13th International Workshop on OpenCL and SYCL

IWOCL 2025

SYCL-Graph in GROMACS

Andrey Alekseenko, KTH Royal Institute of Technology
Ewan Crawford, Codeplay Software

Pablo Reble and Adam Fidel, Intel.

Ben Tracy, Fábio Mestre, and Konrad Kusiak, Codeplay Software.

April 7-11, 2025 | Heidelberg, Germany | iwocl.org

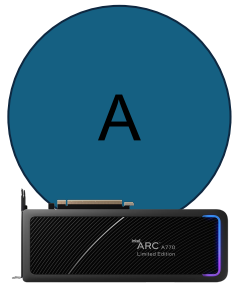
KHRONOS GROUP

Agenda

- SYCL-Graph Overview
- GROMACS Overview
- Case Study
  - Native Command Extension Integration
  - Performance Plots
  - Trace Examination
- SYCL-Graph Future Work

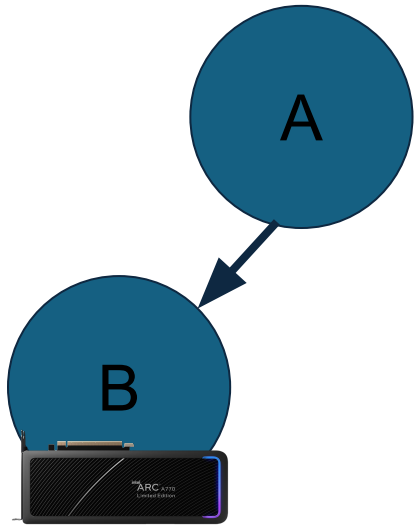
SYCL-Graph Overview

SYCL Execution Model Graph



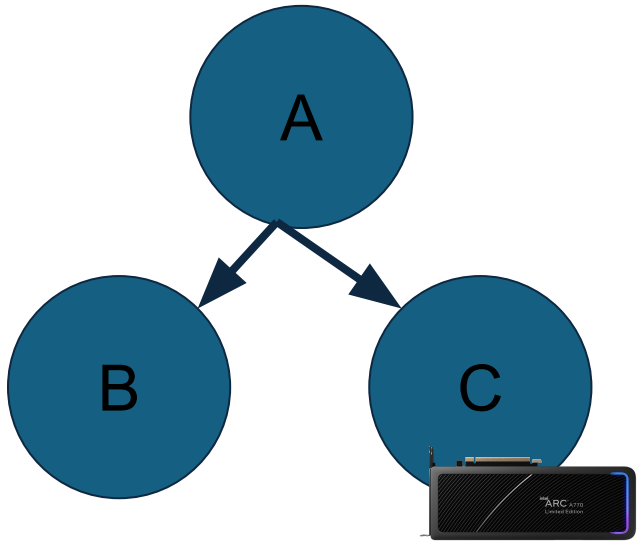
- SYCL defines an execution graph where nodes are tasks that will execute on a device.
- An edge is a happens-before dependency between tasks.
- No cycles in the graph, it is a Directed Acyclic Graph (DAG).
- Submissions to the device are made at the same time as user defines the commands.

SYCL Execution Model Graph



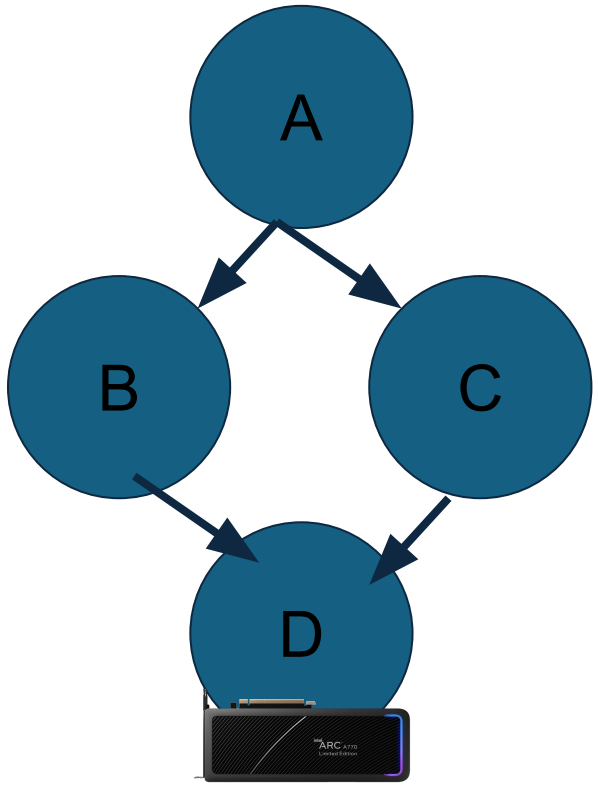
- SYCL defines an execution graph where nodes are tasks that will execute on a device.
- An edge is a happens-before dependency between tasks.
- No cycles in the graph, it is a Directed Acyclic Graph (DAG).
- Submissions to the device are made at the same time as user defines the commands.

SYCL Execution Model Graph



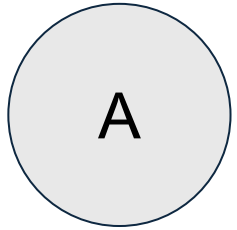
- SYCL defines an execution graph where nodes are tasks that will execute on a device.
- An edge is a happens-before dependency between tasks.
- No cycles in the graph, it is a Directed Acyclic Graph (DAG).
- Submissions to the device are made at the same time as user defines the commands.

SYCL Execution Model Graph



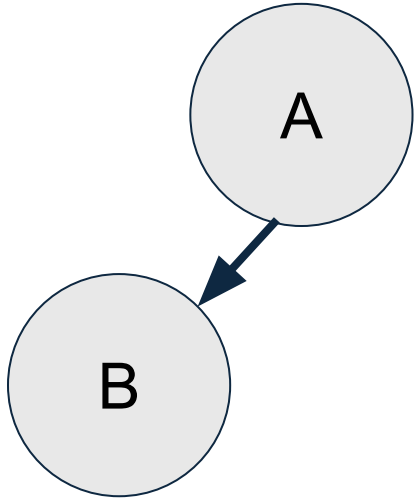
- SYCL defines an execution graph where nodes are tasks that will execute on a device.
- An edge is a happens-before dependency between tasks.
- No cycles in the graph, it is a Directed Acyclic Graph (DAG).
- Submissions to the device are made at the same time as user defines the commands.

SYCL-Graph Extension



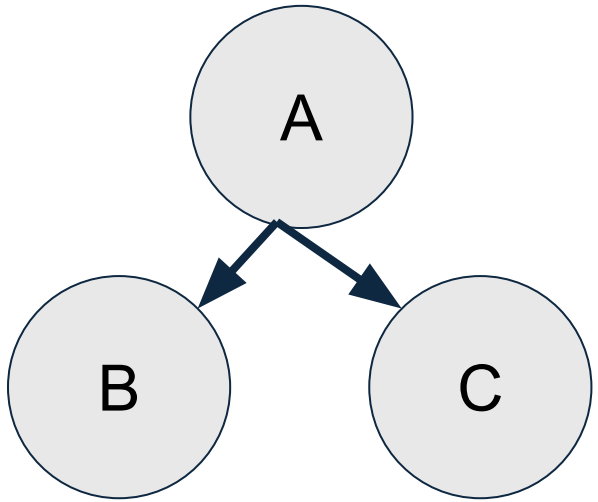
- SYCL-Graph extension separates definition of the task graph from submission to the device.
- Defines an API to give user explicit control over graph construction and submission.
- User defined graph is known to SYCL runtime ahead of submission.

SYCL-Graph Extension



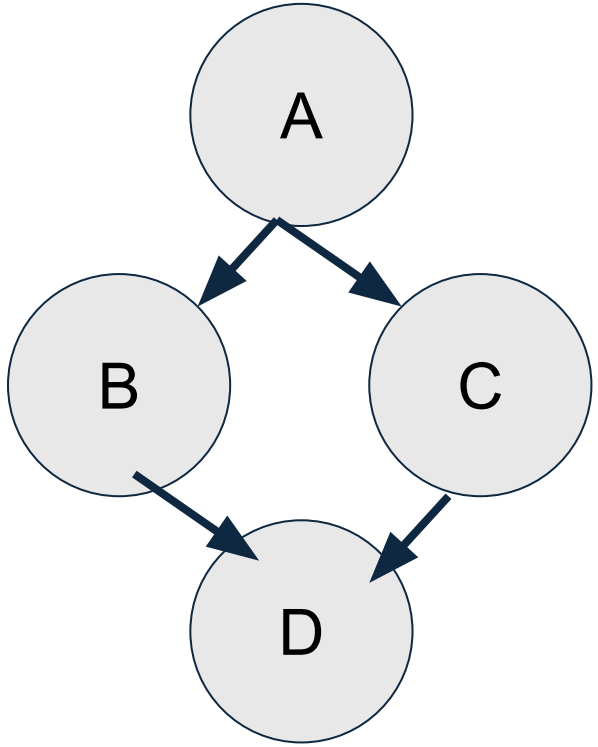
- SYCL-Graph extension separates definition of the task graph from submission to the device.
- Defines an API to give user explicit control over graph construction and submission.
- User defined graph is known to SYCL runtime ahead of submission.

SYCL-Graph Extension



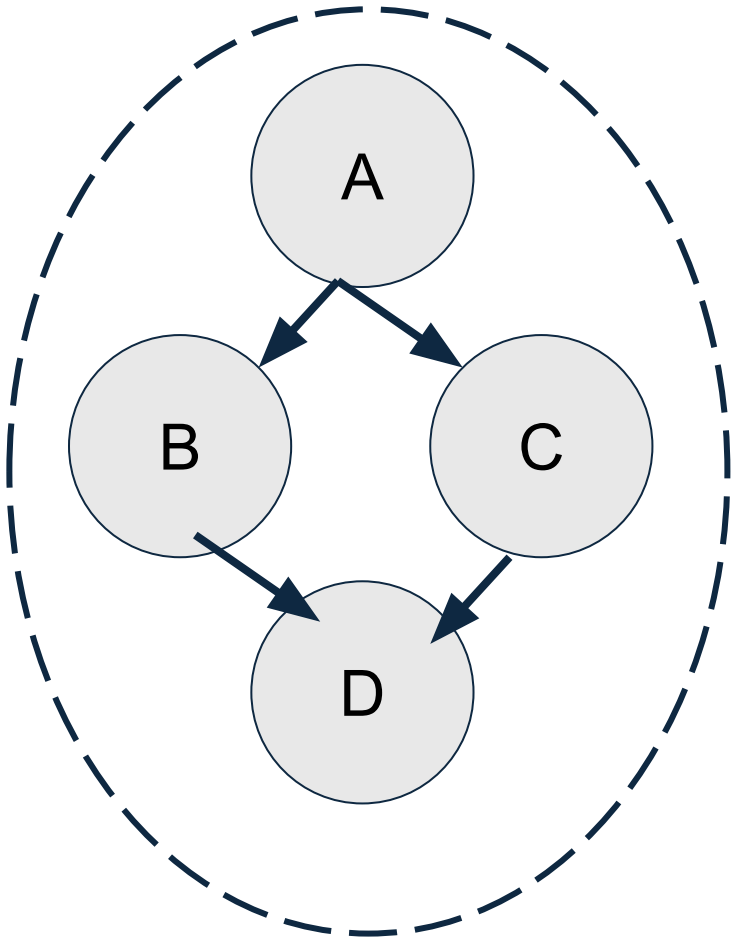
- SYCL-Graph extension separates definition of the task graph from submission to the device.
- Defines an API to give user explicit control over graph construction and submission.
- User defined graph is known to SYCL runtime ahead of submission.

SYCL-Graph Extension



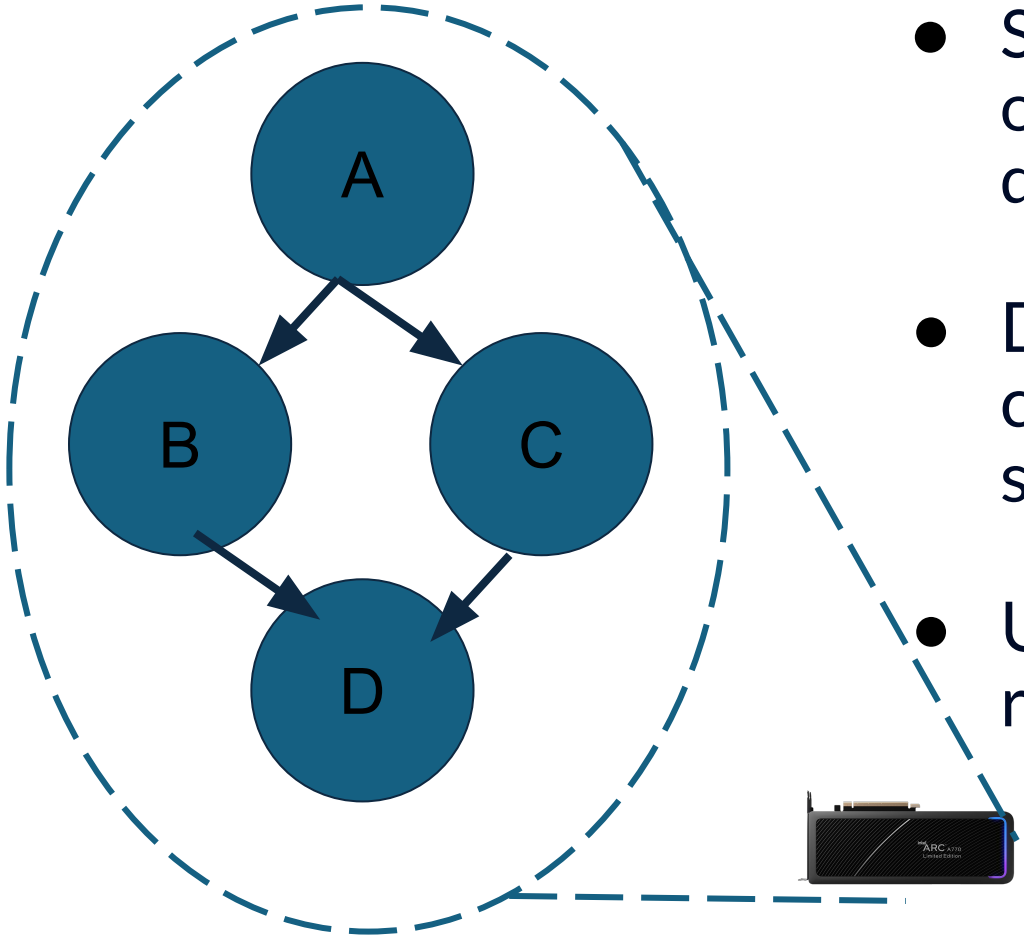
- SYCL-Graph extension separates definition of the task graph from submission to the device.
- Defines an API to give user explicit control over graph construction and submission.
- User defined graph is known to SYCL runtime ahead of submission.

