Stewardship of Software for Scientific and High-Performance Computing RFI
The SYCL Portable Programming Model for Accelerators

Thomas Applencourt (ANL), Brandon Cook (LBNL),
Mehdi Goli (Codeplay), Kevin Harms (ANL), Nevin Liber (ANL),
Vincent Pascuzzi (BNL), Michael Wong (Codeplay)

December 13, 2021

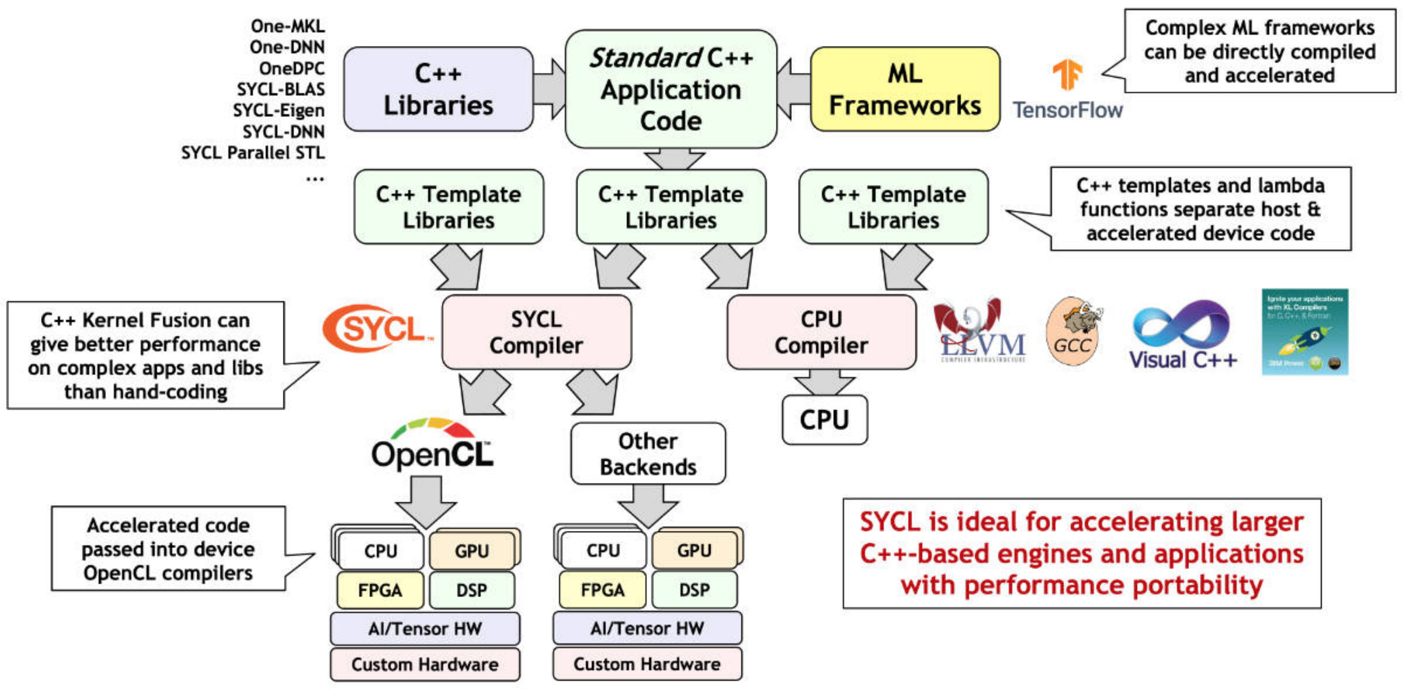
One-MKL
One-DNN
OneDPC
SYCL-BLAS
SYCL-Eigen
SYCL-DNN
SYCL Parallel STL
...
C++ Libraries
Standard C++ Application Code
ML Frameworks
TensorFlow
Complex ML frameworks can be directly compiled and accelerated
C++ Template Libraries
C++ Template Libraries
C++ Template Libraries
C++ templates and lambda functions separate host & accelerated device code
C++ Kernel Fusion can give better performance on complex apps and libs than hand-coding
SYCL
SYCL Compiler
CPU Compiler
LLVM
GCC
Visual C++
CPU
OpenCL
Other Backends
Accelerated code passed into device OpenCL compilers
CPU
GPU
FPGA
DSP
AI/Tensor HW
Custom Hardware
CPU
GPU
FPGA
DSP
AI/Tensor HW
Custom Hardware
SYCL is ideal for accelerating larger C++-based engines and applications with performance portability

