SYCL State of the Union Keynote SYCLcon 2023

Build a thriving community

Michael Wong
SYCL WG Chair
Codeplay Distinguished Engineer
ISOCPP Foundation Director
ISO C++ Directions Group
michael@codeplay.com | wongmichael.com/about

Agenda

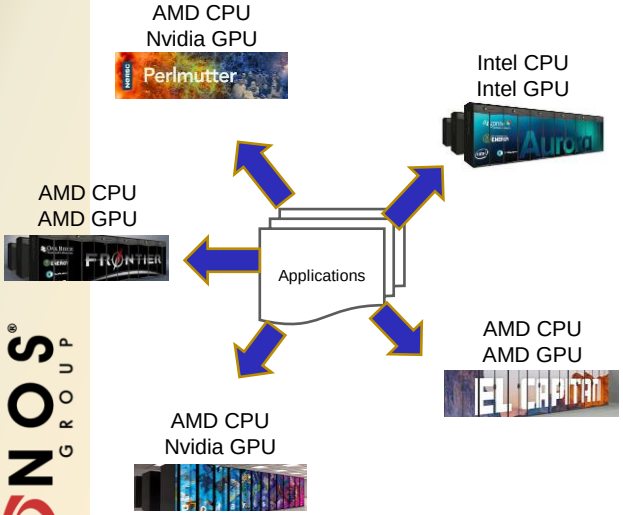
Amazing Growth

Highlights of last 12 months

Ecosystem and future growth Directions

Programming Models Must Persist

US National Laboratory Supercomputers 2021-2023



Programming models should have high quality, portable implementations

Which programming languages have formal conformance test?

ECMAscript, HTTP, Kubernetes,

Khronos languages

SYCL 2020 Conformance Test Suite released

Expressiveness and simplicity for heterogeneous programming in modern C++

New Features

Unified Shared Memory | Parallel Reductions | Subgroup Operations | Class template Argument Deduction

https://github.com/KhronosGroup/SYCL-CTS

Significant SYCL adoption in Embedded, Desktop and HPC Markets

SYCL Conformance Test Suite

SYCL2020 Features Work Plan
4.15.3 Atomic Refs
4.14.3 Marrays
4.17.3 Group Functions + 4.17.4 Group algorithms
4.6.5.2 Queue Shortcut Functions
4.7.2 Buffer Class - SYCL2020
4.6.3 Context Class + 4.6.4 Device Class - SYCL 2020
4.3 Header file - SYCL2020
4.6.1.1 Device Selector - SYCL2020
4.17.2 Function Objects - SYCL2020
4.17.5 Math Functions
4.17.6 Integer Functions
4.17.7 Common Functions
4.17.8 Geometric Functions
4.17.9 Relational Functions
4.15.1 Barriers and Fences
4.16.1 Stream Class Interface - SYCL2020
4.7 Accessor - SYCL2020
4.13 Error handling
4.11.12 Kernel Bundle - SYCL2020
4.11.13.2 Kernel Information Descriptor - SYCL2020
4.11.14 device_image class - SYCL2020
4.11.13 Kernel class - SYCL2020
4.9.1.7 Group Class
4.9.1.8 Sub-group Class
4.5.2 Common Reference Semantics - SYCL2020
4.5.4 is_property_xxx - SYCL2020
4.6.5 (partly) Queue Class Constructor - SYCL2020
4.11.8 Querying if kernel bundle exists - SYCL2020
4.9 Expressing Parallelism - SYCL2020 (not group, sub-group)
B.1./B.2. Full/Reduced feature set

