IXPUG 2025

SYCL Graph

Reducing Kernel Launch overhead for Intel GPUs and more

Pablo Reble, Intel Corporation

Executive summary

SYCL Graph is a oneAPI vendor extension to the SYCL 2020 specification

Portable interface that improves performance of SYCL applications across multiple backends and devices.

Explicit lazy execution model guarantees batching and deferred execution of commands.

Access through:

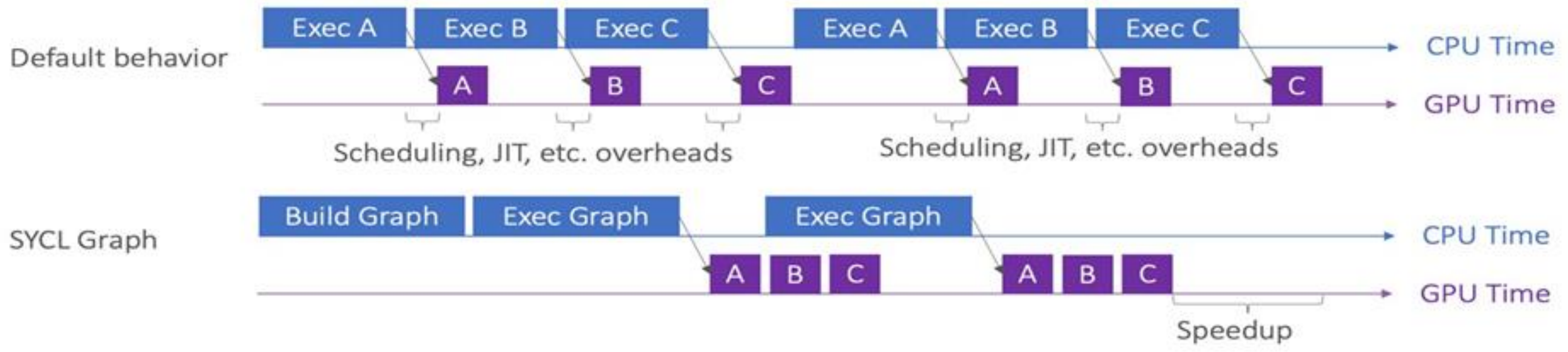
- Framework integration
- SYCLomatic
- Direct Programming

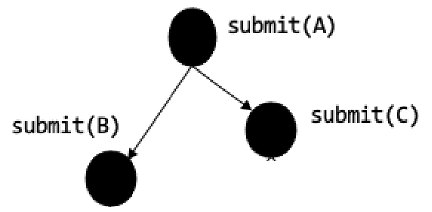
Agenda

- Motivation
- Integration in Applications and Frameworks
  - Case-study: GROMACS
- SYCLomatic
- Direct Programming: SYCL example
- Summary and Future Work

Motivation

- Command Graph is defined once and submitted as many times as required
- Increasing Throughput when kernel launch latency is a bottleneck.
- Enable optimizations across the defined graph (eliminate redundant edges, customized scheduling, …)



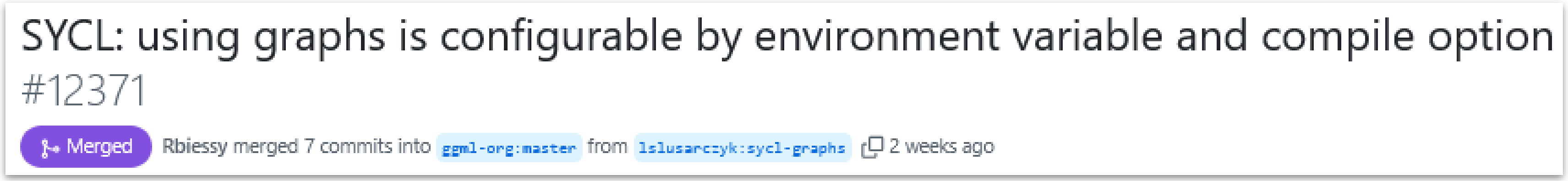


How to get access?