Figure 1: Example workflow for SYCL
Courtesy of khronos.org

The Stewardship of Software for Scientific and High-Performance Computing requests information about important software dependencies for software packages (Q1). The information provided is based on input from the authors who are users of SYCL, implementers of SYCL and participants in the SYCL WG specification process.

SYCL

A key element for maintainable long-lived software is the use of modern languages and portable programming models. SYCL [1] fulfils both of these requirements given that it is based on modern C++ and provides portability across a range of CPUs and accelerator architectures. SYCL 2020 [2] is based on ISO C++17 (with some additional features aligned with C++20), is single-source and fully C++ compliant. The SYCL programming model will be a critical element as part of the Department of Energy Stewardship program, and is one of the primary programming models for the forth coming Aurora [3] supercomputer from the Argonne Leadership Computing Facility. The ALCF, in partnership with NERSC and OLCF, are collaborating with Codeplay to extend the open-source Data Parallel C++ (DPC++) SYCL compiler, led by Intel, to support Perlmutter (Nvidia-based), Frontier (AMD-based) and the future LANL HPC (ARM-based) which provides SYCL2020 support on all DOE Exascale Computing Project platforms. In addition to DPC++, there are numerous open-source SYCL libraries, including those developed for AI/ML and linear algebra applications.

Background

SYCL (pronounced ‘sickle’) is a royalty-free, cross-platform abstraction layer that enables code for heterogeneous processors to be written using standard ISO C++ with the host and kernel code for an application contained in the same source file. The latest specification, SYCL 2020, is based on ISO C++17 standard, and features standard programming with templates and lambda functions to develop optimized code which can be offloaded to special purpose compute accelerators such as GPUs, FPGA, or AI/ML accelerators. The SYCL specification is designed to be a higher level abstraction above low-level native acceleration APIs with interoperatbility between existing libraries and other parallel programming models and can be built on top of OpenMP, Vulkan, OpenCL, Kokkos, Raja, or some other backend. Figure 1 above describes an example code flow of how SYCL interacts with and integrates into application development.

Community

The SYCL community is well established and based around the Khronos Group. The Khronos Group is an open organization that any person or organization may join at a cost based on type of membership. DOE is currently represented in the SYCL committee of the Khronos Group by ALCF, which is now a member of Khronos.

All members may participate in the creation, editing and publication of the specification with each organization having an equal vote. The SYCL working group has a wide range of members who work on embedded processing up through supercomputing. This community participation has resulted in a large number of independent implementations that support various different hardware from multiple vendors. Figure 2 provides a graphical description of the existing ecosystem showing different implementations and which hardware they support. This selection of implementations range from fully open-source with community support up to proprietary with full commercial support.

Research and Development

As discussed above, SYLC has an active community in research and development around the use of SYCL as well as related to implementations. Examples of runtime work are Codeplay working with ALCF and NERSC to extend the Intel *joint matrix* proposal to support Nvidia tensor cores in

