

Empowering HPC Education with SYCL: Open-Source Modules for Modern Supercomputers

Erik Pautsch
Loyola University Chicago
Chicago, IL USA
epautsch@luc.edu

Jorge Velez
University of Illinois Urbana-Champaign
Champaign, IL USA
jorgev2@illinois.edu

Raymundo Hernandez-Esparza
Argonne National Laboratory
Lemont, IL USA
rhernandezesparza@anl.gov

Álvaro Vázquez-Mayagoitia
Argonne National Laboratory
Lemont, IL USA
vama@alcf.anl.gov

Silvio Rizzi
Argonne National Laboratory
Lemont, USA
srizzi@anl.gov

George K. Thiruvathukal
Loyola University Chicago
Chicago, IL USA
gthiruvathukal@luc.edu

Abstract—This paper explores SYCL as a versatile tool for high-performance computing (HPC), providing practical guidance tailored for educators and students. SYCL’s portability across a wide range of hardware platforms positions it as a compelling alternative to CUDA, especially within modern supercomputers featuring diverse accelerators. By developing open-source tutorial modules, this work seeks to democratize HPC education, making these resources accessible in workshops and to underserved communities, including those in Latin America.

Building on the foundational work of UnoAPI [1], our project explores SYCL’s potential to enrich HPC education through three targeted modules: addressing a traditional graph problem, generating volumetric data on particle electron density, and visualizing data with the marching cubes algorithm [2], [3]. These modules showcase SYCL’s versatility across varied computational tasks, empowering learners with the skills needed to excel in heterogeneous computing environments.

For access to the repository containing the example projects and more information, visit: <https://github.com/SYCLTutorials/Intro2024>

I. INTRODUCTION

High-performance computing is evolving rapidly with the integration of heterogeneous environments that combine GPUs, FPGAs, and other specialized processors. As these systems become more complex, the need for portable and efficient code across diverse hardware platforms is increasingly critical. SYCL offers a solution by providing a vendor-neutral programming model that allows developers to write code once and run it across multiple types of accelerators, making it an attractive alternative to CUDA [4].

However, teaching parallelism in such environments presents significant challenges, particularly when traditional educational resources do not address the nuances of different programming models. Our research aims to overcome these obstacles by developing open-source educational materials that are accessible to a global audience, including underserved regions. Building on the work of unoAPI [1], we have created tutorial modules that demonstrate SYCL’s practical applications in high-performance computing, from traditional graph problems to advanced scientific visualization.

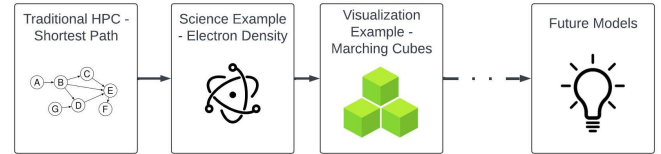


Figure 1. Flow of SYCL tutorial modules: Starting with a traditional HPC problem (Shortest Path), progressing through a scientific computation (Electron Density), and culminating in a visualization example (Marching Cubes). The modular design allows for future expansion into additional models and applications.

This paper presents three modules developed as part of this effort. The first module addresses a traditional graph problem – finding the shortest path – commonly used in HPC education. The second module demonstrates SYCL’s capabilities in scientific computing by generating volumetric data based on particle electron density. The third module focuses on porting the marching cubes algorithm from CUDA to SYCL [2], [3], illustrating how SYCL can be applied to visualization techniques. These resources are designed to empower the next generation of computer scientists with the tools and knowledge they need to succeed in an increasingly heterogeneous computational landscape.

II. METHOD

Module 1: Porting Familiar Computer Science Codes

The first module introduces a classic HPC problem – finding the shortest path in a graph. This problem is well-suited for demonstrating parallelism and serves as an accessible entry point for educators and students new to SYCL. The module begins by exploring the implementation of this problem using basic SYCL constructs. We highlight how SYCL handles parallelism in a manner similar to CUDA, yet with the added flexibility of running across different hardware platforms [4].

This module serves as a foundational example, helping students build confidence in using SYCL before moving on to the more complex scientific and visualization tasks.

Module 2: Porting a Scientific Code for Particle Electron Density

The second module focuses on generating volumetric data based on particle electron density, a task commonly encountered in scientific computing fields such as computational chemistry and physics. This module was designed to demonstrate SYCL's capabilities in handling complex, data-intensive computations.

During the implementation, we emphasized best practices for memory management and data access within SYCL, ensuring that the computational resources were managed efficiently. This module also demonstrates how SYCL can be used to solve problems typically associated with scientific research, thus broadening the appeal and application of SYCL in educational settings [4].

The output of this module – volumetric data – can naturally flow into the third module, where it serves as the input for visualization using the marching cubes algorithm. This progression demonstrates to students how different computational tasks can be integrated within a single framework.

Module 3: Porting the Marching Cubes Visualization Algorithm

The marching cubes algorithm, widely used in scientific visualizations and graphics [2], was selected as an effective teaching tool for introducing SYCL. Initially, the SYCLomatic tool was employed to automate the port from CUDA to SYCL, providing a baseline implementation [3]. However, several challenges emerged that required manual adjustments

SYCLomatic's translation of CUDA's block size and thread count into ND-Range kernels led to unnecessary complexity for educational purposes. We simplified the implementation by refactoring the code to use the 'range' model instead, making it more accessible for learners.

Additionally, SYCLomatic's use of the DPCT namespace caused compilation issues, which we resolved by switching to the standard SYCL namespace. This change improved code compatibility and aligned with best practices

Finally, SYCLomatic's handling of texture objects through unified shared memory was replaced with SYCL's buffer/accessor model. This adjustment provided a clearer demonstration of SYCL's memory management features, an essential concept for students.

III. PORTABILITY AND EDUCATIONAL IMPACT

A. Portability Across Educational Settings

One of the key goals of our project is to ensure that the educational modules are portable and adaptable across different educational settings. By leveraging SYCL's platform-agnostic nature [4], these modules can be used on a wide range of hardware, from local clusters to supercomputers, ensuring

that educators can incorporate them into their curriculum regardless of their institution's specific computing resources.

To maximize accessibility, all modules have been designed with clear documentation and step-by-step instructions, enabling educators to adapt the materials to their specific teaching needs. The open-source nature of the project further enhances its portability. We want to allow educators to customize the modules or extend them with additional content as needed.

B. Inspiring Interest Through Non-Traditional Examples

While traditional HPC examples like graph problems are important, we believe that showcasing non-traditional applications of parallelism can inspire greater interest and excitement among students. By including modules that focus on scientific computation and visualization, we demonstrate the broad applicability of SYCL beyond conventional HPC tasks.

These non-traditional examples, such as the generation of electron density data and the visualization of complex surfaces, help to contextualize the importance of parallel computing in a wide array of scientific and engineering fields. By exposing students to these diverse applications, we hope to foster a deeper understanding of the power and potential of parallel computing, encouraging them to explore its use in their own areas of interest.

ACKNOWLEDGMENT

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.ff

REFERENCES

- [1] G. K. Thiruvathukal, "unoapi: A unified parallel computing api," <https://unoapi.org/>, 2024, accessed: 2024-08-02.
- [2] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.
- [3] N. Corporation, "Marching cubes cuda implementation," <https://github.com/NVIDIA/cuda-samples>, 2024, accessed: 2024-08-02.
- [4] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, and X. Tian, "Data parallel c++: Mastering dpc++ for programming of heterogeneous systems using c++ and sycl," Apress, 2021, ISBN: 978-1484275282. [Online]. Available: <https://www.apress.com/gp/book/9781484275282>