13th International Workshop on OpenCL and SYCL

IWOCL 2025

SYCL State of the Union

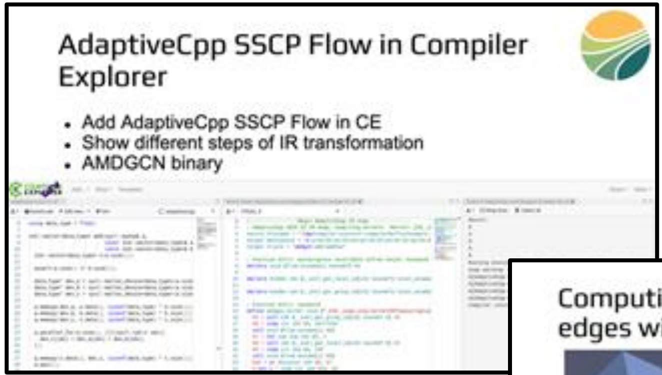
Tom Deakin, University of Bristol

SYCL Working Group Chair

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

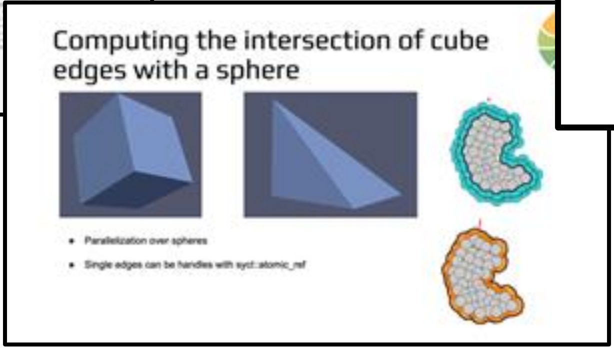
SYCL Hackathon at IWOCL'25

- **Hackathon Goals:**
  - Support projects using SYCL with access to mentors.
  - Share top tips and latest topics through presentations.
- **4 projects being worked on, with great results.**



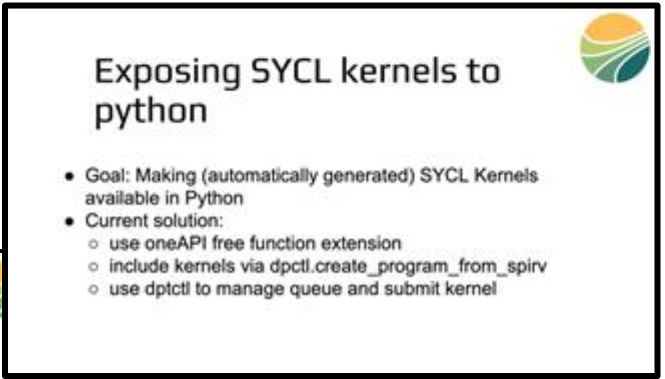
AdaptiveCpp SSCP Flow in Compiler Explorer

- Add AdaptiveCpp SSCP Flow in CE
- Show different steps of IR transformation
- AMDGCN binary



Computing the intersection of cube edges with a sphere

- Parallelization over spheres
- Single edges can be handles with sycl::atomic_ref



Exposing SYCL kernels to python

- Goal: Making (automatically generated) SYCL Kernels available in Python
- Current solution:
  - use oneAPI free function extension
  - include kernels via dpctl.create_program_from_spirv
  - use dptctl to manage queue and submit kernel

SYCL Working Group Priorities through 2025

Deliver standardised and implemented improvements and features as KHR extensions: e.g.,

New kernel submission APIs to reduce submission latency

Default queue construction overhead reductions through default context

Support for complex numbers, and plans for other reduced-precision data types

Default context query [ratified]