SYCL-Graph Extension



- SYCL-Graph extension separates definition of the task graph from submission to the device.
- Defines an API to give user explicit control over graph construction and submission.
- User defined graph is known to SYCL runtime ahead of submission.

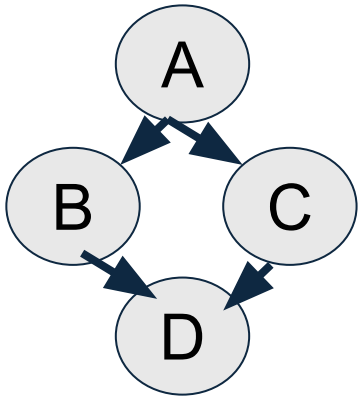
SYCL-Graph Extension



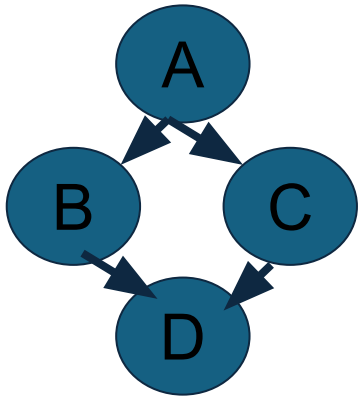
- SYCL-Graph extension separates definition of the task graph from submission to the device.
- Defines an API to give user explicit control over graph construction and submission.
- User defined graph is known to SYCL runtime ahead of submission.

Benefits of SYCL-Graph

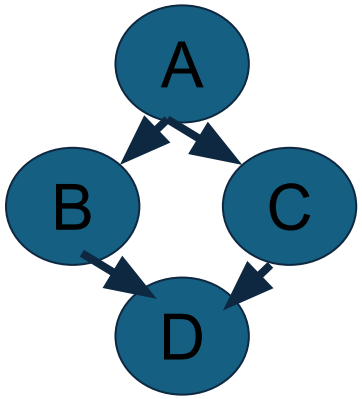
- A graph can be defined once and submitted as many times as required.
- Reduces latency when submitting commands to the device.
- Optimizations become available across the defined graph.



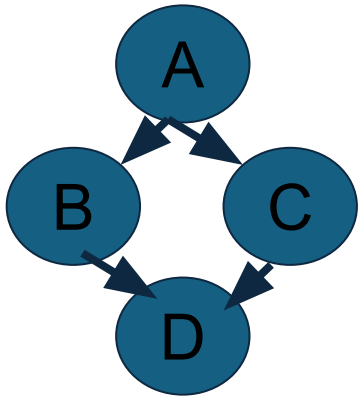
Define



Execute



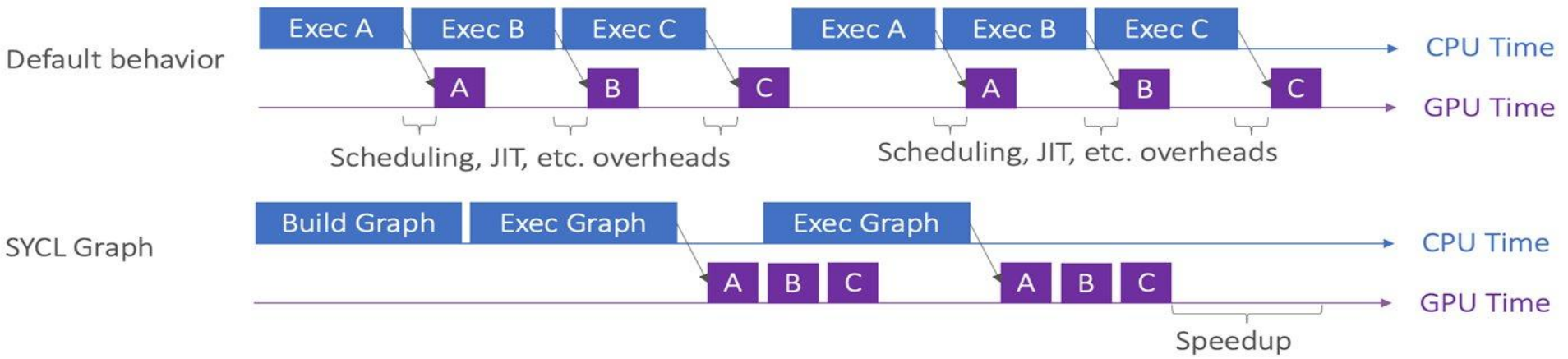
Execute



Execute

Benefits of SYCL-Graph

- A graph can be defined once and submitted as many times as required.
- Reduces latency when submitting commands to the device.
- Optimizations become available across the defined graph.



Benefits of SYCL-Graph

- A graph can be defined once and submitted as many times as required.
- Reduces latency when submitting commands to the device.
- Optimizations become available across the defined graph.

Benefits of SYCL-Graph

- A graph can be defined once and submitted as many times as required.
- Reduces latency when submitting commands to the device.
- Optimizations become available across the defined graph.

We will Illustrate these in this presentation

Current Status of SYCL-Graph

- sycl_ext_oneapi_graph (aka SYCL-Graph) is an experimental oneAPI vendor extension to SYCL 2020.
  - Project goal to bring to SYCL WG as a KHR extension.
- Open source development in DPC++.
- Included oneAPI releases from 2024.0 onwards
- Implemented for Level-Zero, OpenCL, CUDA, and HIP backends to SYCL.

Integration by upstream applications

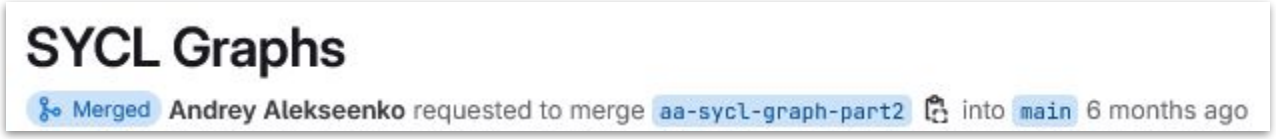
LLama.cpp

https://github.com/ggml-org/llama.cpp/pull/12371

Kokkos

https://github.com/kokkos/kokkos/pull/6912

GROMACS



https://gitlab.com/gromacs/gromacs/-/merge_requests/4604

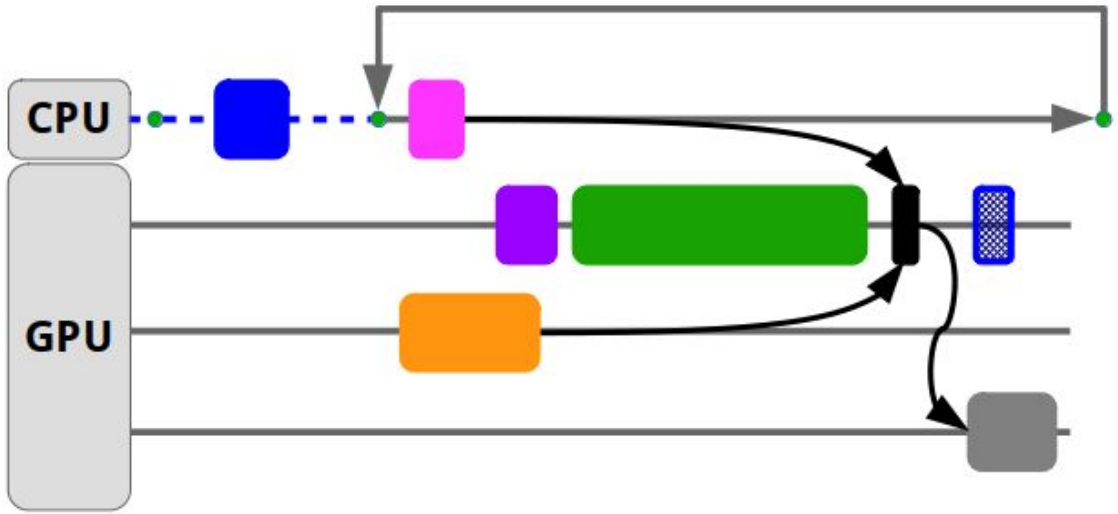
GROMACS Overview

Intro to GROMACS

- Open-source molecular dynamics engine
- One of the most used HPC codes worldwide
- Mature support for several GPU backends:
  - CUDA: NVIDIA
  - HIP: AMD (recent addition)
  - OpenCL: AMD, Apple, Intel, NVIDIA (deprecated)
  - SYCL (oneAPI & AdaptiveCpp): AMD, Intel, NVIDIA

GROMACS: GPU acceleration layer

- CUDA Graphs implemented in 2023
  - Record, then replay
  - 5-20 operations per step
  - Graph structure is the same, but parameters change every 50-500 steps
  - Currently works only in fully GPU-resident mode, no host-task support



Why SYCL-Graph can benefit GROMACS

- Large systems: performance limited by kernel execution time.
- Smaller systems: performance bottleneck moves to scheduling of tasks to the GPU.
  - Scheduling also on critical path when scaling to multiple nodes.
- SYCL-Graph reduces GPU scheduling latency by defining the graph of GPU tasks prior to submission.

SYCL-Graph implementation

- GROMACS SYCL backend (DPC++) to target the GPUs of Intel (Level Zero), NVIDIA (CUDA), and AMD (HIP)
- With CUDA Graphs in place, enabling SYCL-Graphs was easy
- PME requires using a GPU FFT library:
  - oneMKL for Level Zero.
  - vkFFT for CUDA and HIP.

Capturing GPU FFT calls in the SYCL-Graph

On Level Zero, the oneMKL* calls are passed a sycl::queue that is recording its commands to a graph.

On CUDA/HIP, we need a way to capture the interop with vkFFT native calls as part of the SYCL-Graph:

- Host-task nodes in a SYCL-Graph are implemented by partitioning the graph into multiple backend command-buffers, introducing extra scheduling overhead at execution time.
- Better: Use sycl_ext_codeplay_enqueue_native_command to add commands to native graph representation.

* oneMKL is not fully supported as of 2025.1, and is a work in-progress

Native Command Support in Graphs

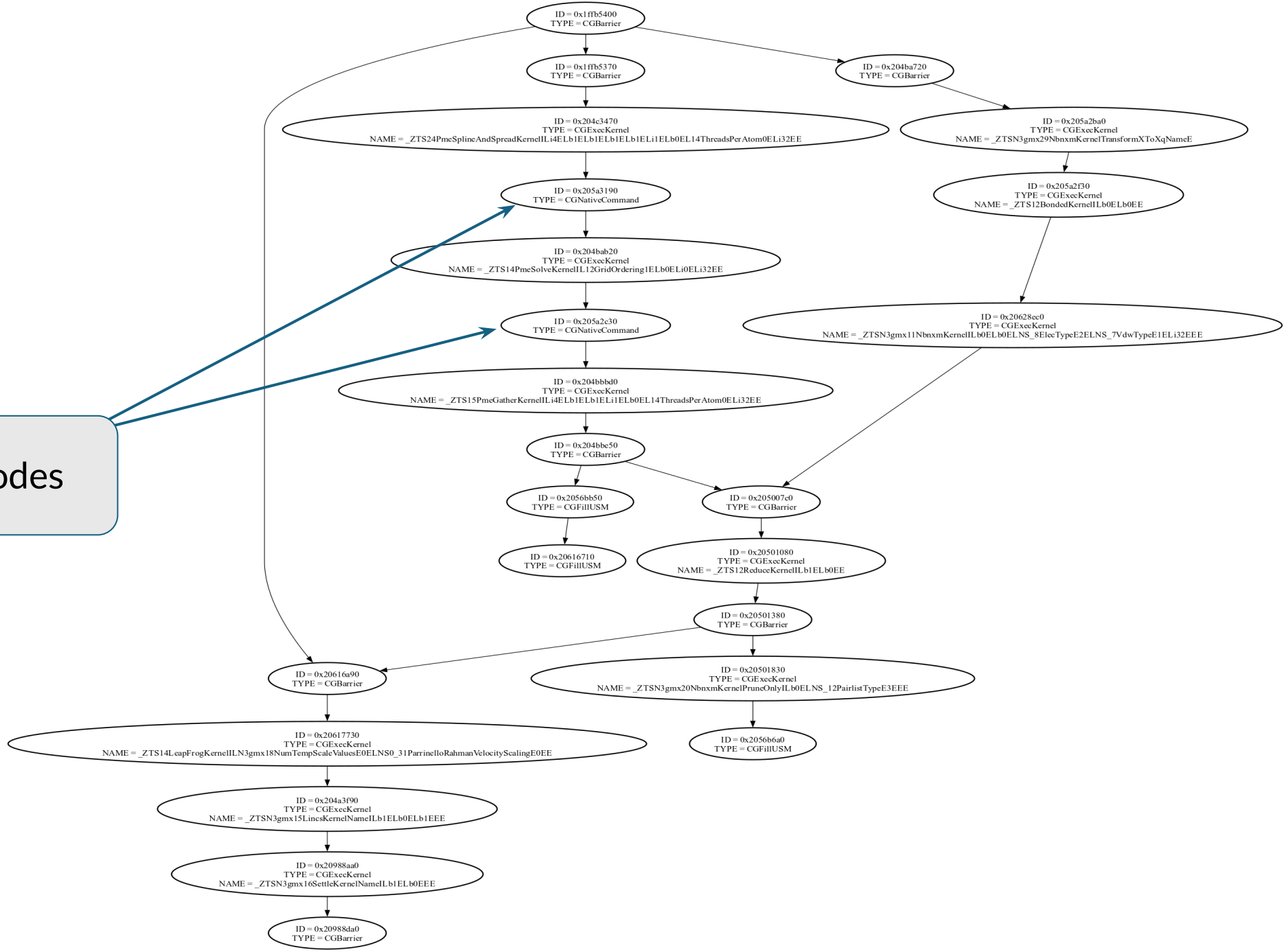
- Supports Level Zero, CUDA, HIP, OpenCL backends
- GROMACS PR open
- Can use native stream record APIs to add vkFFT call to native graph.
- The git branch used for benchmark results.