- Part of Intel® oneAPI C++/DPC++ Compiler
  - basic support for Intel GPUs since 2024.0.
  - CUDA support is available since 2024.1*
  - HIP support is available since 2024.2*
- Open-source
  - Joint project with Codeplay
  - Available on GitHub: https://github.com/intel/llvm
- Experimental API: Feedback is appreciated!

Overview: Integration status for Applications/Frameworks

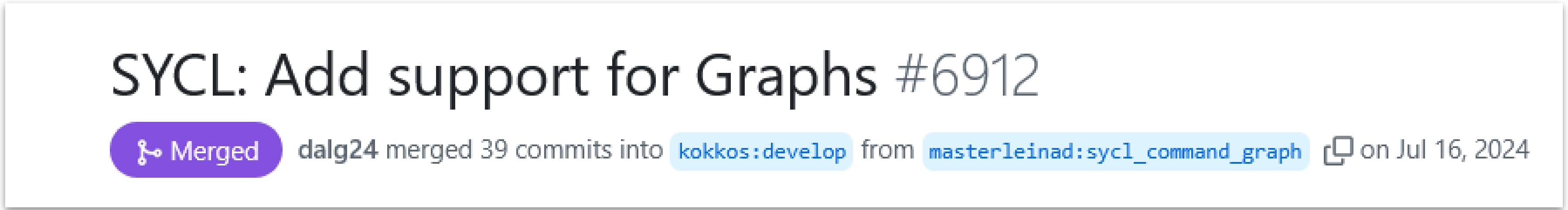
LLama.cpp


SYCL: using graphs is configurable by environment variable and compile option #12371

Merged Rbiessy merged 7 commits into ggml-org:master from lslusarczyk:sycl-graphs 2 weeks ago

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

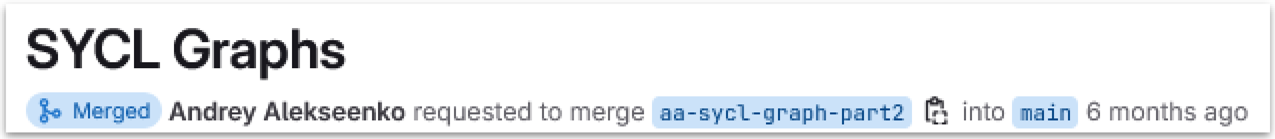
Kokkos


SYCL: Add support for Graphs #6912

Merged dalg24 merged 39 commits into kokkos:develop from masterleinad:sycl_command_graph on Jul 16, 2024

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

GROMACS


SYCL Graphs

Merged Andrey Alekseenko requested to merge aa-sycl-graph-part2 into main 6 months ago