Other extensions from contributors, previously seen at IWOCL, etc

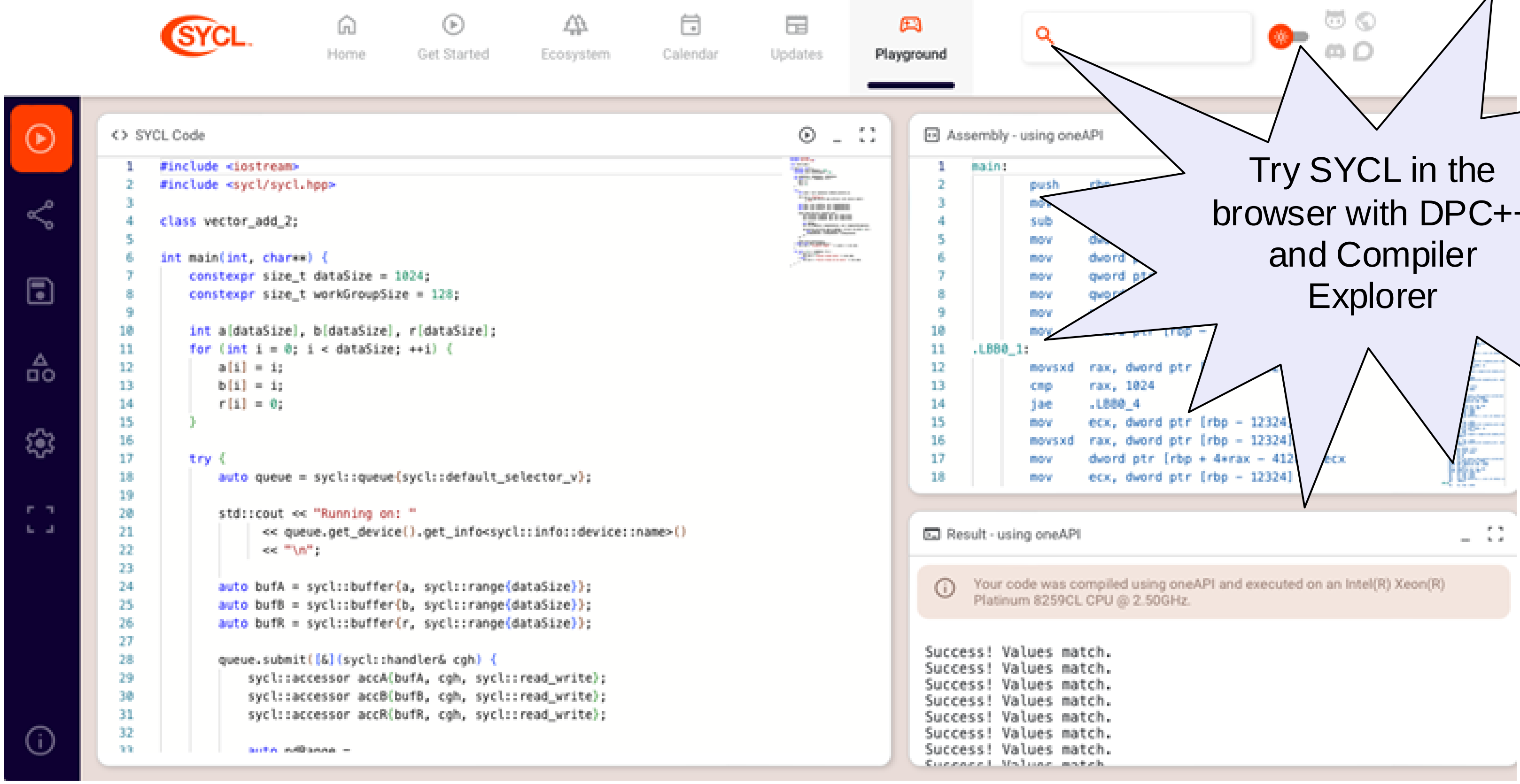
Follow progress and give feedback on Khronos Group's GitHub where the draft spec is developed in public.

Find out more on the Khronos Blog: https://khr.io/17d

SYCL 2020 rev10 released

Available now at https://registry.khronos.org/SYCL/

- **Appendix F: Optional Extensions**
  - The home of KHRs.
  - sycl_khr_default_context
- **Clarifications and fixes to sycl::vec**
- **Clarification on sycl::free**
  - May or may not be blocking; not the same as submitting a free to a queue.
- **Simplified introduction text (“What is SYCL”)**
- **Clarifications on device types**
- **Clarification on default context behavior**
  - Improve performance for applications that construct many (many) queues with default constructor.
- **Clarification of SYCL_LANGUAGE_VERSION macro value**
- **Clarified that Hierarchical Parallelism is deprecated in SYCL 2020**
- **Clarify that the C++ restrictions for device code do not apply to constant expressions**
  - E.g. a constant expression can use “long double” even though this type can not generally be used inside a kernel



