Learning SYCL: challenges and opportunities

Roberto Di Remigio Eikås oneAPI Developer Summit, 9 May 2022



EuroHPC JU Systems



https://eurohpc-ju.europa.eu/discover-eurohpc-ju

EuroPPC National Competence Centre sweden

LUMI

Sustained perf: 375 petaflops Peak perf: 552 petaflops AMD EPYC CPUs, AMD Instinct GPUs

Leonardo

Sustained perf: 249.4 petaflops Peak perf: 322.6 petaflops Intel Ice-Lake (Booster), Intel Sapphire Rapids (data-centric), NVIDIA Ampere GPUs

Vega 6,8 petaflops AMD EPYC CPUs, NVIDIA A100 GPUs

Karolina 9,13 petaflops AMD EPYC CPUs, NVIDIA A100 GPUs

MeluXina: 10 petaflops HPL AMD EPYC CPUs, NVIDIA A100 GPUs

Discoverer: 4,44 petaflops **AMD** EPYC CPUs

Deucalion: 10 petaflops ARM A64FX CPUs, AMD CPUs, NVIDIA GPUs

EURO

ABOUT US 🔻	OUR SERVICES -	NEWS 🕶	Login
------------	----------------	--------	-------







The Swedish EuroCC Hub for High-Performance Computing



TRAINING

Providing training on GPU usage, AI and HPC optimisations as well as on usage of HPC in scientific disciplines such as Life sciences, Chemistry, Climate modelling, Engineering and more.



INDUSTRY – PUBLIC ADMINISTRATION

Assisting SMEs, large businesses and public administration in delivering competitive benefits from advanced HPC and build awareness about HPC and AI/HPDA competences and identifying strategies for technology transfer from academia



PROPOSAL SUPPORT

Helping researchers both in industry and

academia with application forms to apply

for access to EuroHPC IU (pre)exa-scale

system application support.



EUROHPC JU SYSTEMS ACCESS SUPPORTED SOFTWARE

Support users move to pre-exascale systems. Set up mechanisms to educate users about technical requirements for scaling and provide consulting in HPC, GPU acceleration AI/HPDA and data

Want to get *100x* more CPU/GPU power?

Get up to a year access to European *high-performance* computers to *develop*, *test*, *or run* your code for free!

Do you need compute capacity for your Jupyter hub solutions, or for data analysis? Perhaps you use simulations in engineering or life science or any other domain? Check your needs below!

Unsure? We can help you!

IUST CONTACT US





ENCCS training portfolio

Targeting academia, industry, and public sector:

- Programming models
- Domain-specific software
- Hackathons and bootcamps
- How to apply for EuroHPC JU resources

Training material:

- Available online https://enccs.se/training-resources/
- Creative commons: free to reuse and modify, with acknowledgement



Why learn SYCL? HPC heterogeneity

EuroHPC JU Systems and **beyond**:

- Computational power delivered by heterogeneous CPU+GPU architectures
- Heterogeneity also in CPU and GPU vendors

Performance and performance portability:

- GPU ports are costly
- Multiple ports unfeasible



EuroHPC JU Systems



Why learn SYCL? Portability

Low-level, hardware-specific:

- Code duplication
- Time-consuming maintenance.
- Vendor lock-in

pragma-based:

- Highly dependent on compiler support
- Readability and code divergence



Why learn SYCL? Portability

SYCL:

- Standard C++
- High-level abstractions
- Multiple backends





Our goals in designing a SYCL workshop

- Create a training offer of interest for all EuroHPC-JU countries
- Build up competence in cutting-edge programming frameworks
- Collaboration with other competence centres
- Show that SYCL is viable on today's HPC machines







Target audience



Students, researchers, engineers, and programmers with no previous experience with SYCL.

Prerequisites

- A working knowledge of recent C++ standards
- Some familiarity with C++17
- No knowledge of CUDA/HIP/OpenMP/OpenACC!

Overarching philosophy:

- We are **not** teaching the SYCL standard
- Learn by doing: plenty of hands-on sessions



Learning goals

- How to define parallel work? how to offload it to devices?
- Write hardware-agnostic code to express parallelism
- Use buffers and accessors to handle memory across devices
- Evaluate drawbacks and advantages of unified shared memory
- Run examples and exercises on **EuroHPC-JU hardware**



Workshop set up

SYCL



Search docs

Setting up your system

THE LESSON

What is SYCL?

Device discovery

Queues, command groups, and kernels

Data management with buffers and accessors

Data management with unified shared memory

Expressing parallelism with SYCL

📸 » SYCL

G Edit on GitHub

SYCL

SYCL is a C++ abstraction layer for programming heterogeneous hardware with a **single-source approach**. SYCL is built on top of **standard** ISO C++17 and provides a **high-level** and **cross-platform** route for heterogeneous programming.

In this workshop, you will learn to:

- Use the hipSYCL compiler to generate executable for *multiple* hardware targets.
- Write hardware-agnostic code to express parallelism using the queue, command group, and kernel abstractions.
- Use buffer and accessors to handle memory across devices.
- Evaluate drawbacks and advantages of unified shared memory.

© Prerequisites

Before attending this workshop, please make sure that you have access to a machine with the hipSYCL compiler (v0.9.1) and CMake (>=3.14) installed.