SYCL 2020 Adopters Program

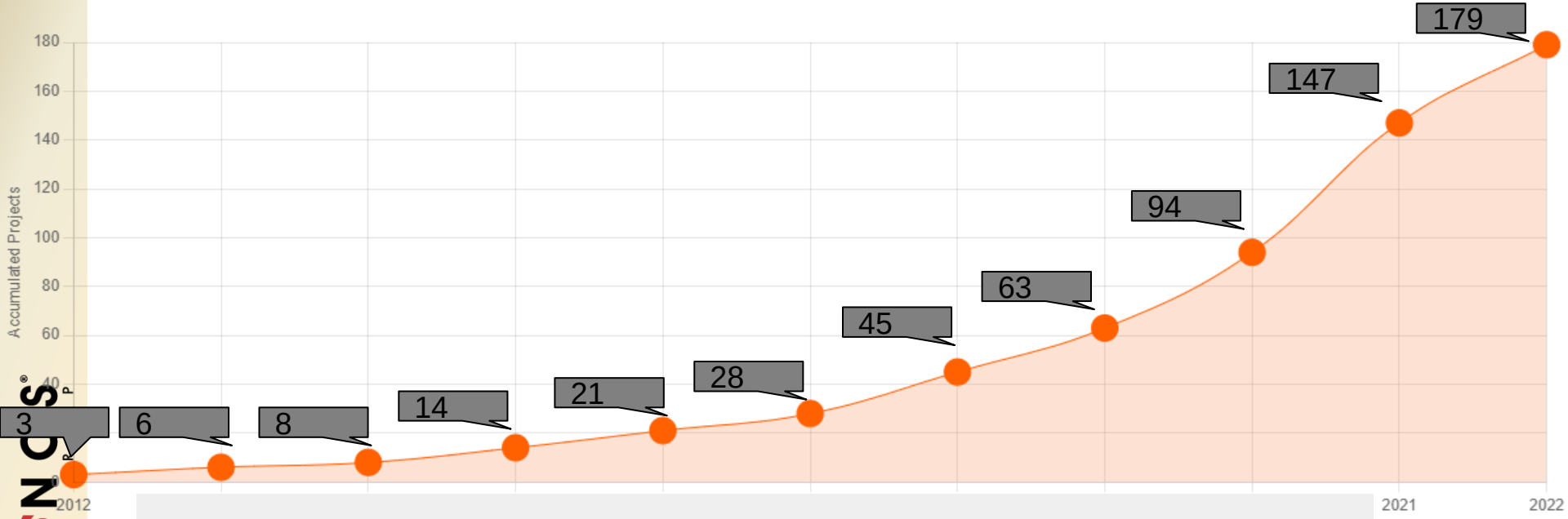
Becoming an Adopter of a Khronos standard gives you access to the Khronos Conformance Testing Process:

- Download the source of the Khronos conformance tests to port and run on your implementation.
- Access the Adopters Mailing list; a priority channel for two-way interaction with Khronos Members who can offer assistance on running tests.
- Upload generated test results for Working Group review and approval to become officially conformant.
- Submit an unlimited number of products for that version of the standard (and earlier versions as indicated in the pricing table below).

	Adopter	Implementer
Develop Products		
Access to the public Khronos Specifications, documentation and support files	✔	✔
Develop license-free, royalty-free products using Khronos Technologies	✔	✔
Conformance Testing		
Access to Adopter mailing list	✔	✖
Formal Review Process	✔	✖
Submit products to the conformance process (Must sign Adopter agreement and pay Adopter fee)	✔	✖
Conformant Product can use API Trademark (Must pass conformance tests)	✔	✖
Marketing		
Opportunity to mention products and be quoted in Khronos press releases, articles, and newsletters	✔	✖
Company logo and description on Khronos web site	✔	✖