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

class vector_add_2;

int main(int, char**) {
    constexpr size_t dataSize = 1024;
    constexpr size_t workGroupSize = 128;

    int a[dataSize], b[dataSize], r[dataSize];
    for (int i = 0; i < dataSize; ++i) {
        a[i] = i;
        b[i] = i;
        r[i] = 0;
    }

    try {
        auto queue = sycl::queue{sycl::default_selector_v};

        std::cout << "Running on: "
                  << queue.get_device().get_info<sycl::info::device::name>()
                  << "\n";

        auto bufA = sycl::buffer{a, sycl::range{dataSize}};
        auto bufB = sycl::buffer{b, sycl::range{dataSize}};
        auto bufR = sycl::buffer{r, sycl::range{dataSize}};

        queue.submit([&](sycl::handler& cgh) {
            sycl::accessor accA{bufA, cgh, sycl::read_write};
            sycl::accessor accB{bufB, cgh, sycl::read_write};
            sycl::accessor accR{bufR, cgh, sycl::read_write};

            auto ndRange =
```

```
main:
        push    rbp
        mov
        sub
        mov     dw
        mov     dword
        mov     qword pt
        mov     qword
        mov
        mov          ptr [rbp -
.LBB0_1:
        movsxd  rax, dword ptr [
        cmp     rax, 1024
        jae     .LBB0_4
        mov     ecx, dword ptr [rbp - 12324]
        movsxd  rax, dword ptr [rbp - 12324]
        mov     dword ptr [rbp + 4*rax - 412    ecx
        mov     ecx, dword ptr [rbp - 12324]
```

Your code was compiled using oneAPI and executed on an Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz.