This workshop is organized in collaboration with CSC and IZUM. We will work on the exercises using the Vega supercomputer, a EuroHPC Joint Undertaking petascale system.

https://enccs.github.io/sycl-workshop/

Website in sphinx-lesson format

- No slides. All content spelled out on the webpage => Better for self-learning
- Based on Carpentries and CodeRefinery formats



Workshop set up

ENCCS / sycl-workshop (Public) Code O Issues 7 11 Pull reques	ts 🕟 Actions 🕕 Security 🗠 Insights		다 Notifications 💱 Fork 2 ☆ Star 6
t ^p main → t ^p 2 branches	5 🚫 1 tag	Go to file Code →	About SYCL materials for ENCCS workshop ¢ enccs.github.io/sycl-workshop/
 .github/workflows content .gitignore LICENSE LICENSE.code 	First commit Formatting Save work First commit First commit	10 months ago 5 months ago 6 months ago 10 months ago 10 months ago	workshop syd □ Readme ≰t View license ☆ 6 stars ③ 6 watching ¥ 2 forden
Image: Markefile Imag	First commit Lesson name is SYCL First commit Add package so that I can build locally	10 months ago 7 months ago 10 months ago 6 months ago	Releases 1 © First iteration: November 8-9 (Latest) on Nov 22, 2021

All content in GitHub repo

- github.com/ENCCS/sycl-workshop
- Easily accessible
- Open for contributions
- Use issues to keep track of what to improve



Lesson structure

30 min	What is SYCL?
30 min	Device discovery
30 min	Queues, command groups, and kernels
30 min	Data management with buffers and accessors
30 min	Data management with unified shared memory
30 min	Expressing parallelism with SYCL: basic data-parallel kernels
30 min	Expressing parallelism with SYCL: nd-range data-parallel kernels
40 min	The task graph: data, dependencies, synchronization
40 min	Heat equation mini-app
30 min	Using sub-groups in SYCL
30 min	Profiling SYCL applications
40 min	Buffer-accessor model vs unified shared memory

<u>First run</u> 8-9 November 2021. Using **VEGA** <u>https://en-vegadocs.vega.izum.si/</u> <u>Second run</u> 19-21 April 2022. Using **Karolina** <u>https://docs.it4i.cz/karolina/introduction/</u>

Max 4 hours per day

- Reduce learning overload
- Allow new knowledge to sediment

Episodes

- Maximum 40 minutes
- At least 50% time for hands-on exercises
- One episode per main concept

Episodes: keep participants engaged in active learning

Starts with:

- Questions: What? How? Why?
- Objectives: learning goals at a glance

? Questions

• How do we organize work in a SYCL application?

Objectives

- Learn about queues to describe ordering of operations.
- Command groups.
- Understand that kernels are units of parallelism in SYCL.

Ends with:

• Keypoints: take-home messages

Keypoints

- Buffers and accessors delegate memory management issues to the SYCL runtime.
- SYCL lets you abstract away the intricacies of host-device data dependencies.
- It can be hard to adapt an existing code to the buffer-accessor model.
- There might be performance overhead when adopting the buffer-accessor model.



Episodes: keep participants engaged in active learning

Visual cues

 Signposting of activities during the lesson: function signatures, typealongs, hands-on exercises

Hello, world" with SYCL

You can find the file with the complete source code in the content/code/day-1/00_hello folder. Worry not about the details in the code, we will dig into what is happening here at great length during the rest of the lesson.

Some buffer constructors 1. We can construct a buffer using just a range. Data must be initialized in some other fashion: buffer(const range<dimensions> &bufferRange, const property_list &propList={}); 2. We can set the data at construction passing a host pointer and a range : buffer(T *hostData, const range<dimensions> &bufferRange, const property_list &propList={});

#include <iostream>

#include <sycl/sycl.hpp>

using namespace sycl;

int
main()

const std:/string secret { "Ifmmn-lynsme\"\A121(nltnssz-l

AXPY with SYCL buffers and accessors

We will now write an AXPY implementation in SYCL, using the buffer and accessor API. This will be a generic implementation: it will work with any arithmetic type, thanks to C++ templates.

Don't do this at home, use optimized BLAS!



Episodes: keep participants engaged in active learning

"Lecture"

- Use clear prose, in active voice
 - Not a university lecture
 - Not a manual
- Tables, figures, schematics
 - Distill information
 - Easier to recall concepts
- Exercises must be self-explanatory
 - Solution available within the repo
 - Correctness can be self-checked





Lesson delivery

- 1 "lecturer" + 4-6 helpers
 - "Lecturer" goes through concepts
 - "Lecturer" introduces hands-on exercises
 - Helpers lead hands-on exercises in **breakout rooms**

• Fully remote using Zoom

- Low barrier for international audience
- Hands-on in breakout rooms: 1 helper + 6-8 participants.
- Main room recorded and uploaded to YouTube
- Use HackMD collaborative scratchpad for Q&A
 - \circ $\,$ All questions are in public record $\,$
 - Anyone can answer
 - No risk of talking over one another



Outlook, Challenges, Opportunities

- **First** SYCL training offered by a EuroCC competence centre
- Well-attended: interest in vendor-agnostic, standard-based frameworks!
- Great catalyst for European collaborations

Tricky concepts to teach effectively

- Host-device synchronization
- Potential for performance portability difficult to fit in a 2-3 day event



Call to action

More free, online training resources for self-learning are needed:

- Focus on performance analysis of SYCL applications
- Analysis of performance portability aspects
- Debugging SYCL applications



Thank you!



enccs.se