SYCL Projects cumulative growth

Projects - SYCL.tech



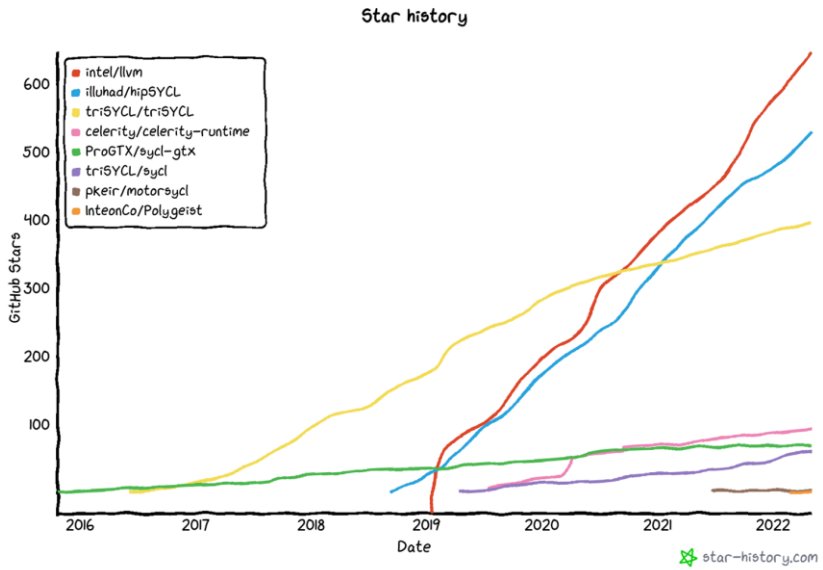
Benchmarks

Frameworks

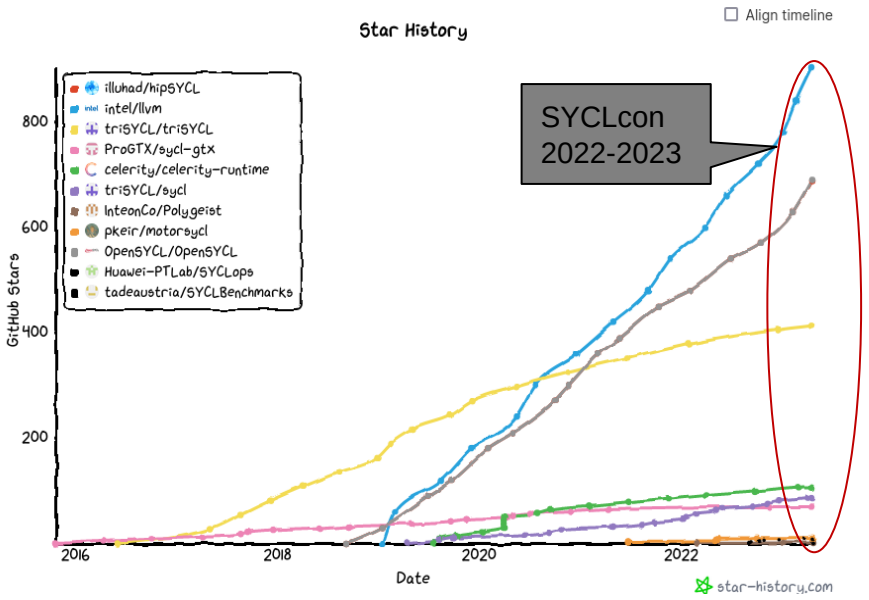
Libraries

Scientific

SYCL user and developer Phenomenal Growth



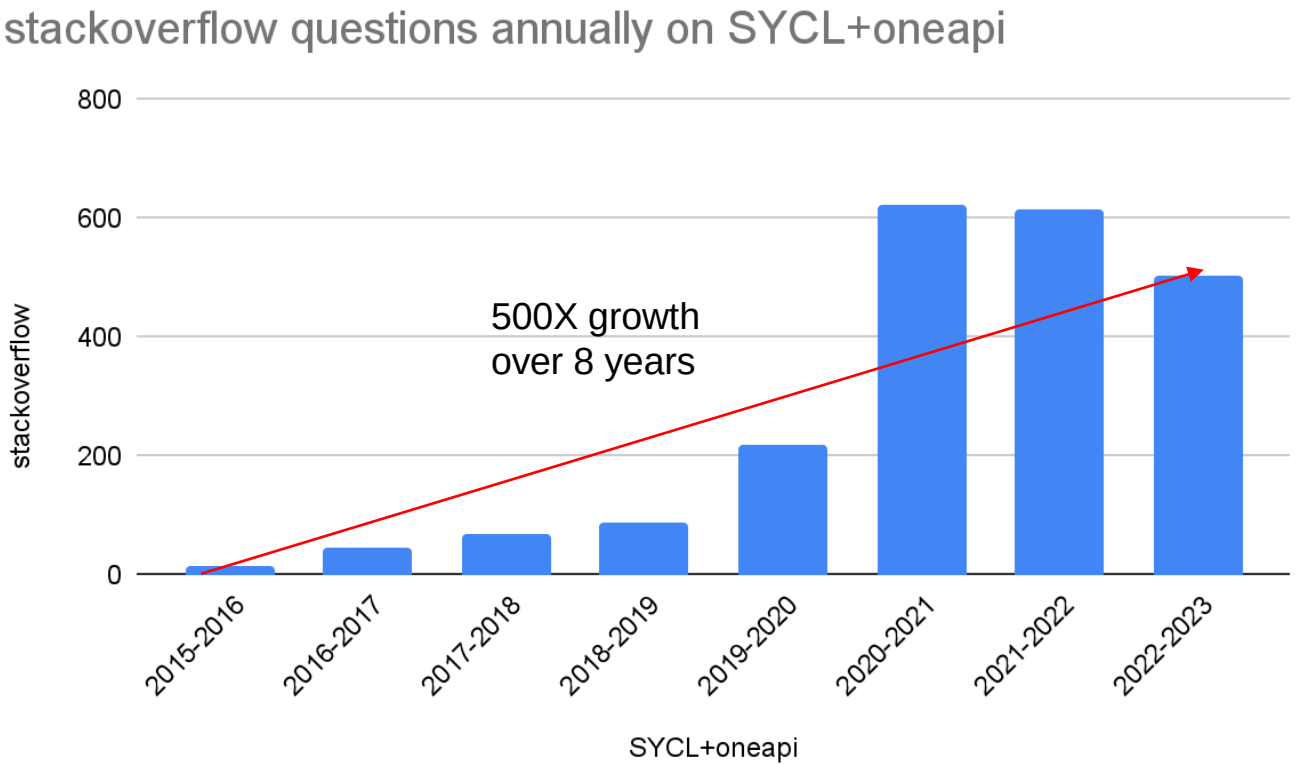
SYCLCon 2022



SYCLCon 2023

A few open-source SYCL implementations/prototypes on GitHub

500x growth over 8 years



SYCL is mainstream

Open Standards and Open Source implementations, community driven

Open cross-company collaboration

Co-design for all forms of extreme heterogeneity

Open Source without a community is useless

Companies can play in the Khronos ecosystem w/o revealing IP

Focus on ease of portability support, capable of many backends, and demonstrated to support many platforms

Agenda

Amazing Growth

Highlights of last 12 months

Ecosystem and future growth Directions

SYCL 2020 V7 highlights (Editor's corner)

Ronan Keryell, AMD

Usual 6-month maintenance release, many clarifications, no new feature.

- clarify buffer creation with nullptr;
- align more "concurrent" wording with ISO C++;
- precise that work-items provide weakly parallel forward progress guarantee;
- import forward progress definition from ISO C++ and clarify various aspects on atomicity and synchronization;
- C++17 replaced by just the C++ core language;
- fix description of max_work_item_sizes and clarify relationship to kernel dimensionality;
- clarify "group" meaning in algorithm descriptions;
- improve readability of group barrier description;
- mention kernel_handler in kernel function definition;
- relax requirement on backend traits being available;
- clarify the "reducer" member types and constants;
- clarify native_specialization_constant when empty;
- allow "empty" shared_ptr for buffer construction;
- add static constexpr `dimensions` member to all range/id-like types;
- clarify blocking behavior of `queue::submit;
- clarifications to device copyable;
- clarify USM allocation of zero size coherent with std::malloc;
- clarify sycl::atomic_ref;
- clarify queue profiling behavior when unsupported;
- clarify the wording for the use of property::queue::in_order;
- reword guarantee about host-to-device fence synchronization;
- add single source single compiler pass (SSCP) to the glossary;
- add half to sycl::plus and sycl::multiplies and fix trait use;
- clarify any_device_has / all_devices_have;
- clarify that objects in global, local, or private address space can also be accessed via the generic address space;
- disallow ++ and -- for sycl::vec<bool>;
- no assignment for read-only accessors;
- clarifications to sub-group;
- clarify is_group and bool_constant alias relations;
- clarify out-of-bounds behavior for group_broadcast

Plus many various changes...