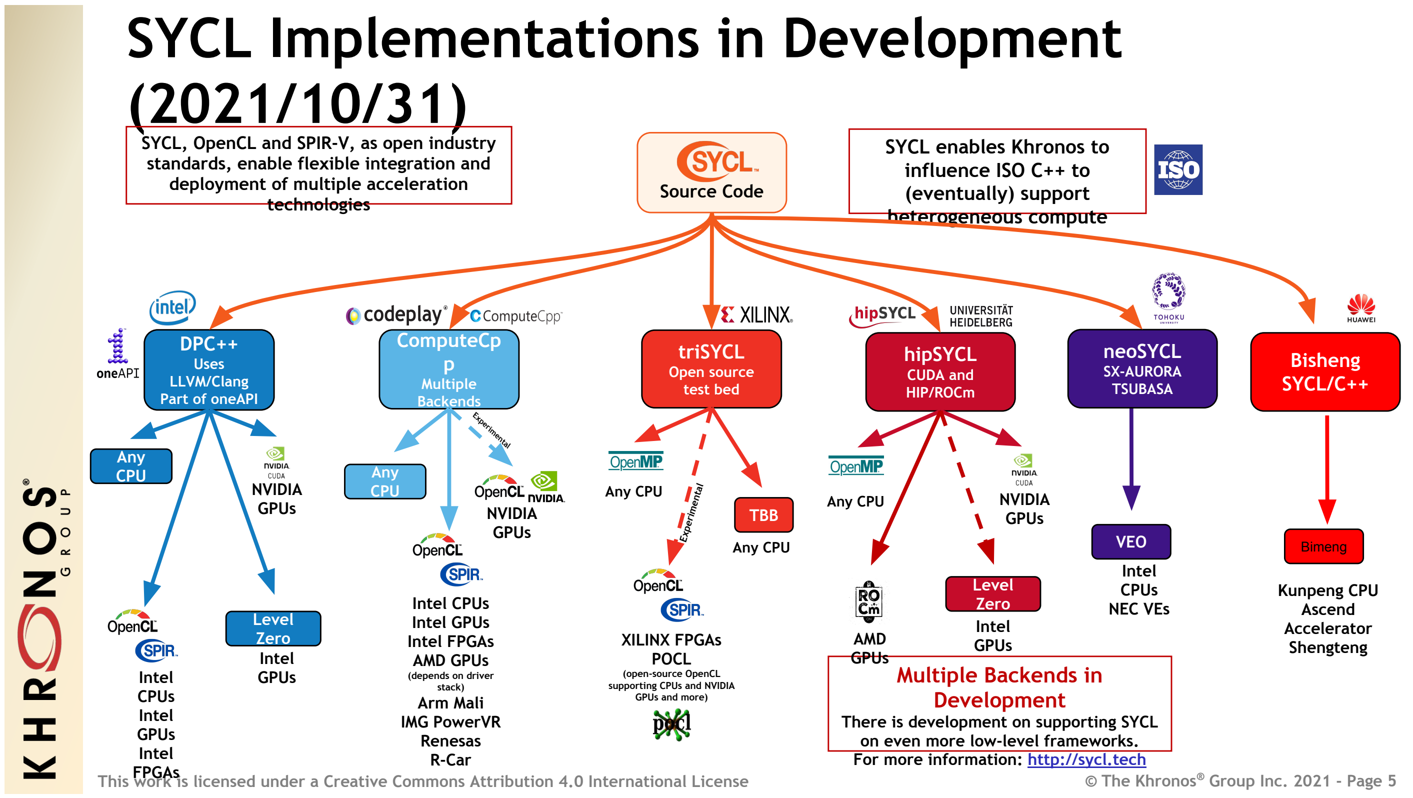
SYCL Implementations in Development (2021/10/31)
SYCL, OpenCL and SPIR-V, as open industry standards, enable flexible integration and deployment of multiple acceleration technologies
SYCL Source Code
SYCL enables Khronos to influence ISO C++ to (eventually) support heterogeneous compute
ISO
intel
oneAPI
DPC++ Uses LLVM/Clang Part of oneAPI
Any CPU
NVIDIA GPUs
OpenCL SPIR
Intel CPUs Intel GPUs Intel FPGAs
Level Zero
Intel GPUs
codeplay
ComputeCpp
ComputeCpp Multiple Backends
Experimental
Any CPU
OpenCL NVIDIA
NVIDIA GPUs
OpenCL SPIR
Intel CPUs Intel GPUs Intel FPGAs AMD GPUs (depends on driver stack) Arm Mali IMG PowerVR Renesas R-Car
XILINX
triSYCL Open source test bed
OpenMP
Any CPU
Experimental
TBB
Any CPU
OpenCL SPIR
XILINX FPGAs POCL (open-source OpenCL supporting CPUs and NVIDIA GPUs and more)
pocl
hipSYCL
UNIVERSITÄT HEIDELBERG
hipSYCL CUDA and HIP/ROCm
OpenMP
Any CPU
NVIDIA CUDA
NVIDIA GPUs
ROCm
AMD GPUs
Level Zero
Intel GPUs
TOHOKU
neoSYCL SX-AURORA TSUBASA
VEO
Intel CPUs NEC VEs
HUAWEI
Bisheng SYCL/C++
Bimeng
Kunpeng CPU Ascend Accelerator Shengteng
Multiple Backends in Development
There is development on supporting SYCL on even more low-level frameworks.
For more information: http://sycl.tech
This work is licensed under a Creative Commons Attribution 4.0 International License
© The Khronos® Group Inc. 2021 - Page 5
KHRONOS GROUP