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

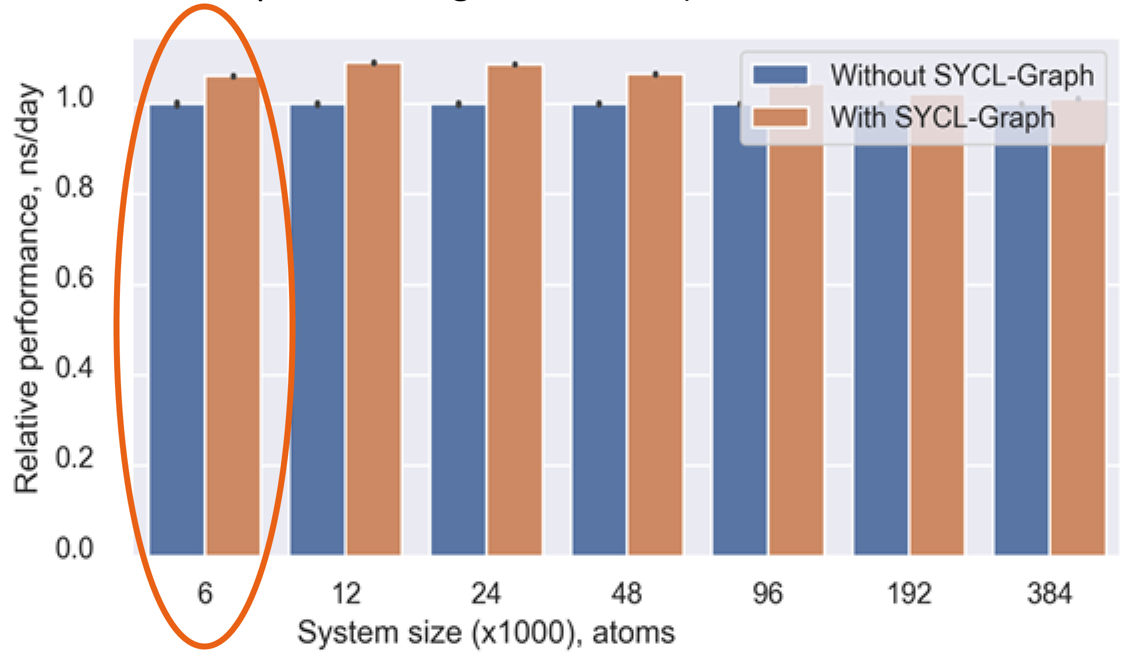
Case study: GROMACS

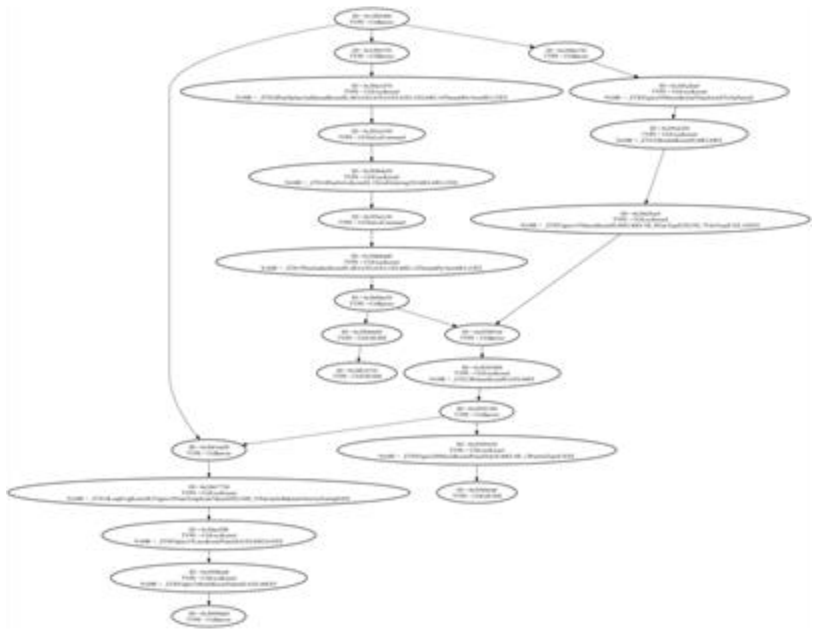
Grappa PME on Intel GPUs

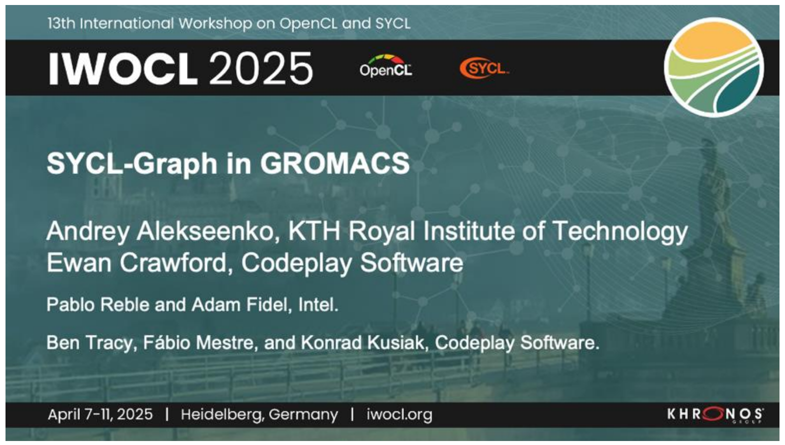
*Other names and brands may be claimed as the property of others. SYCL is a trademark of the Khronos Group Inc.

Results for GROMACS - Grappa PME

Relative performance on Intel® Data Center GPU Max Series,
(Without SYCL-Graph = 1.0, Higher is better)







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.