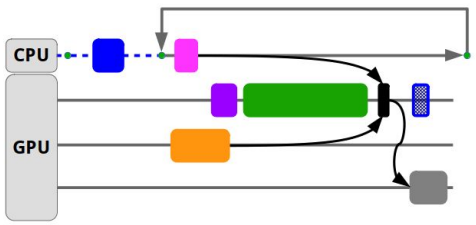
```
#if defined(SYCL_EXT_ONEAPI_ENQUEUE_NATIVE_COMMAND)
  sycl::ext::oneapi::experimental::submit(
      queue_,
      [&](sycl::handler& cgh)  {
        cgh.ext_codeplay_enqueue_native_command(
            [=](sycl::interop_handle& h) {
              if (h.ext_codeplay_has_graph()) {
                auto NativeQueue = h.get_native_queue<backend>();
                auto NativeGraph = h.ext_codeplay_get_native_graph<backend>();
                BeginCaptureToGraph(NativeQueue, NativeGraph);

                launchVkFft(launchParams_);

                EndCaptureToGraph(NativeQueue, NativeGraph);
              } else {
                launchVkFft(launchParams_);
              }
            });
      });
#else // ...
```

oneAPI

Graph Structure

Native Command Nodes





oneAPI

Case Study

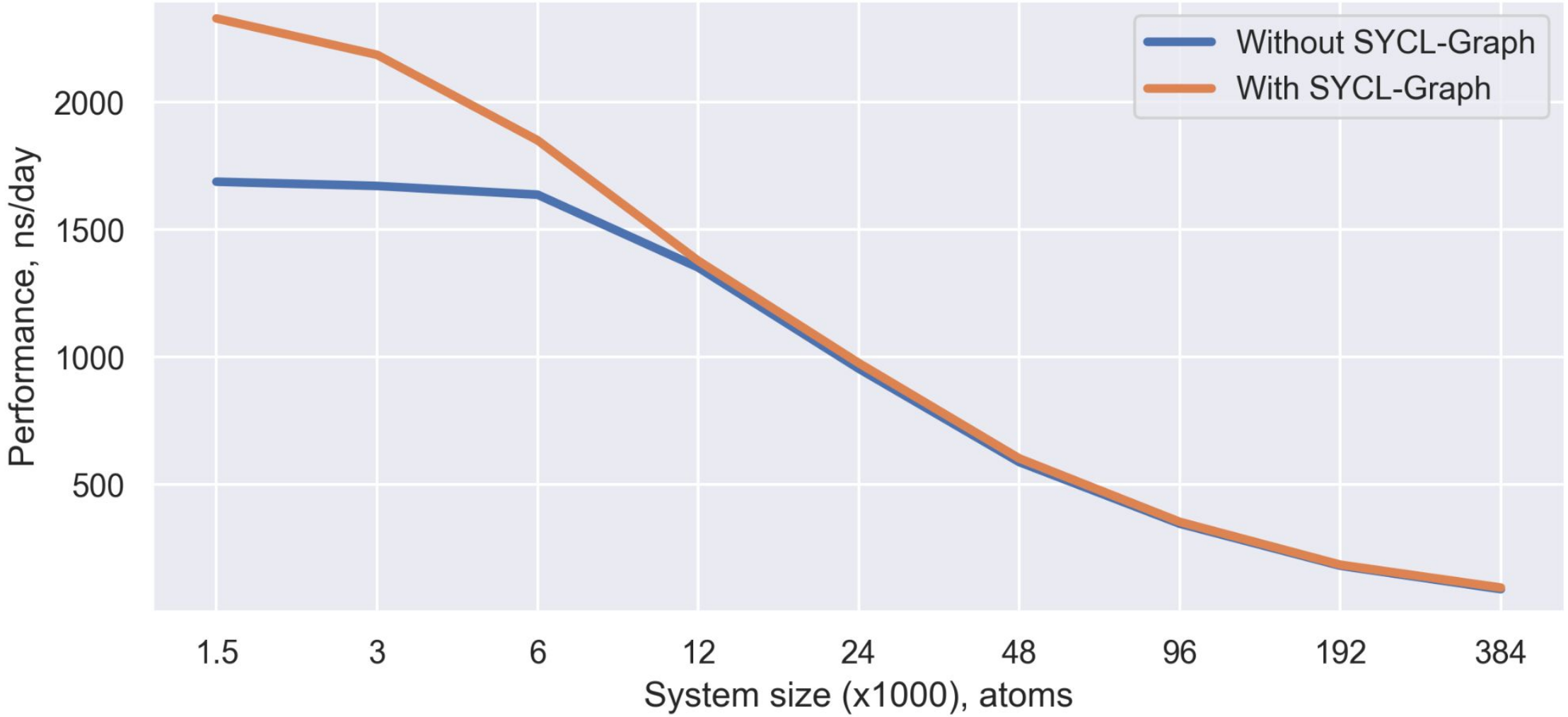
Benchmarks

- Grappa PME: A set of artificial benchmark systems roughly mimicking biomolecular systems, using long-range (PME) treatment of electrostatics. https://zenodo.org/records/11234002

- Contains system size ranging from 1500 atoms to 46 million atoms. Allowing us to benchmark the performance of the SYCL-Graph extension across different system sizes (kernel durations)

- NVIDIA (A100), Intel (B580, PVC1100)

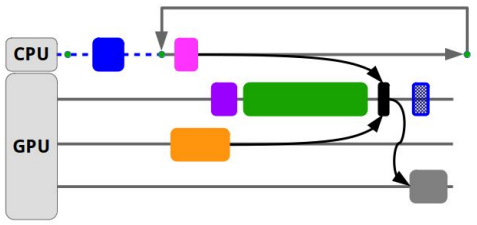
Absolute performance on NVIDIA A100

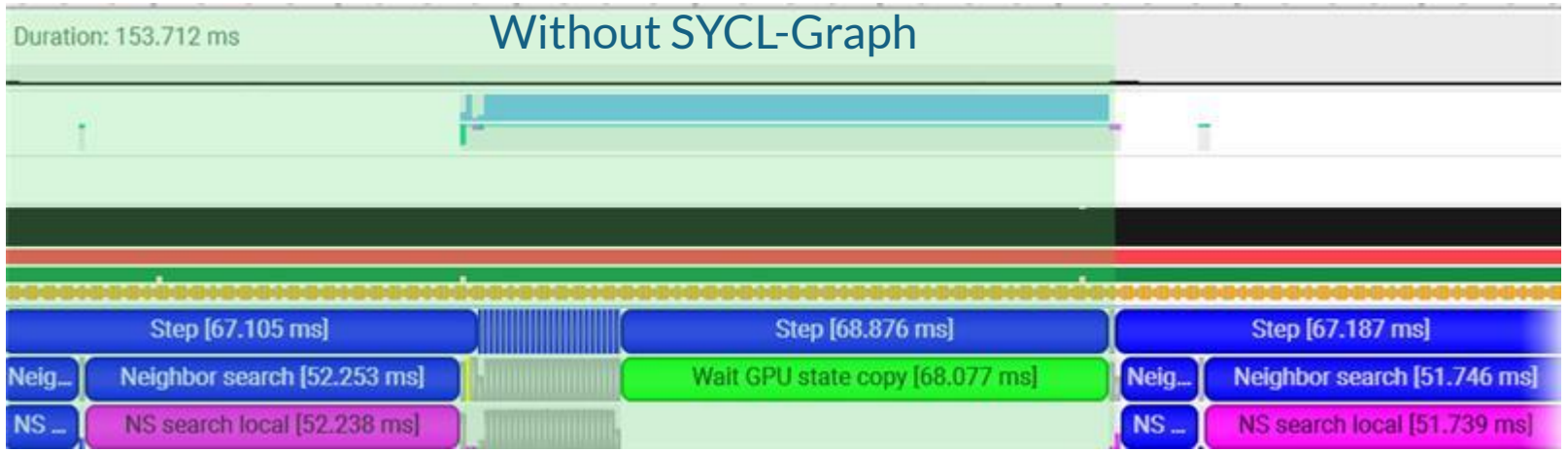


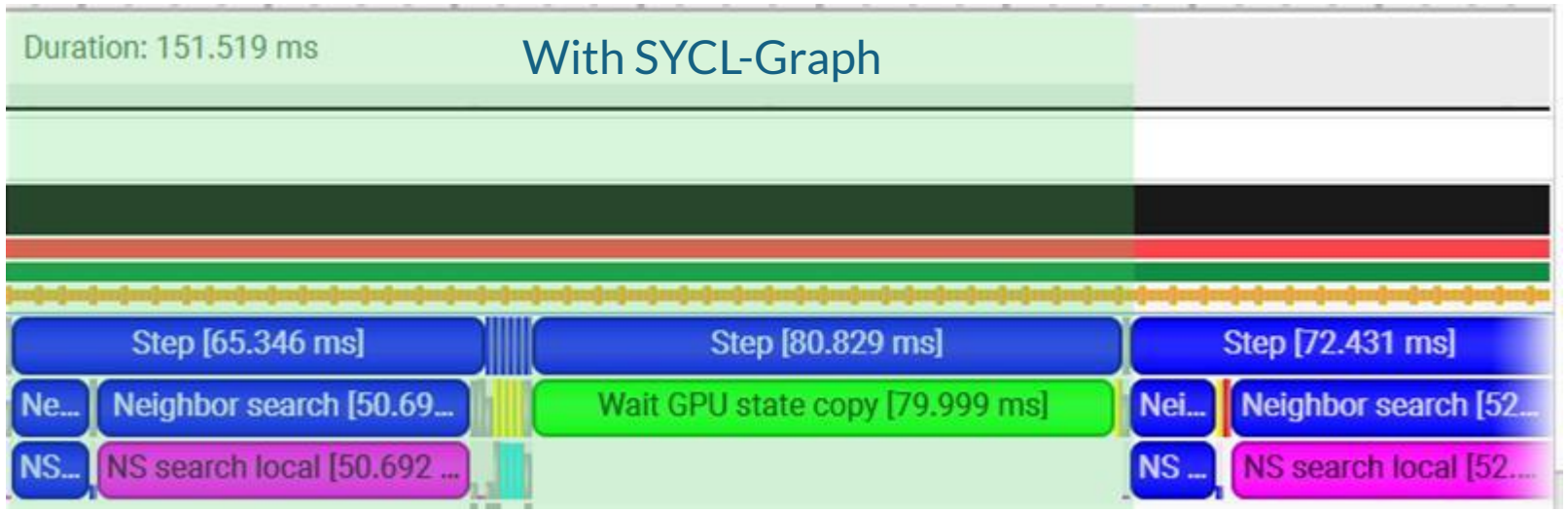
- For configuration see [1] on "Test Configuration" slide.
- Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.
- Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary.

192k System Size: Comparing 100 Steps

- For large system sizes, GROMACS is bound by GPU execution time from kernels.
- Improving the GPU scheduling latency doesn't alleviate this, so performance remains the same.