Figure 2: SYCL Ecosystem
Courtesy of khronos.org

addition to Intel AMX instructions. The work by Ke et. al [4] to build a SYCL runtime for the SX Aurora TSUBASA processor.

Applications code development focusing on portability is demonstrated by several works including, Mills et. al [5] showing portable performance of PETSC on GPUs, Cali et. al [6] creating a portable implementation of a conjugate gradient solver for CPUs, GPUs and FPGAs, and Passcuzi et. al [7] developing a performance portable random number generator. Finally, a BLAS implementation built in SYCL to provide portability for BLAS operations by Aliaga et. al [8].

Conclusion

The SYCL programming model should be a continued focus for DOE, as it offers performance portability across various vendor hardware and interoperability with both open-source and closed-source (proprietary) software. As SYCL evolves, the DOE should continue to play a key part to drive HPC-critical features into the specification while preventing unwanted behaviors from leaking into the specification. Effective stewardship of portable programming models will be a continued investment in the SYCL specification through standards body participation, support of open-source SYCL implementations and further research and development of new capabilities as hardware advances.

Acknowledgements

This work used resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility operated under contract DE-AC02-06CH11357. Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

[1] "SYCL website," http://sycl.tech/.

[2] "SYCL 2020 Speciication," https://www.khronos.org/registry/SYCL/specs/sycl-2020/html/sycl-2020.html.

[3] "Aurora Supercomputer," https://alcf.anl.gov/aurora.

[4] Y. Ke, M. Agung, and H. Takizawa, "Neosycl: A sycl implementation for sx-aurora tsubasa," in *The International Conference on High Performance Computing in Asia-Pacific Region*, ser. HPC Asia 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 50–57. [Online]. Available: https://doi.org/10.1145/3432261.3432268

[5] R. T. Mills, M. F. Adams, S. Balay, J. Brown, A. Dener, M. Knepley, S. E. Kruger, H. Morgan, T. Munson, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and J. Zhang, "Toward performance-portable petsc for gpu-based exascale systems," 2021.

[6] S. Cali, W. Detmold, G. Korcyl, P. Korcyl, and P. Shanahan, "Implementation of the conjugate gradient algorithm for heterogeneous systems," 2021.

[7] V. R. Pascuzzi and M. Goli, "Achieving near native runtime performance and cross-platform performance portability for random number generation through SYCL interoperability," *CoRR*, vol. abs/2109.01329, 2021. [Online]. Available: https://arxiv.org/abs/2109.01329

[8] J. I. Aliaga, R. Reyes, and M. Goli, "Sycl-blas: Leveraging expression trees for linear algebra," in *Proceedings of the 5th International Workshop on OpenCL*, ser. IWOCL 2017. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3078155.3078189