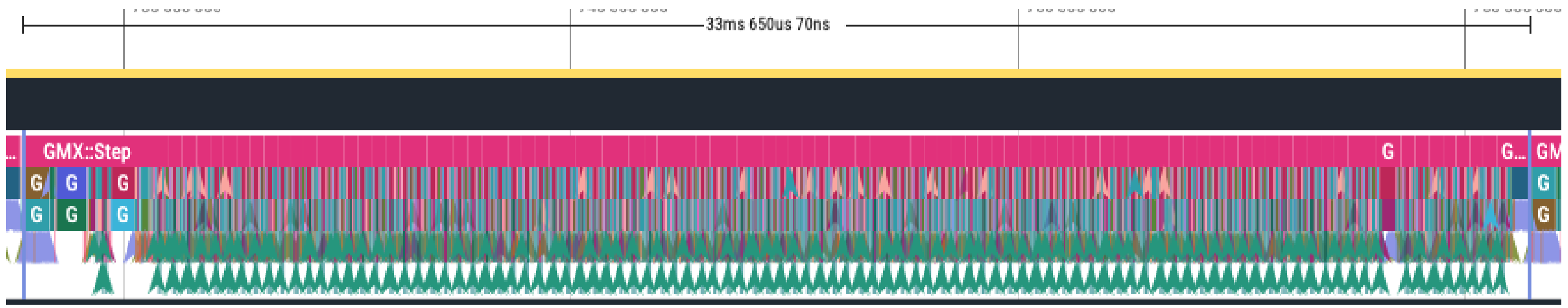
April 7-11, 2025 | Heidelberg, Germany | iwocl.org

- 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.

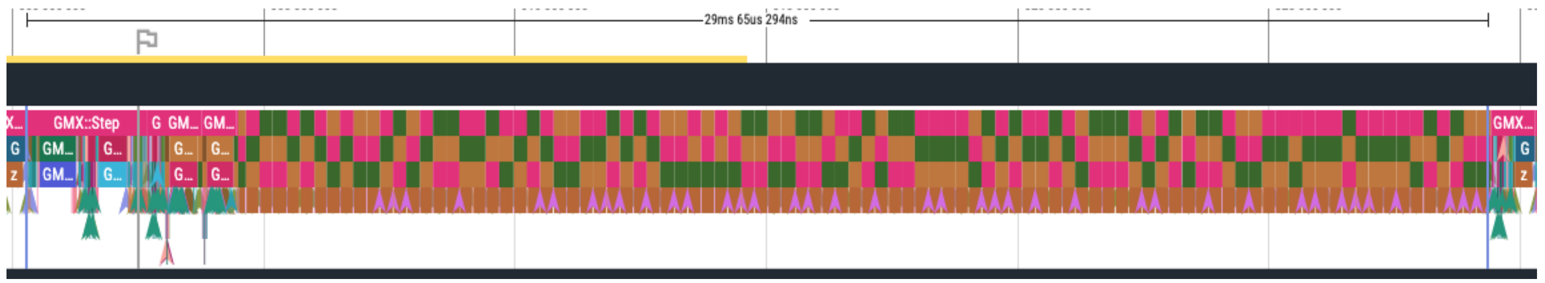
Grappa PME 6k System Size: 100 Steps

- 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: ~300µs
  - Graph Launch Step: ~270µs
  - Graph Capture: 150µs
  - Graph instantiate: ~800µs

Without SYCL-Graph



With SYCL-Graph



Traces with Perfetto UI with itt/unitrace of benchmarked 6k system size configuration