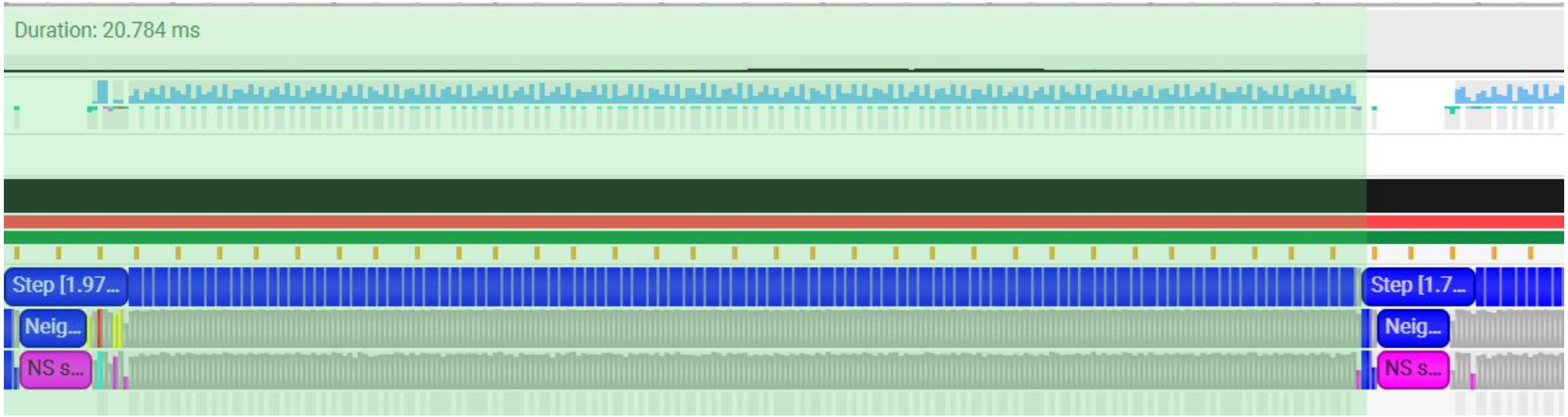


Taken from A100 Nsight trace of benchmarked 192k system size configuration

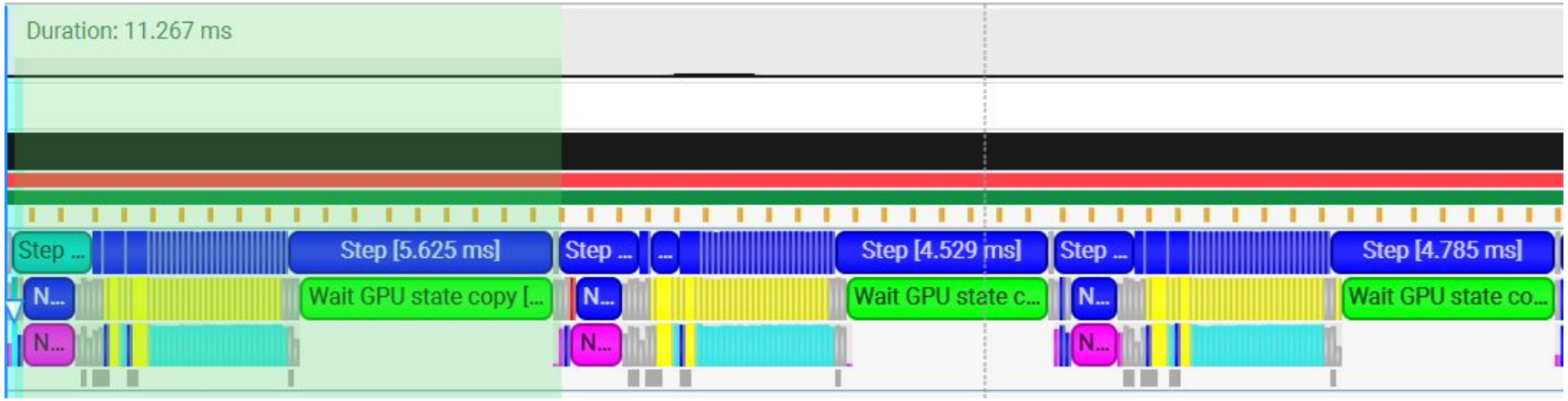
oneAPI

3k System Size: Comparing 100 Steps

Without SYCL-Graph



With SYCL-Graph

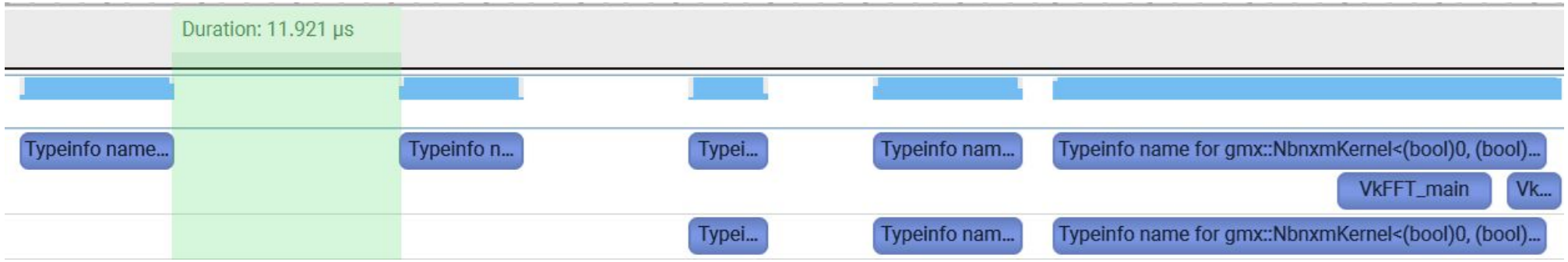


Taken from A100 Nsight trace of benchmarked 3k system size configuration

oneAPI

Single Step: Without SYCL-Graph

GPU: Gaps in execution

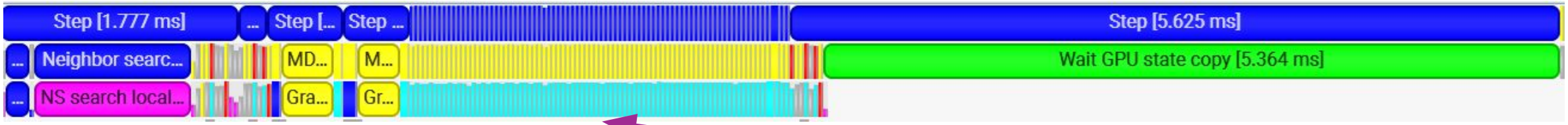


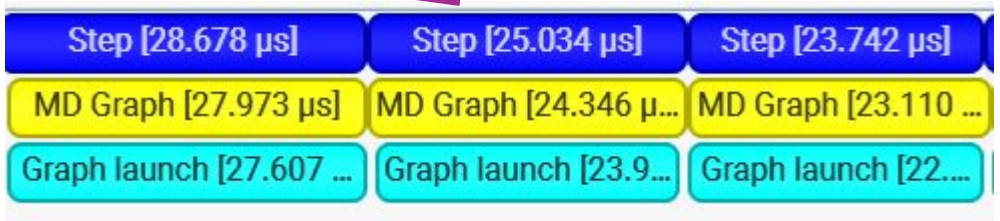
CPU: 180-200 µs/step

Taken from A100 Nsight trace of benchmarked 3k system size configuration

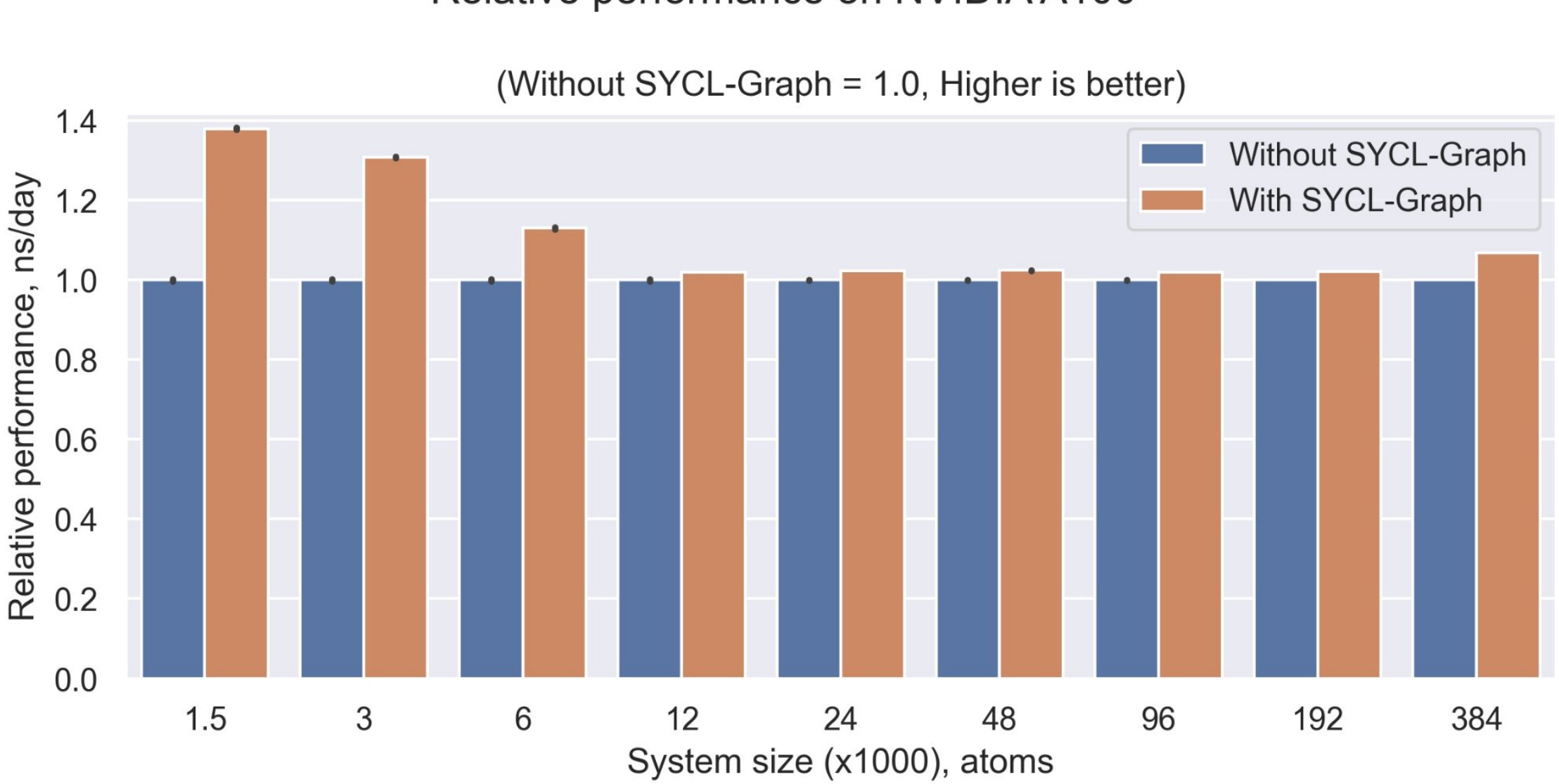
oneAPI

100 Steps: With SYCL-Graph



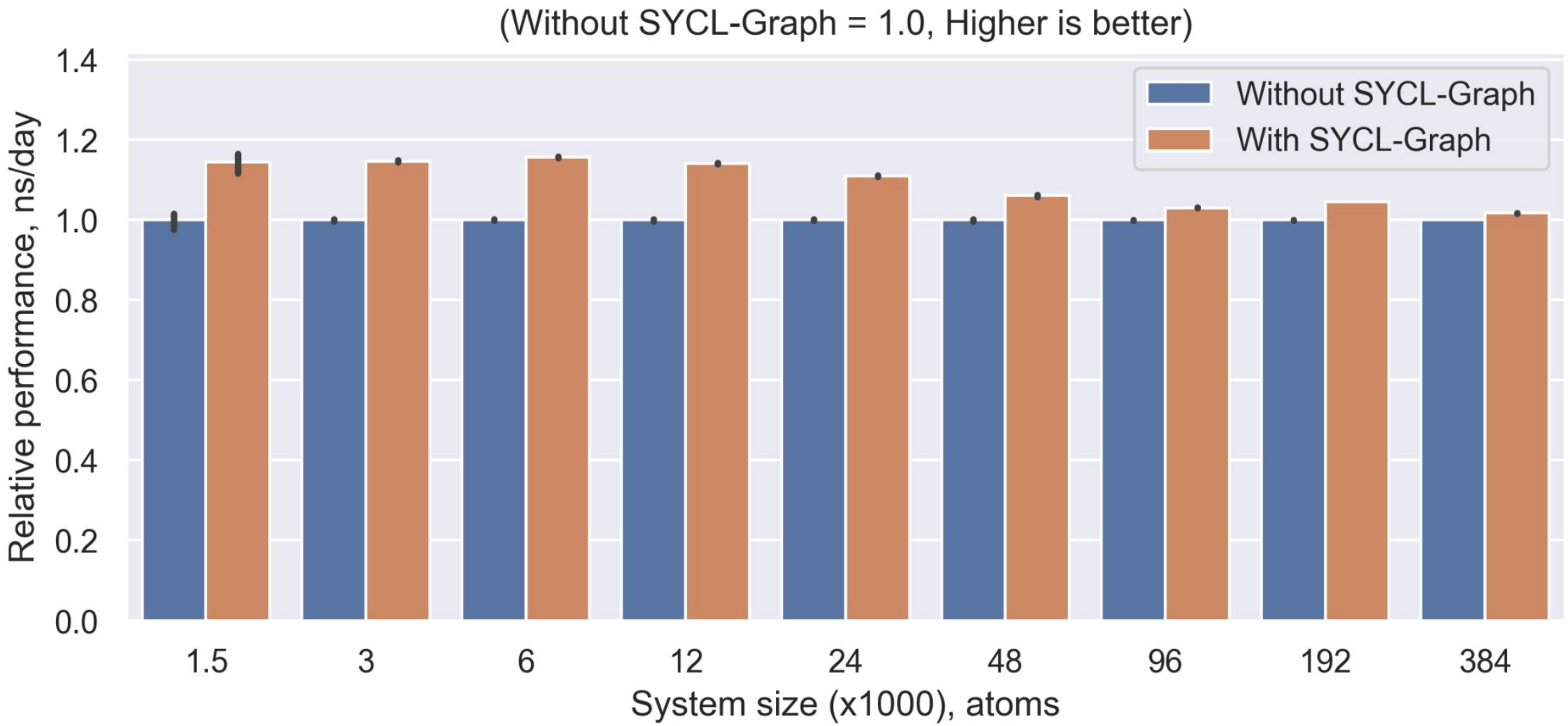


- After every Neighbor Search (NS) step we get two long steps ~500µs when (re)instantiating the graphs, and then around 20µs/step for the next 97 steps.
- "Wait GPU state copy" before the next NS step indicates that GPU is still busy.
- Shows that SYCL-Graph allowed us to push the bottleneck from CPU scheduling to GPU execution.



- For configuration see [1] on "Test Configuration" slide.
- Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.
- Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary.

Relative performance on Intel Arc B580



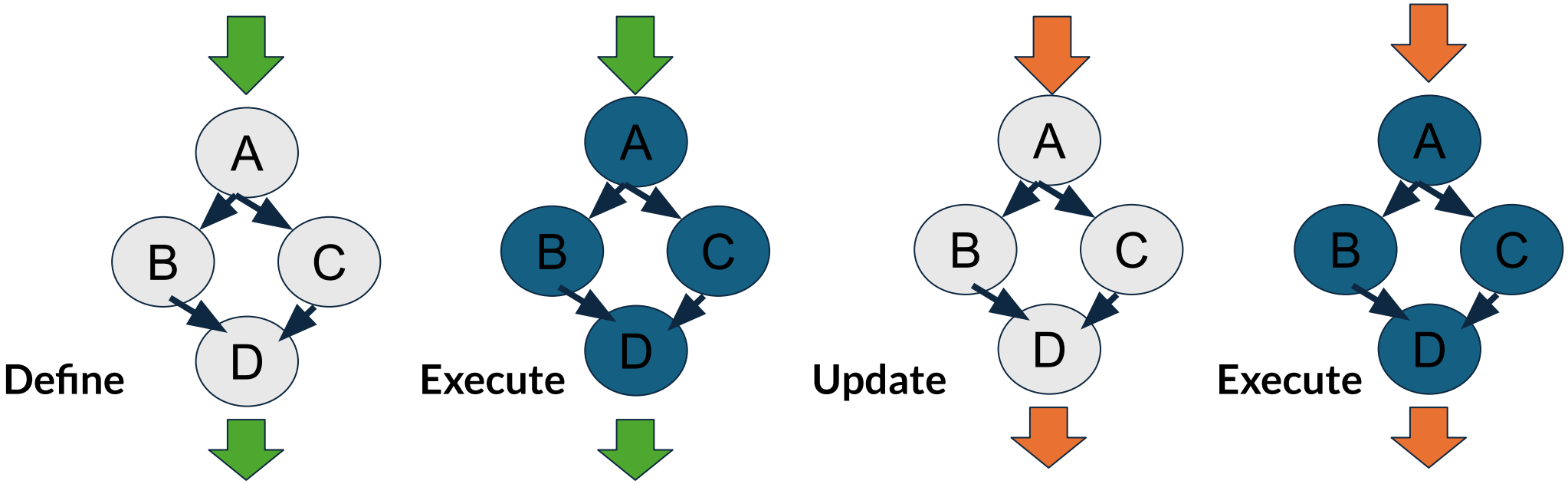
- For configuration see [2] on "Test Configuration" slide.
- Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.
- Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary.

oneAPI

SYCL-Graph Future Work

Graph Update

- Update is a feature in SYCL-Graph that allows the inputs and outputs of a graph to be modified between executions of an instantiated graph.
- More efficient than recreating a graph with the same topology of nodes and edges when only inputs/outputs are different.