SYCL Work in Progress

Open Standard for Single Source C++ Parallel Heterogeneous Programming

Processor-In-Memory extensions (SYCL-Extension-Document/proposed at master · SAITPublic/SYCL-Extension-Document (github.com)**)**

Integrate mdspan (OpenSYCL/extensions.md at feature/pointer-future · OpenSYCL/OpenSYCL (github.com)**)**

Generalized dimensions (triSYCL/generalized_dimension.cpp at master · triSYCL/triSYCL (github.com)**)**

oneAPI numba-dppy(Heterogeneous Programming Using Data Parallel Extension for Numba*... (intel.com)**)**

User-driven kernel fusion([SYCL][Doc] Add kernel fusion extension proposal by victor-eds · Pull Request #7098 · intel/llvm (github.com)**)**

Clarify address spacing rules(Improved address space inference for SYCL programs - YouTube**)**

SYCL design philosophy (SYCL Design Philosophy v0.1 by ProGTX · Pull Request #136 · codeplaysoftware/standards-proposals (github.com)**)**

USM-buffer interop(ComputeCpp Extensions - Guides - ComputeCpp™ Community Edition - Products - Codeplay Developer
https://github.com/OpenSYCL/OpenSYCL/blob/develop/doc/buffer-usm-interop.md
https://github.com/KhronosGroup/SYCL-Docs/issues/391))

Host task interop([SYCL][DOC] Improved host task synchronization extension by npmiller · Pull Request #7076 · intel/llvm (github.com)**,**
https://github.com/OpenSYCL/OpenSYCL/blob/develop/doc/enqueue-custom-operation.md)