```
Success! Values match.
Success! Values match.
Success! Values match.
Success! Values match.
Success! Values match.
Success! Values match.
Success! Values match.
Success! Values match.
```

Try SYCL in the browser with DPC++ and Compiler Explorer

https://sycl.tech

SYCL Reference

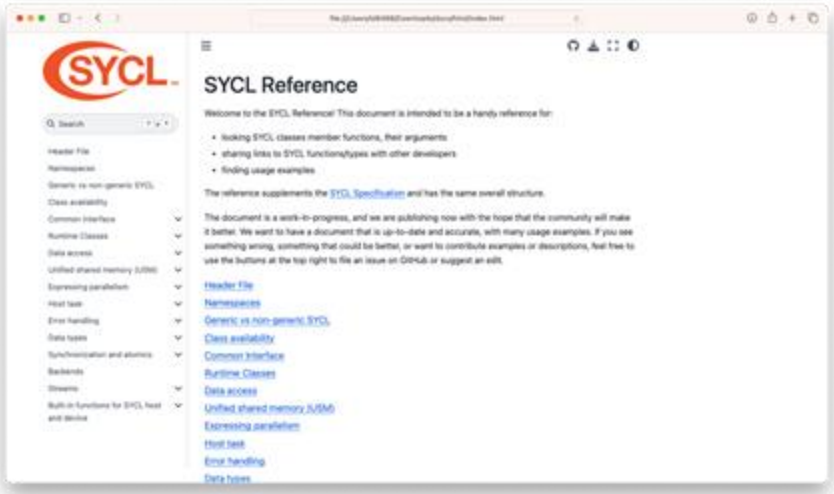
Resource to support SYCL developers

Inspired by cppreference.com

Short descriptions of SYCL 2020 API

Specification remains the canonical document

https://www.khronos.org/sycl/reference



SYCL Reference

Welcome to the SYCL Reference! This document is intended to be a handy reference for:

- looking SYCL classes member functions, their arguments
- sharing links to SYCL functions/types with other developers
- finding usage examples

The reference supplements the SYCL Specification and has the same overall structure.

The document is a work-in-progress, and we are publishing now with the hope that the community will make it better. We want to have a document that is up-to-date and accurate, with many usage examples. If you see something wrong, something that could be better, or want to contribute examples or descriptions, feel free to use the buttons at the top right to file an issue on GitHub or suggest an edit.

Header File
Namespaces
Generic vs non-generic SYCL
Class availability
Common Interface
Runtime Classes
Data access
Unified shared memory (USM)
Expressing parallelism
Host task
Error handling
Data types

SYCL Implementations

- **Intel oneAPI DPC++: SYCL 2020 conformance for Intel CPUs and GPUs**
  - Also supports any CPU in LLVM via native CPU
  - Also supports NVIDIA GPUs and AMD GPUs with Codeplay oneAPI Plugins
- **AdaptiveCpp:** Community-driven supporting LLVM-supported CPUs, and Intel, AMD and NVIDIA GPUs
- **C-DAC ParaS compiler:** x86, IBM, Arm CPUs, and NVIDIA GPUs
- **SimSYCL:** single-threaded, library-only implementation for debugging and verification
- **triSYCL:** Research implementation
- **Celerity:** SYCL abstractions for distributed systems
- **neoSYCL:** Supports CPU and SX-Aurora TSUBASA

Unified Acceleration Foundation

oneAPI Specification and Projects

- SYCL powers the UXL Foundation libraries