Enhancing SYCL-Graph Update Support

- SYCL-Graph provides APIs for updating an executable graph.
- Kernel nodes are the only type of node which can currently be updated on the device.
- The graphs created from PME workload contains native-command nodes and USM fills, which are not yet supported for update.

```
template<>
class command_graph<graph_state::executable> {
public:
    command_graph() = delete;

    void update(node& node);
    void update(const std::vector<node>& nodes);
    void update(const command_graph<graph_state::modifiable>& graph);
};
```

Update In Applications

- GROMACS CUDA backend uses CUDA-Graph whole-graph update to avoid re-instantiating the graph where possible.

- Being able to use the equivalent SYCL-Graph whole-graph update API in GROMACS would allow further performance benefits.
  - Graph parameters change every 5-50 steps while structure stays the same.
  - Improve ~500µs time seen on A100 when reinstantiating the SYCL-Graph.

- Llama.cpp has a similar use-case for update to avoid excess instantiations of a queue recorded graph.

Summary

- SYCL-Graph provides the user with a mechanism for first defining a graph of commands, and then submitting it to the device with low-latency.
- Integrated SYCL-Graph usage into GROMACS shows performance benefits on the PME benchmark for small system sizes on NVIDIA and Intel GPUs.
- Future development of the SYCL-Graph update feature is expected to enable further performance benefits to GROMACS.

Benchmarked Configurations

[1] SYCL Grappa PME NVIDIA A100

- **Testing Date**: Performance results are based on testing by Codeplay as of 03/04/25 and may not reflect all publicly available updates.
- **Configuration Details**: Ubuntu 22.04.5 LTS. Testing using Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz, GPU: NVIDIA A100 PCIE 40GB GPU memory. GROMACS branch https://gitlab.com/gromacs/gromacs/-/merge_requests/4954 compiled with CMake Options: -DGMX_GPU=SYCL -DGMX_SYCL_ENABLE_GRAPHS=ON, -DGMX_FFT_LIBRARY=FFTW3, -DGMX_BUILD_OWN_FFTW=ON, -DGMX_GPU_FFT_LIBRARY=vkfft, -DGMX_SYCL_ENABLE_EXPERIMENTAL_SUBMIT_API=ON, -DGMX_GPU_NB_CLUSTER_SIZE=8, -DCMAKE_BUILD_TYPE=Release,, -DSYCL_CXX_FLAGS_EXTRA=-fsycl-targets=nvidia_gpu_sm_80. DPC++ compiled with 27aae7ae2cb5a89a88b5a04af1ae8466d34e9e1a with CUDA 12.6.2
- **Run Command**: "gmx mdrun -pme gpu -pmefft gpu -s pme.tpr -nb gpu -update gpu -bonded gpu -nstlist 100 -notunepme -resetstep 2000 -ntomp 12 -nobackup -noconfout -pin on" with the "SYCL_CACHE_PERSISTENT=1 ONEAPI_DEVICE_SELECTOR=cuda:gpu" environment variables, and for the *with SYCL-Graph* case only also environment variable "GMX_CUDA_GRAPH=1"

[2] SYCL Grappa PME Intel Arc B580

- **Testing Date**: Performance results are based on testing by Intel as of 07/04/25 and may not reflect all publicly available updates.
- **Configuration Details**: Ubuntu 24.04.1 LTS. 134th Gen Intel(R) Core(TM) i9-13900K CPU, GPU: Intel(R) B580 with driver version 1.6.32567+16. GROMACS commit 2eddb00c55e70289a6369f4acc5f2e5f04d25f2a compiled with CMake Options: -DGMX_SYCL_ENABLE_GRAPHS=ON, -DGMX_FFT_LIBRARY=MKL, -DGMX_BUILD_OWN_FFTW=ON, -DGMX_GPU_FFT_LIBRARY=MKL, -DGMX_SYCL_ENABLE_EXPERIMENTAL_SUBMIT_API=ON, -DGMX_GPU_NB_CLUSTER_SIZE=8, -DCMAKE_BUILD_TYPE=Release, -DGMX_OPENMP=OFF. DPC++ compiled with aa0068bd07e9861713093f84db95d4e75ec1edbe and oneMKL 2025.0
- **Run Command**: "gmx mdrun -pme gpu -pmefft gpu -s pme.tpr -nb gpu -update gpu -bonded gpu -nstlist 100 -nsteps 10000 -resetstep 2000 -notunepme -ntmpi 1 -ntomp 12 -nobackup -noconfout -pin on" with the "SYCL_CACHE_PERSISTENT=1 ONEAPI_DEVICE_SELECTOR=level_zero:gpu UR_L0_CMD_BUFFER_USE_IMMEDIATE_APPEND_PATH=1 " environment variables, and for the *with SYCL-Graph* case only also environment variable "GMX_CUDA_GRAPH=1"

[3] SYCL Grappa PME Intel Arc A770

- **Testing Date**: Performance results are based on testing by Codeplay as of 04/04/25 and may not reflect all publicly available updates.
- **Configuration Details**: Ubuntu 22.04.1 LTS. 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz CPU, GPU: Intel(R) A770 with driver version 1.6.32567+18. GROMACS branch https://gitlab.com/gromacs/gromacs/-/merge_requests/4954 compiled with CMake Options: -DGMX_GPU=SYCL, -DGMX_SYCL_ENABLE_GRAPHS=ON, -DGMX_FFT_LIBRARY=MKL, -DGMX_GPU_FFT_LIBRARY=MKL, -DGMX_SYCL_ENABLE_EXPERIMENTAL_SUBMIT_API=ON, -DGMX_GPU_NB_CLUSTER_SIZE=4, -DCMAKE_BUILD_TYPE=Release. DPC++ compiled with 27aae7ae2cb5a89a88b5a04af1ae8466d34e9e1a and oneMKL 2025.0
- **Run Command**: "gmx mdrun -pme gpu -pmefft gpu -s pme.tpr -nb gpu -update gpu -bonded gpu -nstlist 100 -notunepme -resetstep 2000 -ntomp 4 -nobackup -noconfout -pin on" with the "SYCL_CACHE_PERSISTENT=1 ONEAPI_DEVICE_SELECTOR=level_zero:gpu UR_L0_CMD_BUFFER_USE_IMMEDIATE_APPEND_PATH=1 " environment variables, and for the *with SYCL-Graph* case only also environment variable "GMX_CUDA_GRAPH=1"

Benchmarked Configurations (continued)

[4] SYCL Grappa PME Intel PVC

- **Testing Date**: Performance results are based on testing by Codeplay as of 04/04/25 and may not reflect all publicly available updates.
- **Configuration Details**: Ubuntu 22.04.5 LTS. Testing using Intel(R) Xeon(R) Gold 5418Y CPU, GPU: Intel(R) Data Center GPU Max 1100 with driver version 1.6.32567+18. GROMACS branch https://gitlab.com/gromacs/gromacs/-/merge_requests/4954 compiled with CMake Options: -DGMX_GPU=SYCL, -DGMX_SYCL_ENABLE_GRAPHS=ON, -DGMX_FFT_LIBRARY=MKL, -DGMX_GPU_FFT_LIBRARY=MKL, -DGMX_SYCL_ENABLE_EXPERIMENTAL_SUBMIT_API=ON, -DGMX_GPU_NB_CLUSTER_SIZE=8, -DCMAKE_BUILD_TYPE=Release,, -DGMX_GPU_NB_NUM_CLUSTER_PER_CELL_X=1. DPC++ compiled with 27aae7ae2cb5a89a88b5a04af1ae8466d34e9e1a and oneMKL 2025.0
- **Run Command**: "gmx mdrun -pme gpu -pmefft gpu -s pme.tpr -nb gpu -update gpu -bonded gpu -nstlist 100 -ntomp 4 -notunepme -resetstep 2000 -nobackup -noconfout -pin on" with the "SYCL_CACHE_PERSISTENT=1 ONEAPI_DEVICE_SELECTOR=level_zero:gpu UR_L0_CMD_BUFFER_USE_IMMEDIATE_APPEND_PATH=1 " environment variables, and for the *with SYCL-Graph* case only also environment variable "GMX_CUDA_GRAPH=1"

[5] CUDA Grappa PME NVIDIA A100

- **Testing Date**: Performance results are based on testing by Codeplay as of 03/05/25 and may not reflect all publicly available updates.
- **Configuration Details**: Ubuntu 22.04.5 LTS. Testing using Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz, GPU: NVIDIA A100 PCIE 40GB GPU memory. GROMACS branch https://gitlab.com/gromacs/gromacs/-/merge_requests/4954 compiled with CMake Options: -DGMX_GPU=CUDA -DGMX_FFT_LIBRARY=FFTW3 -DGMX_BUILD_OWN_FFTW=ON -DGMX_GPU_FFT_LIBRARY=cuFFT -DGMX_GPU_NB_CLUSTER_SIZE=8 -DCMAKE_BUILD_TYPE=Release -DGMX_OPENMP=ON with CUDA 12.6.2
- **Run Command**: "mdrun -pme gpu -pmefft gpu -s pme.tpr -nb gpu -update gpu -bonded gpu -notunepme -resetstep 2000 -nstlist 100 -ntomp 12 -nobackup -noconfout -pin on" with environment variable "GMX_CUDA_GRAPH=1" for the *with SYCL-Graph* case only .

References

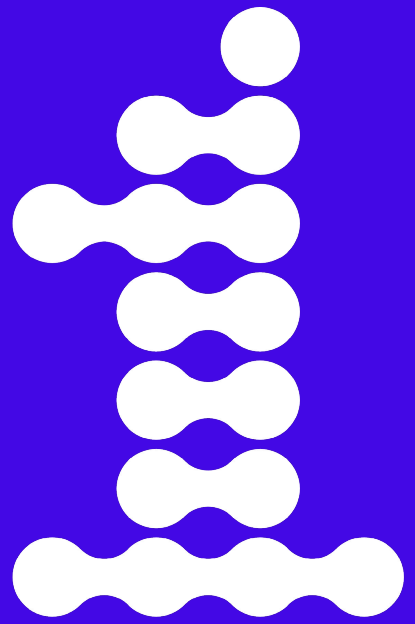
- IWOCL 2023 talk “Comparing the Performance of SYCL Runtimes for Molecular Dynamics Applications”
- IWOCL 2023 talk “Towards Deferred Execution of a SYCL Command Graph”.
- SYCL-Graph extension spec https://github.com/intel/llvm/blob/sycl/sycl/doc/extensions/experimental/sycl_ext_oneapi_graph.asciidoc
- SYCL-Graph usage guide https://github.com/intel/llvm/blob/sycl/sycl/doc/syclgraph/SYCLGraphUsageGuide.md

Acknowledgements

- James Brodman, Greg Lueck, Gordon Brown, Julian Miller, Maxime France-Pillois, Hugh Delaney, and Yejun Guo for their contributions and valuable feedback to the SYCL-Graph project
- Szilárd Páll and Alan Gray for their work on the CUDA Graph implementation in GROMACS

- This work was supported by the Intel Corporation via Intel oneAPI Academic Center of Excellence at KTH and by the DARE SGA1 project (EuroHPC JU, grant agreement 101202459)

oneAPI



oneAPI

Performance varies by use, configuration and other factors.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.