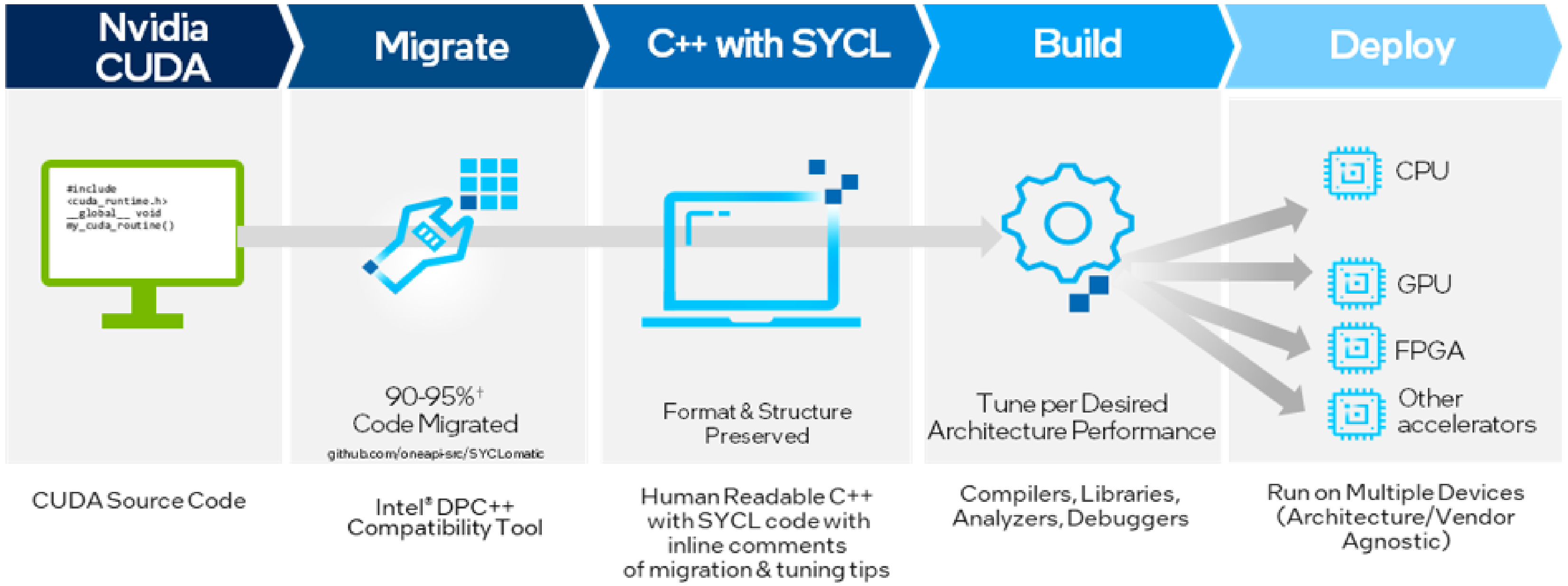
https://github.com/intel/pti-gpu

SYCLomatic

Migration from CUDA* Graphs to SYCL Graph

Access SYCL Graph with Tool support

- SYCLomatic aids in migrating code from CUDA app
- SYCL Graph is an open and multi-platform alternative to CUDA* Graphs
- Basic support for SYCL Graph



Nvidia CUDA
Migrate
C++ with SYCL
Build
Deploy
#include <cuda_runtime.h> __global__ void my_cuda_routine()
90-95%† Code Migrated
github.com/oneapi-src/SYCLomatic
Format & Structure Preserved
Tune per Desired Architecture Performance
CPU
GPU
FPGA
Other accelerators
CUDA Source Code
Intel® DPC++ Compatibility Tool
Human Readable C++ with SYCL code with inline comments of migration & tuning tips
Compilers, Libraries, Analyzers, Debuggers
Run on Multiple Devices (Architecture/Vendor Agnostic)

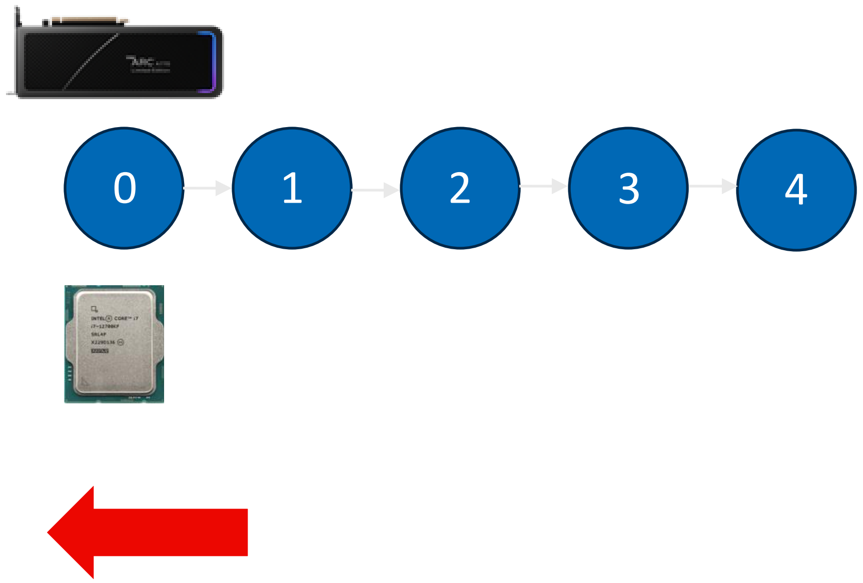
† Intel estimates as of March 2023. Based on measurements on a set of 85 HPC benchmarks and samples, with examples like Rodinia, SHOC, PENNANT. Results may vary.
*Other names and brands may be claimed as the property of others. SYCL is a trademark of the Khronos Group Inc.