Heterogeneous, cross-vendor programming model

Specification

oneDPL
Data
Parallel C++ Library

oneDNN
Deep Neural
Network Library

oneCCL
Collective
Communications Library

oneDAL
Data
Analytics Library

oneTBB
Threading
Building Blocks

oneMath
Math
Kernel Library

Build From Open Source

Open Source Projects

oneDPL
Data
Parallel C++ Library

oneDNN
Deep Neural
Network Library

oneCCL
Collective
Communications Library

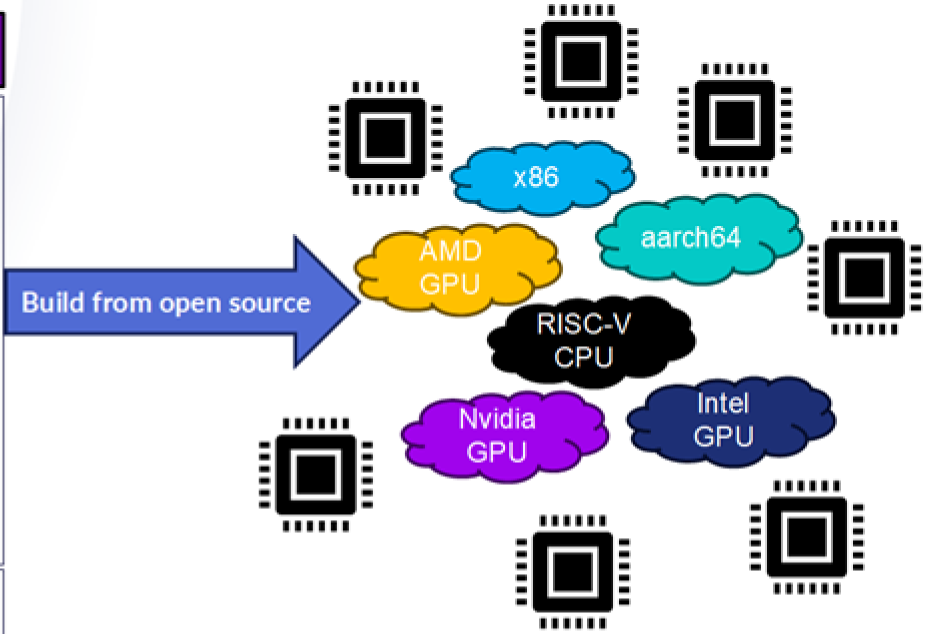
oneDAL
Data
Analytics Library

oneTBB
Threading
Building Blocks

oneMath
Math
Kernel Library

+

SYCL DPC++ LLVM Compiler Project

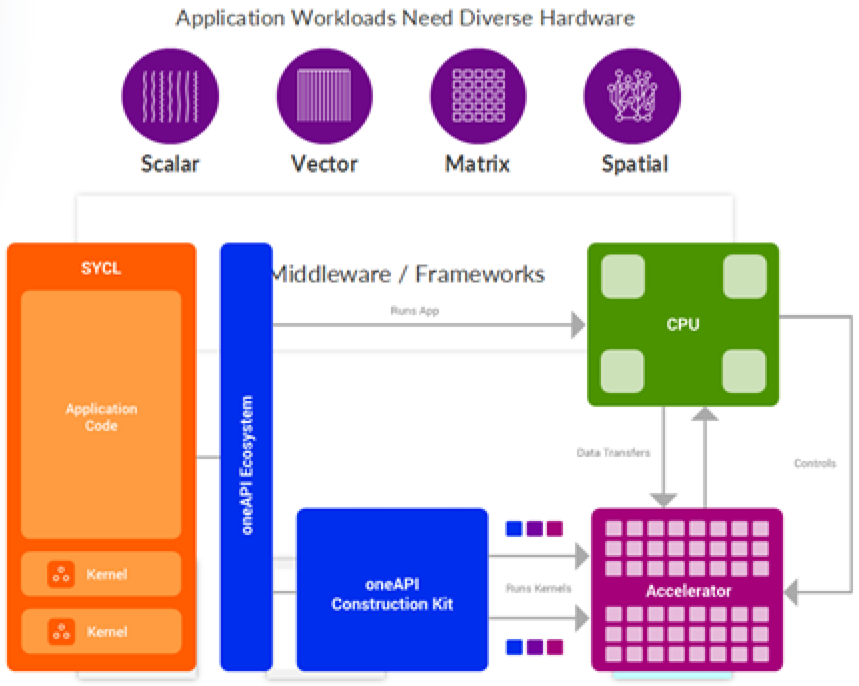


The oneAPI Construction Kit

Has been contributed to the UXL Foundation

The oneAPI Construction Kit brings SYCL and oneAPI to new RISC-V accelerators.

The oneAPI Construction Kit works by enabling the CPU to offload compute-intensive kernels to the custom accelerator



UXL
Unified Acceleration Foundation

SYCL Developer Resources

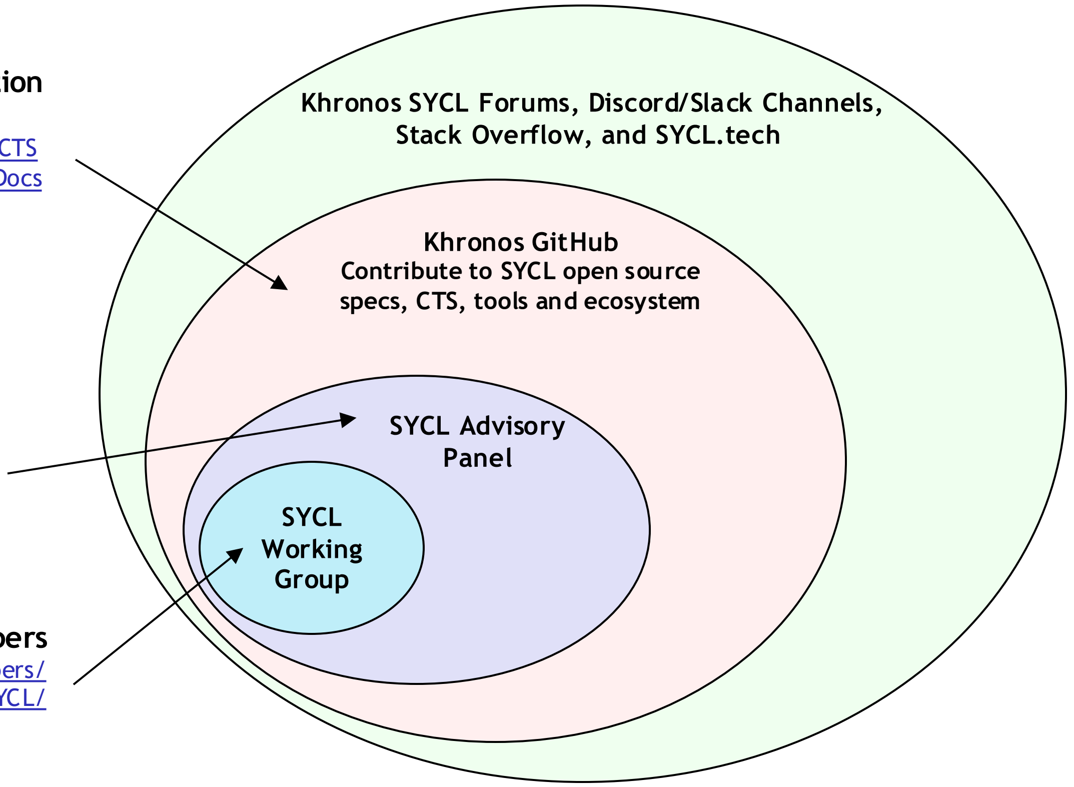
- **I need to learn SYCL**
  - The book
  - Attend a tutorial
  - SYCL Academy: https://github.com/codeplaysoftware/syclacademy
- **I know SYCL, and need more information about an API**
  - SYCL Reference https://www.khronos.org/sycl/reference
- **I need to know the ins-and-outs of an API**
  - SYCL Spec (it's quite readable!) https://registry.khronos.org/SYCL/
- **I still need help!**
  - Forums:
    - https://community.khronos.org/c/sycl/
    - https://stackoverflow.com/questions/tagged/sycl
  - SYCL.tech: https://sycl.tech/
  - Khronos Discord: https://www.khr.io/khrdiscord
  - Ask your implementor

Get involved!

Public contributions to Specification and Conformance Tests
https://github.com/KhronosGroup/SYCL-CTS
https://github.com/KhronosGroup/SYCL-Docs

Join as an Invited Expert (no cost, sign Khronos NDA)
https://www.khronos.org/advisors/

Join as a Khronos members
https://www.khronos.org/members/
https://www.khronos.org/registry/SYCL/


Khronos SYCL Forums, Discord/Slack Channels, Stack Overflow, and SYCL.tech
Khronos GitHub
Contribute to SYCL open source specs, CTS, tools and ecosystem
SYCL Advisory Panel
SYCL Working Group