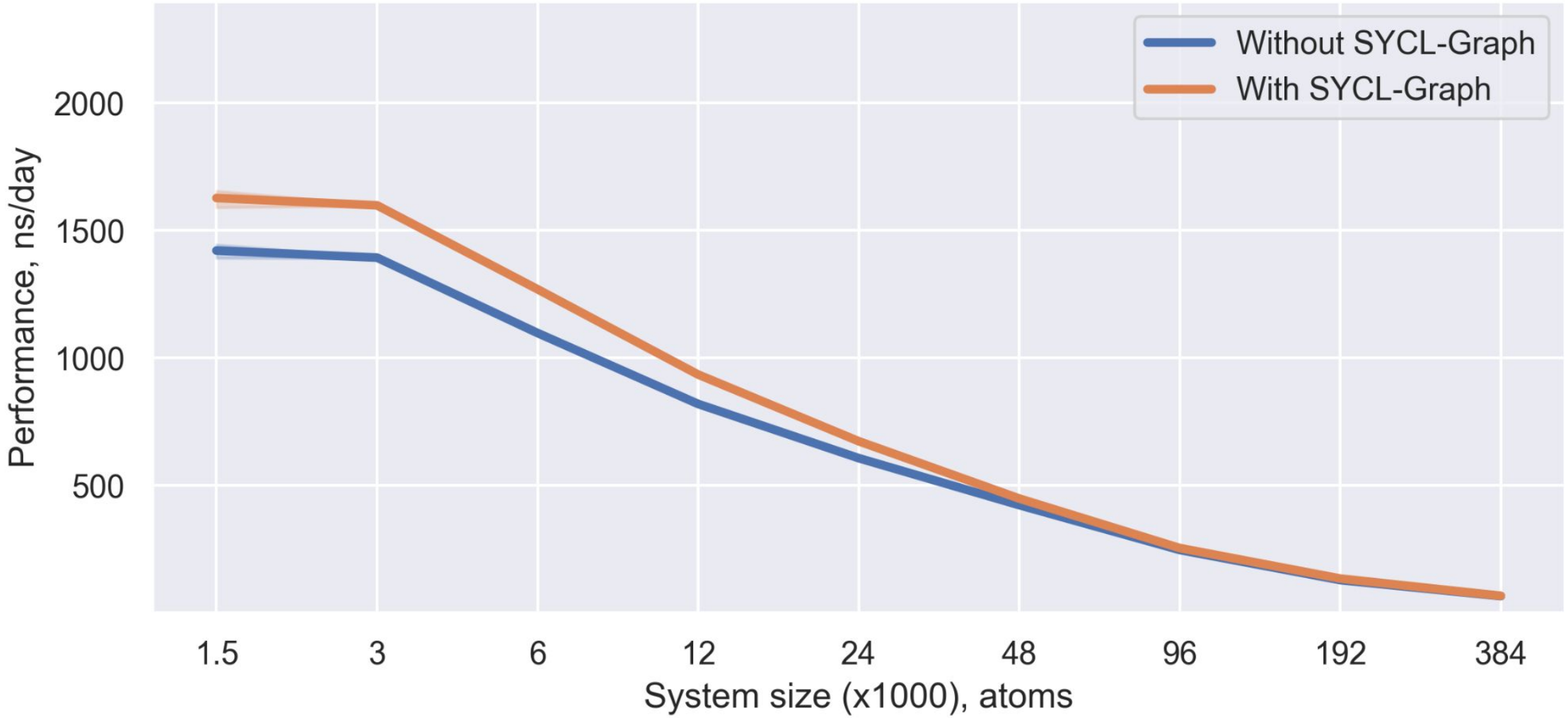
No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

Backup Slides

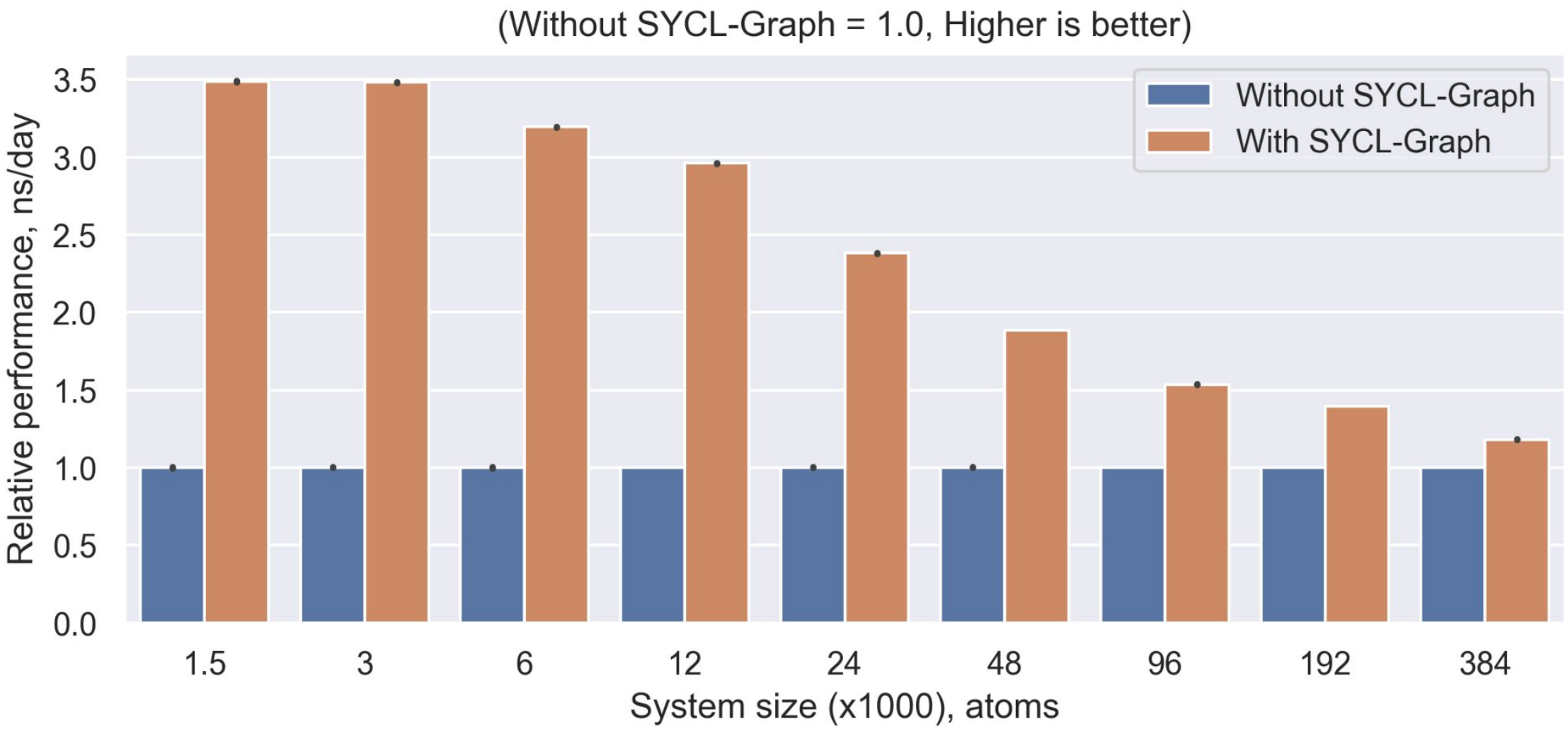
Absolute performance on Intel Arc B580



- For configuration see [2] on "Test Configuration" slide.
- Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.
- Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary.

oneAPI

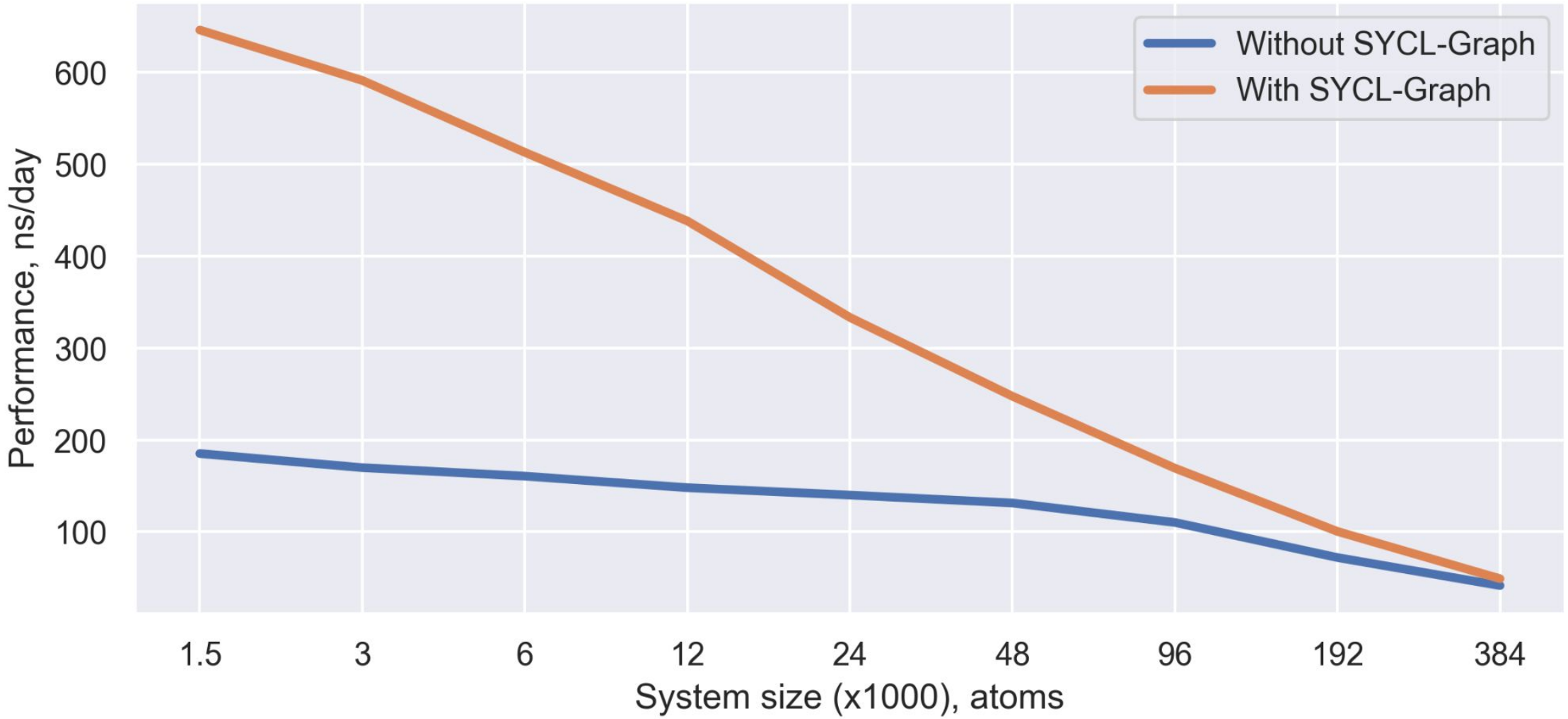
Relative performance on Intel Arc A770



- For configuration see [3] on "Test Configuration" slide.
- Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.
- Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary.

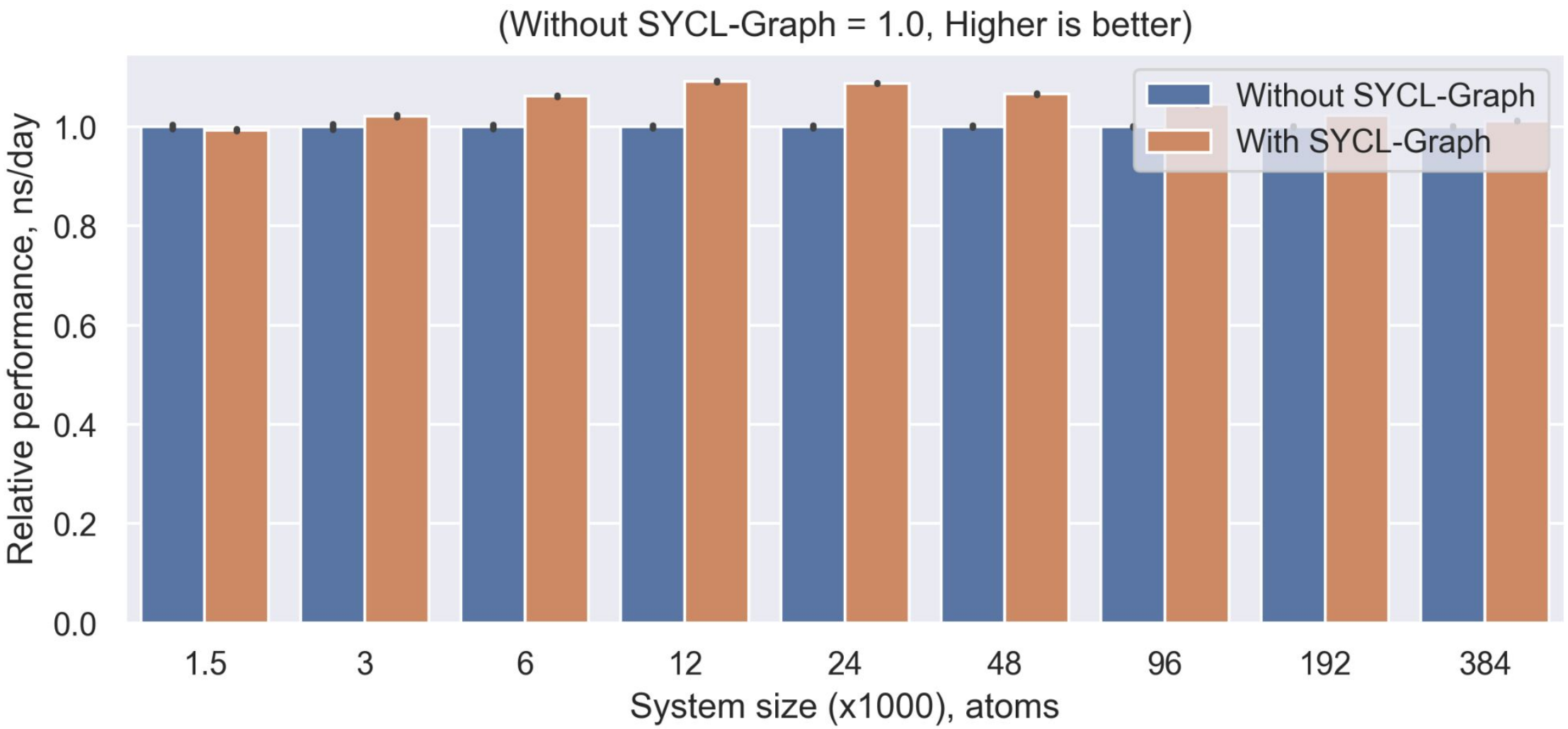
oneAPI

Absolute performance on Intel Arc A770



- For configuration see [3] on "Test Configuration" slide.
- Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.
- Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary.