Memory model, forward Progress, and Context

Ecosystem: A few SYCL Projects

Khronos Group
Request for Proposals

SYCL 2020 CTS

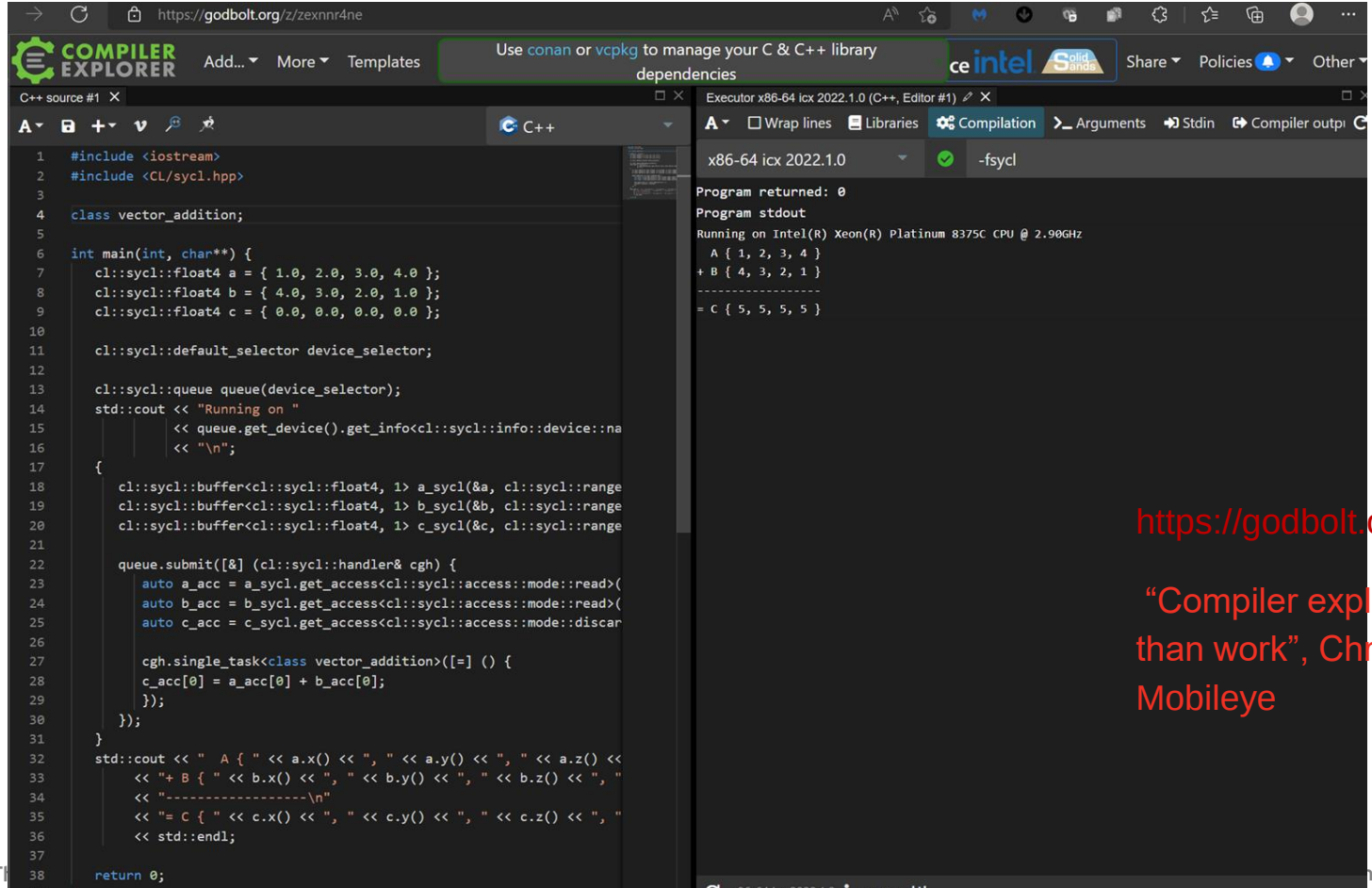
June 2022

https://github.com/KhronosGroup/SYCL-CTS

Khronos Group
Request for Proposals

SYCL 2020 Reference Guide

SYCL on Compiler Explorer (CPU execution)