Direct Programming

Using SYCL Graph's Record & Replay

SYCL Hello World

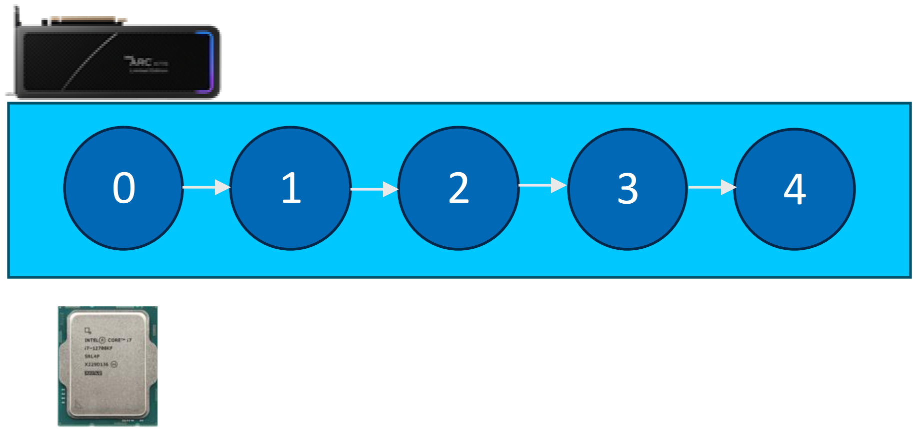
```
#include <sycl/sycl.hpp>


…

sycl::queue q{{sycl::property::queue::in_order{}}};

auto ptr = sycl::malloc_device<int>(1024,q);



for(size_t i=0; i<5; i++) {

  q.submit([&](sycl::handler& h){ h.single_task([=](){ptr[0]=i;});});

}




q.wait();

…
```



*Other names and brands may be claimed as the property of others. SYCL is a trademark of the Khronos Group Inc.

SYCL Graph Hello World

```
#include <sycl/sycl.hpp>
#include <sycl/ext/oneapi/experimental/graph.hpp>
…
sycl::queue q{{sycl::property::queue::in_order{}}};
auto ptr = sycl::malloc_device<int>(1024,q);
sycl::ext::oneapi::experimental::command_graph g{q};
g.begin_recording(q);
for(size_t i=0; i<5; i++) {
  q.submit([&](sycl::handler& h){ h.single_task([=](){ptr[0]=i;});});
}
g.end_recording();
auto exec=g.finalize();
q.ext_oneapi_graph(exec);
q.wait();
…
```



$$\begin{tikzcd} 0 \arrow[r] & 1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \end{tikzcd}$$

Recording Mode

Summary and future work

- SYCL Graph has a portable interface that improves performance by reducing kernel launch overhead.
- Easy integration into AI/HPC Frameworks that is portable across multiple devices.

- Future Work:
  - Update functionality for Intel GPUs
  - Asynchronous memory allocations.
  - Support of oneAPI libraries with Record&Replay (MKL,DNN,CCL,…)

Available on GitHub: https://github.com/intel/llvm

References

SYCL Graph specification

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

IWOCL 2023 Talk on SYCL Graph

https://www.youtube.com/watch?v=aOTAmyr04rM

Codeplay Blog article

https://codeplay.com/portal/blogs/2024/01/22/sycl-graphs

Thank You

Disclaimers

Benchmarked Configurations

[1] 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"