Relative performance on Intel PVC 1100

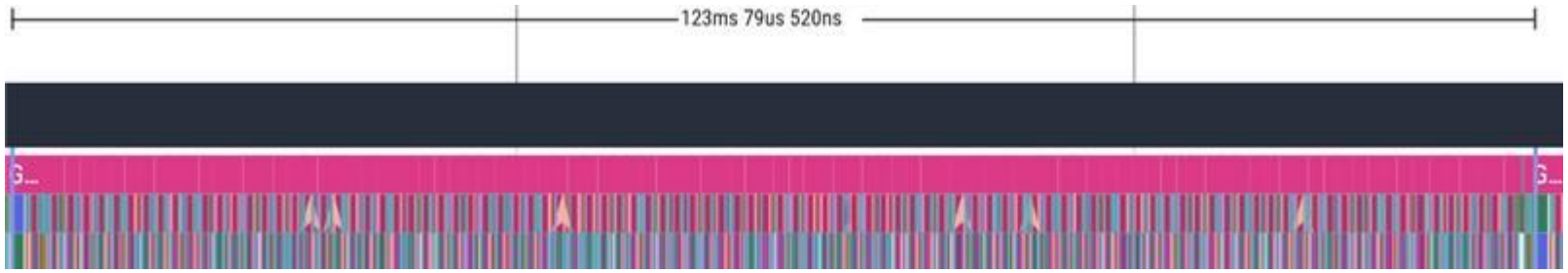


- For configuration see [4] on "Test Configuration" slide.
- Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.
- Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary.

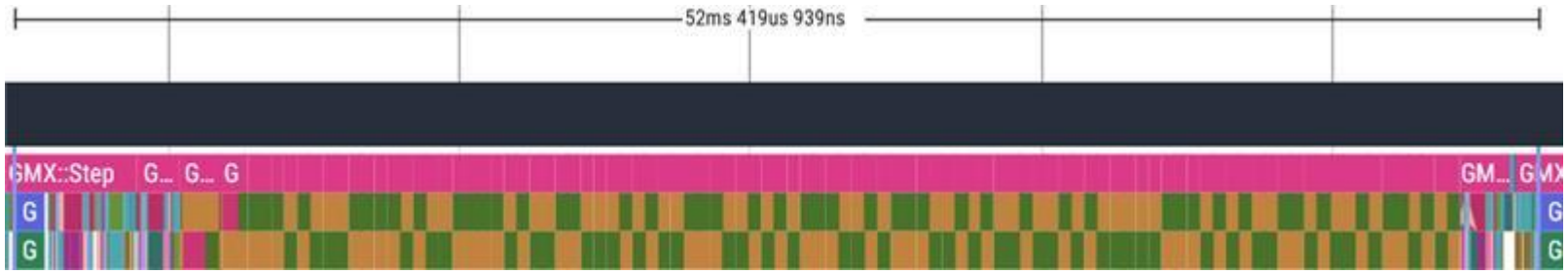
Intel Arc A770 3k System Size: 100 Steps

- 2x speed-up for small sizes
- SYCL-Graph trace has no long "Wait GPU state copy" steps, indicating that performance isn't GPU bound.
- Sample of timings:
  - Eager Step: ~1ms 200µs
  - Graph Launch Step: ~450µs
  - Graph Capture: ~100µs
  - Graph instantiate: ~700µs

Without SYCL-Graph



With SYCL-Graph



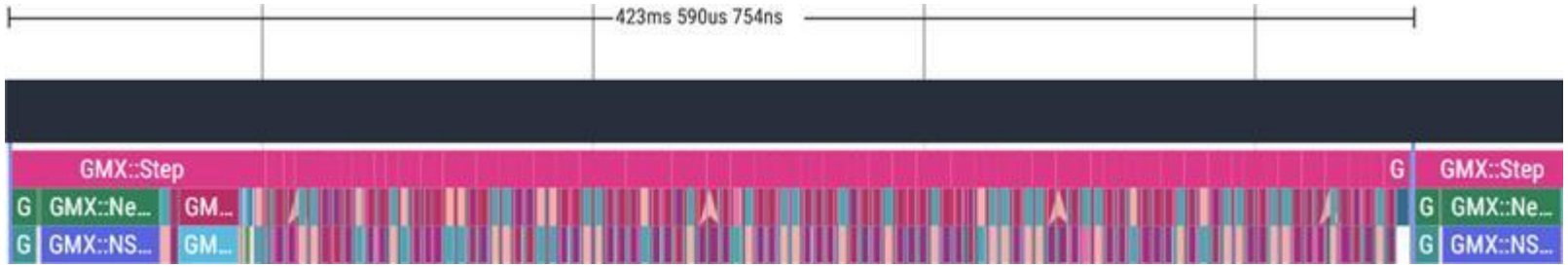
Taken from Perfetto output of A770 unitrace of benchmarked 3k system size configuration

oneAPI

Intel A770 192k System Size: 100 Steps

- 30% performance improvements at 192k size.
- SYCL-Graph trace has no long "Wait GPU state copy" steps, indicating that performance isn't GPU bound.
- Sample of timings:
  - Eager Step: ~3ms 100µs
  - Graph Launch Step: ~2ms 600µs
  - Graph Capture: ~100µs
  - Graph instantiate: ~2ms 500µs

Without SYCL-Graph



With SYCL-Graph

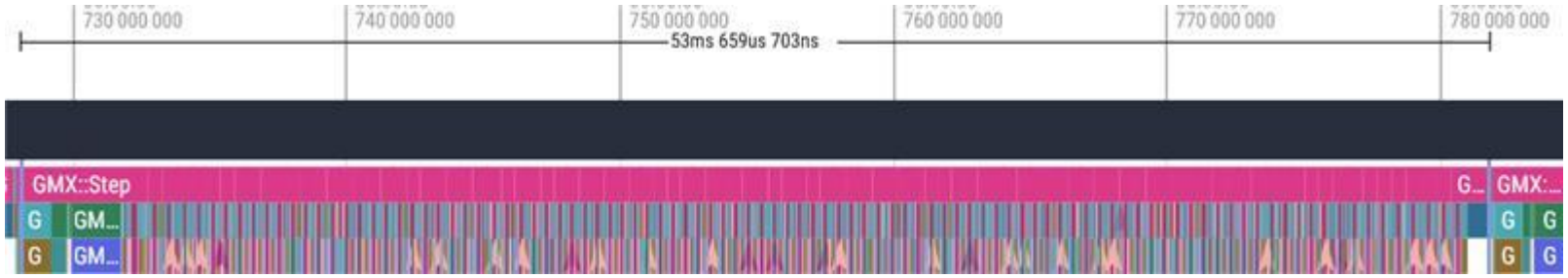
Taken from Perfetto output of A770 unitrace of benchmarked 192k system size configuration

oneAPI

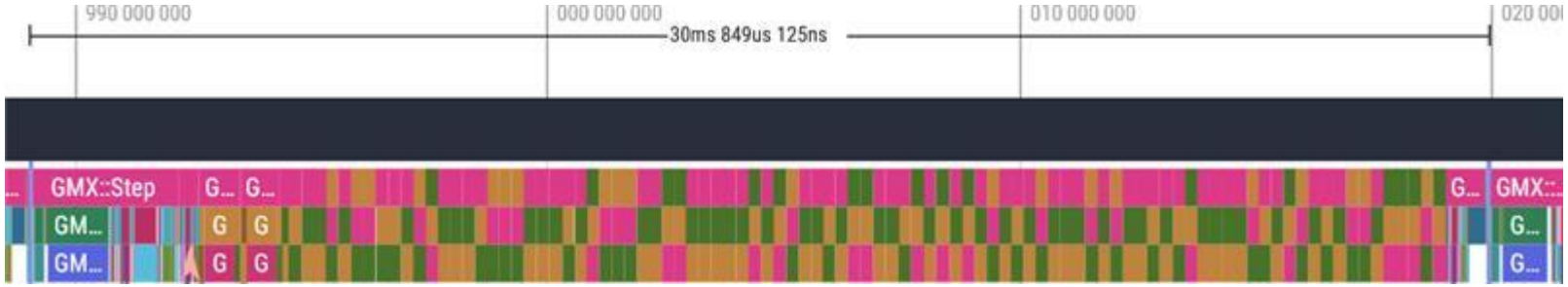
Intel PVC 3k System Size: 100 Steps

- Up to 10% performance benefit across all sizes.
- SYCL-Graph trace has no long "Wait GPU state copy" steps, indicating that performance isn't GPU bound.
- Sample of timings:
  - Eager Step: ~450µs
  - Graph Launch Step: ~260µs
  - Graph Capture: 150µs
  - Graph instantiate: ~800µs

Without SYCL-Graph



With SYCL-Graph



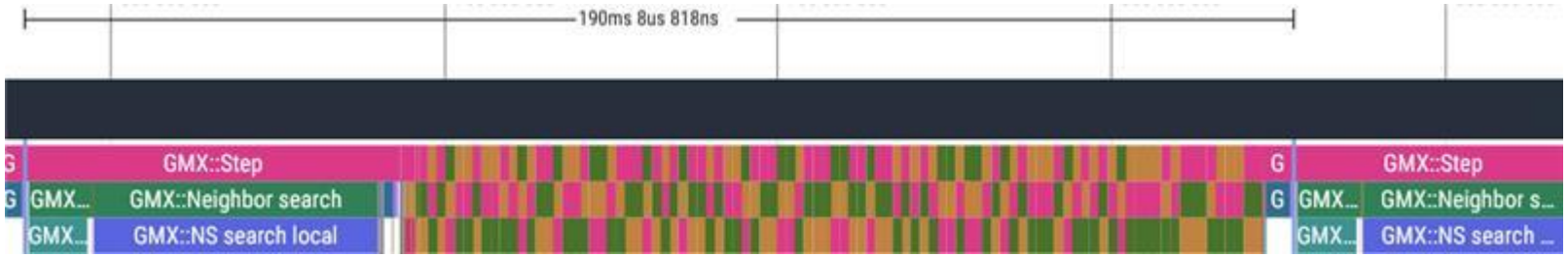
Taken from Perfetto output of PVC unitrace of benchmarked 3k system size configuration

oneAPI

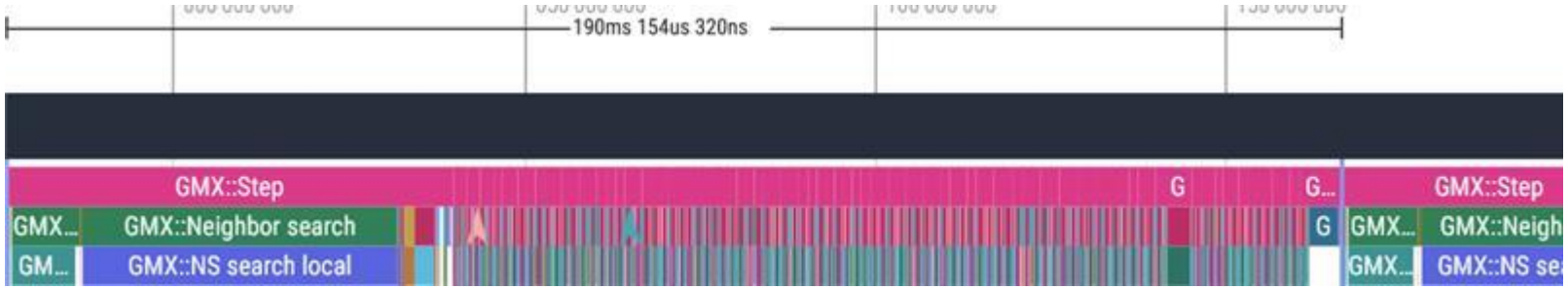
Intel PVC 192k System Size: 100 Steps

- Negligible improvement (3%).
- SYCL-Graph trace has no long "Wait GPU state copy" steps, indicating that performance isn't GPU bound
- Sample of timings:
  - Eager Step: ~1ms 200µs but some ~3ms FFT operation steps
  - Graph Launch Step: ~1ms 300µs
  - Graph Capture: ~150µs
  - Graph instantiate: ~2ms 500µs

Without SYCL-Graph



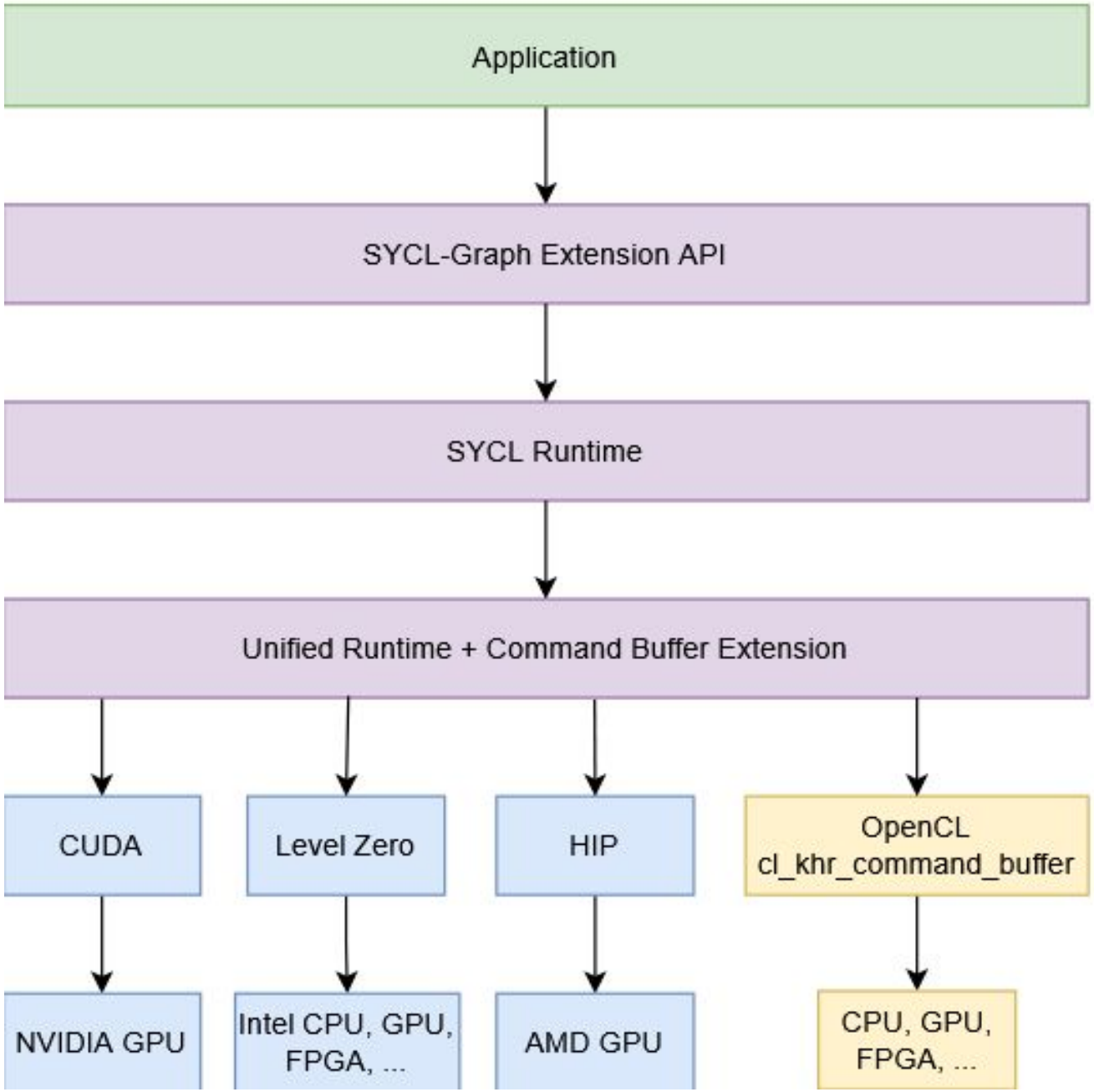
With SYCL-Graph

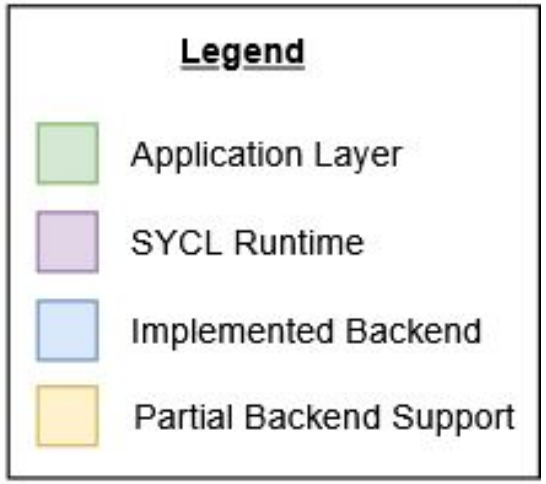


Taken from Perfetto output of PVC unitrace of benchmarked 192k system size configuration

