
- Aksel Alpay (Heidelberg University Computing Centre)
- GROMACS dev team

# Backup slides

# Technical details

- hipSYCL: 0.9.4
- oneAPI: sycl-nightly/20230410
- GROMACS 2023.1-dev
- CUDA: 11.8.0
- ROCm 4.3.3
- Grappa benchmark, GPU-resident mode
- NVIDIA V100
- AMD MI250X

# CUDA vs oneAPI: 6k trace



# HIP: 3072k kernels