oneAPI

SYCL-Graph Architecture





oneAPI

OpenCL Support Status: Commands

Not all command types are supported

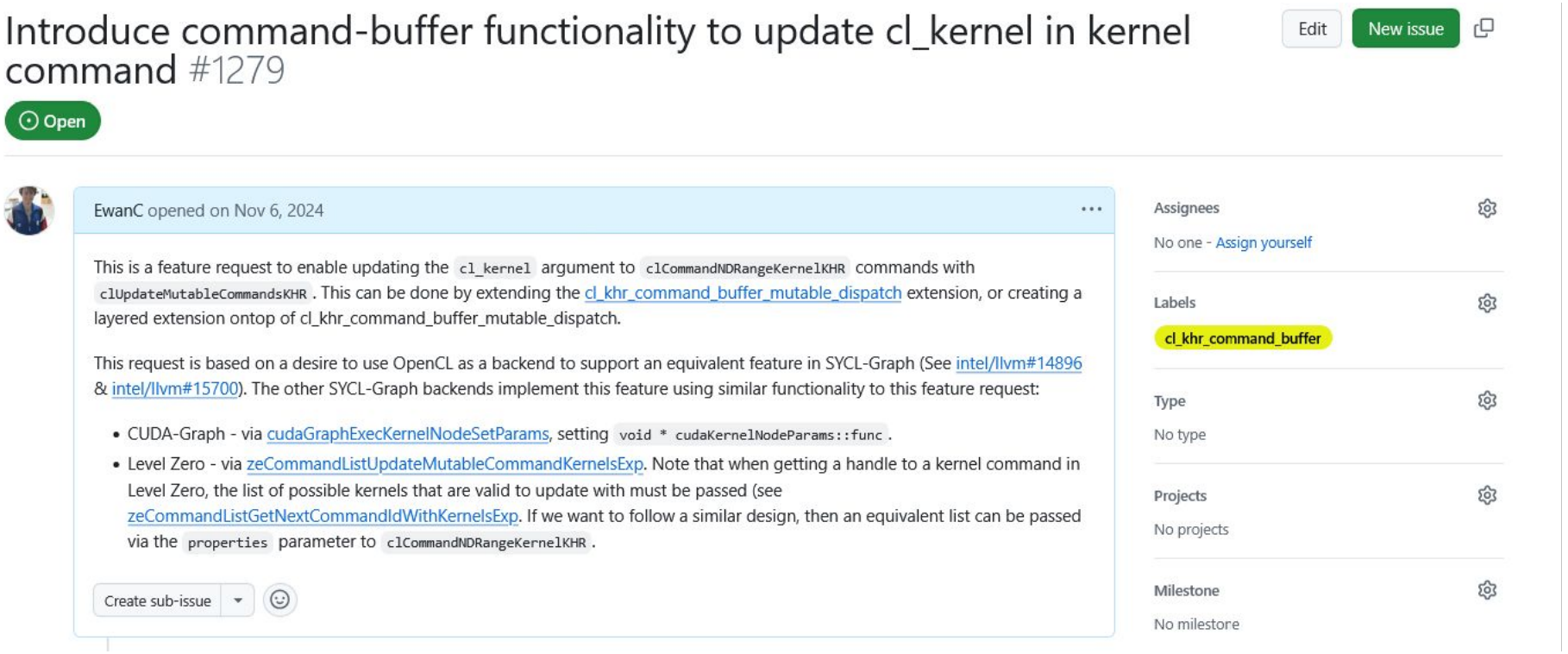
The types of commands which are unsupported, and lead to this exception are:

- `handler::copy(src, dest)` - Where `src` is an accessor and `dest` is a pointer. This corresponds to a memory buffer read command.
- `handler::copy(src, dest)` - Where `src` is an pointer and `dest` is an accessor. This corresponds to a memory buffer write command.
- `handler::copy(src, dest)` or `handler::memcpy(dest, src)` - Where both `src` and `dest` are USM pointers. This corresponds to a USM copy command.
- `handler::fill(ptr, pattern, count)` - This corresponds to a USM memory fill command.
- `handler::memset(ptr, value, numBytes)` - This corresponds to a USM memory fill command.
- `handler::prefetch()`.
- `handler::mem_advise()`.

- USM command limitations should be addressed by cl_khr_unified_svm.
- Buffer read/write commands being discussed by OpenCL WG in https://github.com/KhronosGroup/OpenCL-Docs/issues/1281

OpenCL Support Status: Update

- SYCL-Graph update enables the user to update a kernel command to a different kernel binary.
- Not currently supported in OpenCL https://github.com/KhronosGroup/OpenCL-Docs/issues/1279



Introduce command-buffer functionality to update cl_kernel in kernel command #1279

Edit | New issue

Open

EwanC opened on Nov 6, 2024

This is a feature request to enable updating the `cl_kernel` argument to `clCommandNDRangeKernelKHR` commands with `clUpdateMutableCommandsKHR`. This can be done by extending the cl_khr_command_buffer_mutable_dispatch extension, or creating a layered extension ontop of cl_khr_command_buffer_mutable_dispatch.

This request is based on a desire to use OpenCL as a backend to support an equivalent feature in SYCL-Graph (See intel/llvm#14896 & intel/llvm#15700). The other SYCL-Graph backends implement this feature using similar functionality to this feature request:

- CUDA-Graph - via cudaGraphExecKernelNodeSetParams, setting `void * cudaKernelNodeParams::func`.
- Level Zero - via zeCommandListUpdateMutableCommandKernelsExp. Note that when getting a handle to a kernel command in Level Zero, the list of possible kernels that are valid to update with must be passed (see zeCommandListGetNextCommandIdWithKernelsExp. If we want to follow a similar design, then an equivalent list can be passed via the `properties` parameter to `clCommandNDRangeKernelKHR`.

Create sub-issue

Assignees: No one - Assign yourself

Labels: cl_khr_command_buffer

Type: No type

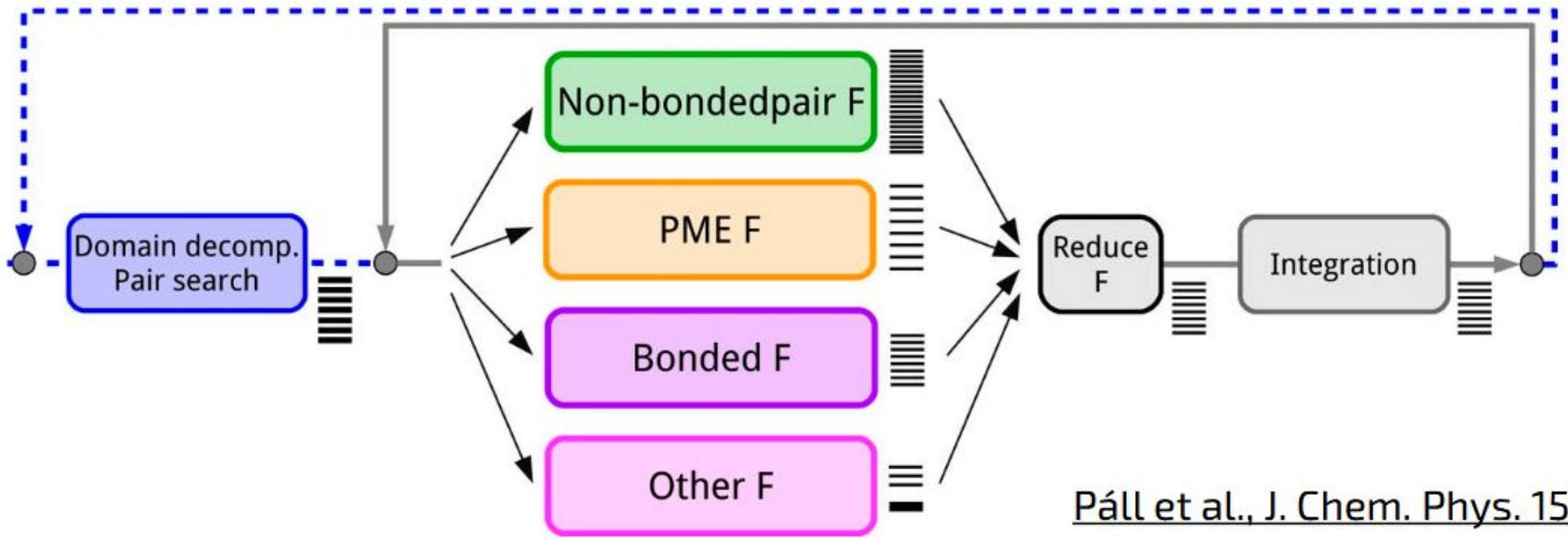
Projects: No projects

Milestone: No milestone

oneAPI

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)

Llama

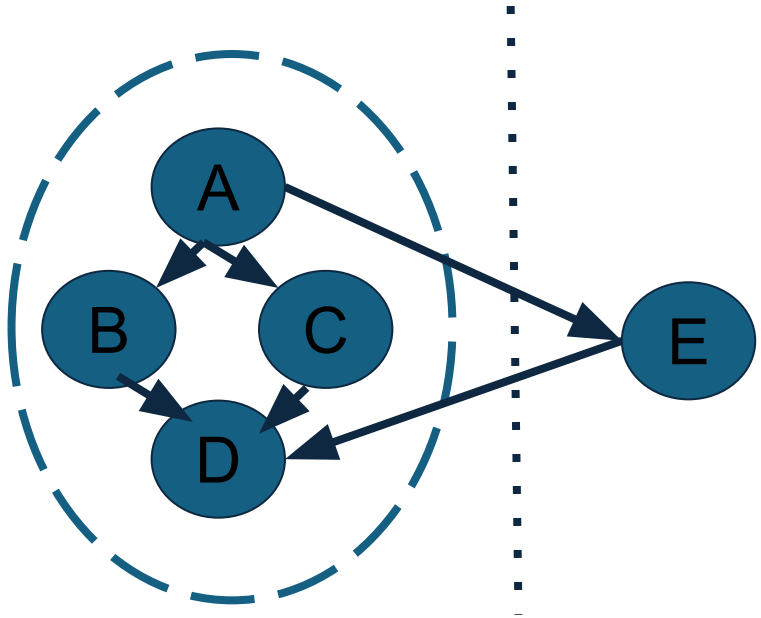
```
sycl_ex::command_graph model_sycl_graph(*(sycl_ctx->stream()));
model_sycl_graph.begin_recording(*(sycl_ctx->stream()));
ggml_backend_sycl_graph_compute_impl(sycl_ctx, cgraph);
model_sycl_graph.end_recording();

if (!sycl_ctx->exec_graph) {
    auto exec_graph = model_sycl_graph.finalize({sycl_ex::property::graph::updatable{}});
    sycl_ctx->exec_graph = std::make_unique<
        sycl_ex::command_graph<sycl_ex::graph_state::executable>>(exec_graph);
} else {
    try {
        sycl_ctx->exec_graph->update(model_sycl_graph);
        GGML_SYCL_DEBUG("[SYCL-GRAPH] update success\n");
    } catch (sycl::exception const & e) {
        GGML_SYCL_DEBUG("[SYCL-GRAPH] Exception when updating graph, %s\n", e.what());
        auto exec_graph = model_sycl_graph.finalize({sycl_ex::property::graph::updatable{}});
        sycl_ctx->exec_graph = std::make_unique<
            sycl_ex::command_graph<sycl_ex::graph_state::executable>>(exec_graph);
    }
}

sycl_ctx->stream()->ext_oneapi_graph(*(sycl_ctx->exec_graph));
```

Moving to stable API

- Evaluation of API use in applications. E.g. GROMACS/Llama/Kokkos/PyTorch have the mechanisms they need to get performance.
- Potentially split extension into multiple extensions. A stable API base extension with experimental extensions layered on top.
- Project goal to bring to Khronos SYCL WG as a KHR extension.

Features - External Synchronization

Ability to synchronize externally to/from nodes in a graph to commands outside of the graph



Features - Graph Owned Memory

- Support the sycl_ext_oneapi_async_memory_alloc extension.
- Allocation and free node types in a graph.
- Memory allocated on finalize time when graph is known, to allow for reuse.