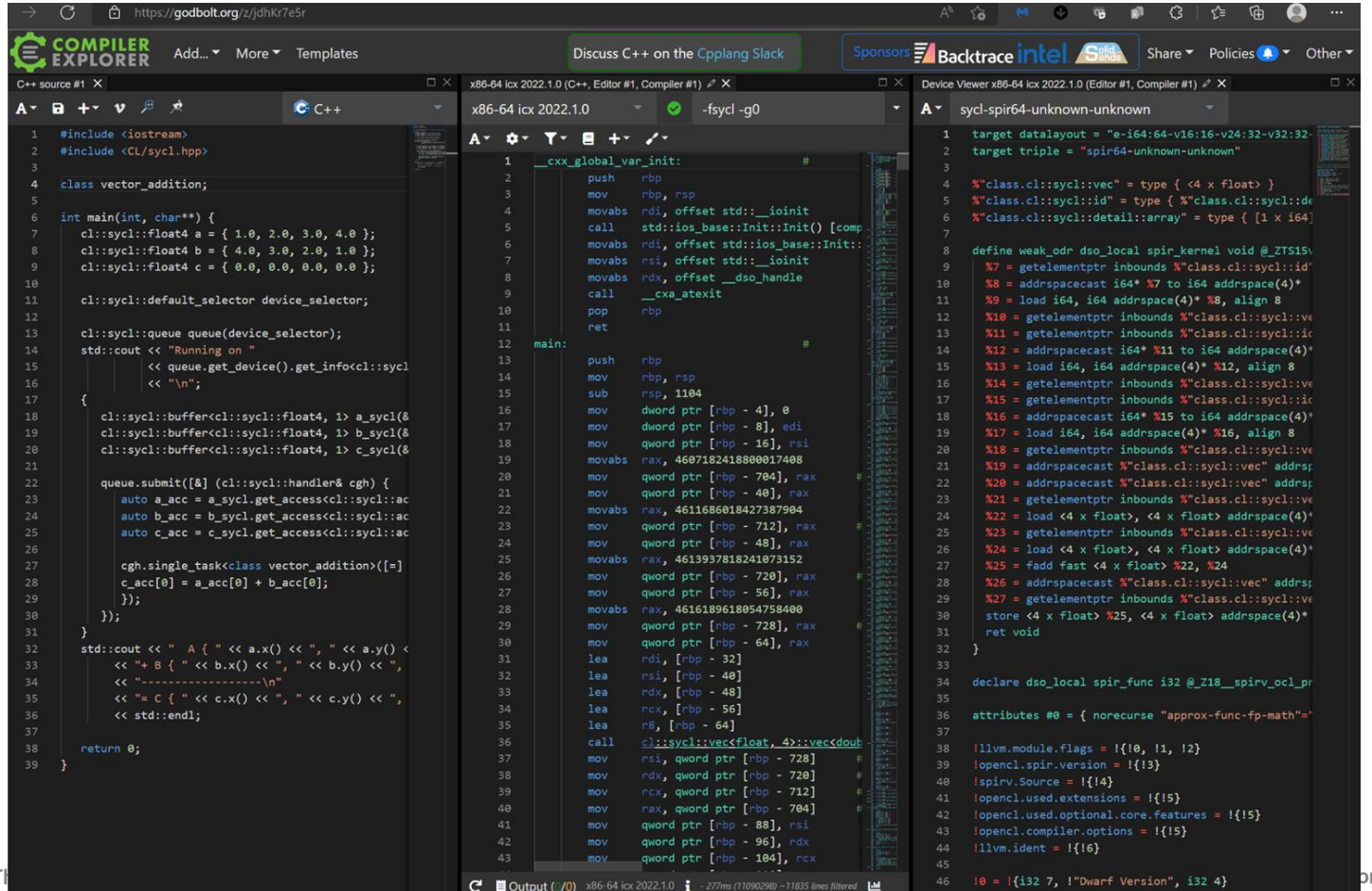


https://godbolt.org/z/zexnnr4ne

"Compiler explorer is more fun than work", Chris Gearing, Mobileye

SYCL IR on Compiler Explorer

https://godbolt.org/z/jdhKr7e5r

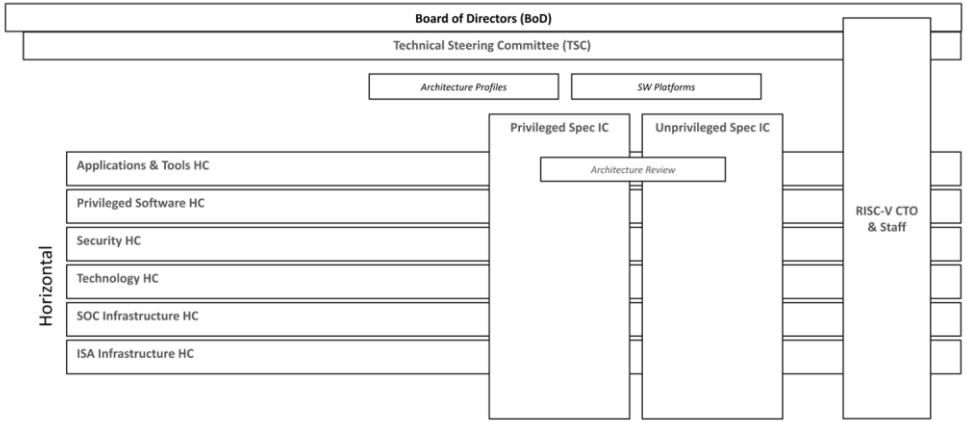


RISC-V

https://riscv.org/

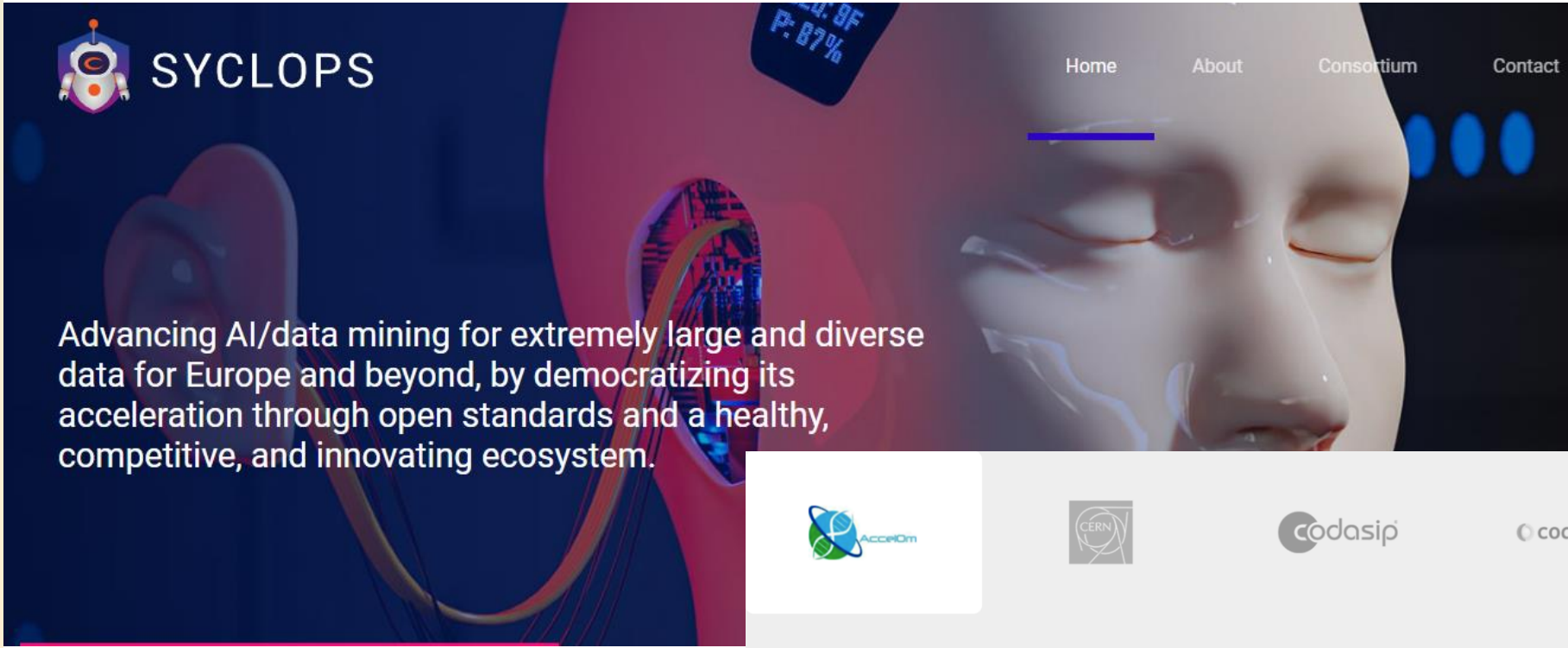
Alan Chia - Lego Color Bricks CC BY-SA 2.0

Technical Organization



SYCLOPS: SYCL RISC-V Datacenter Horizon Projects

https://www.syclops.org/



AERO: SYCL RISC-V Cloud Computing Horizon Project

Parallel Industry Initiatives

C++11

C++14

C++17

C++20

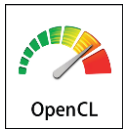
C++23

SYCL 1.2
C++11 Single source programming

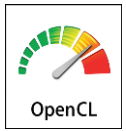
SYCL 1.2.1
C++11 Single source programming

SYCL 2020
C++17 Single source programming
Many backend options

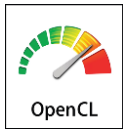
SYCL 202X
C++20 Single source programming
Many backend options



OpenCL 1.2
OpenCL C Kernel Language



OpenCL 2.1
SPIR-V in Core

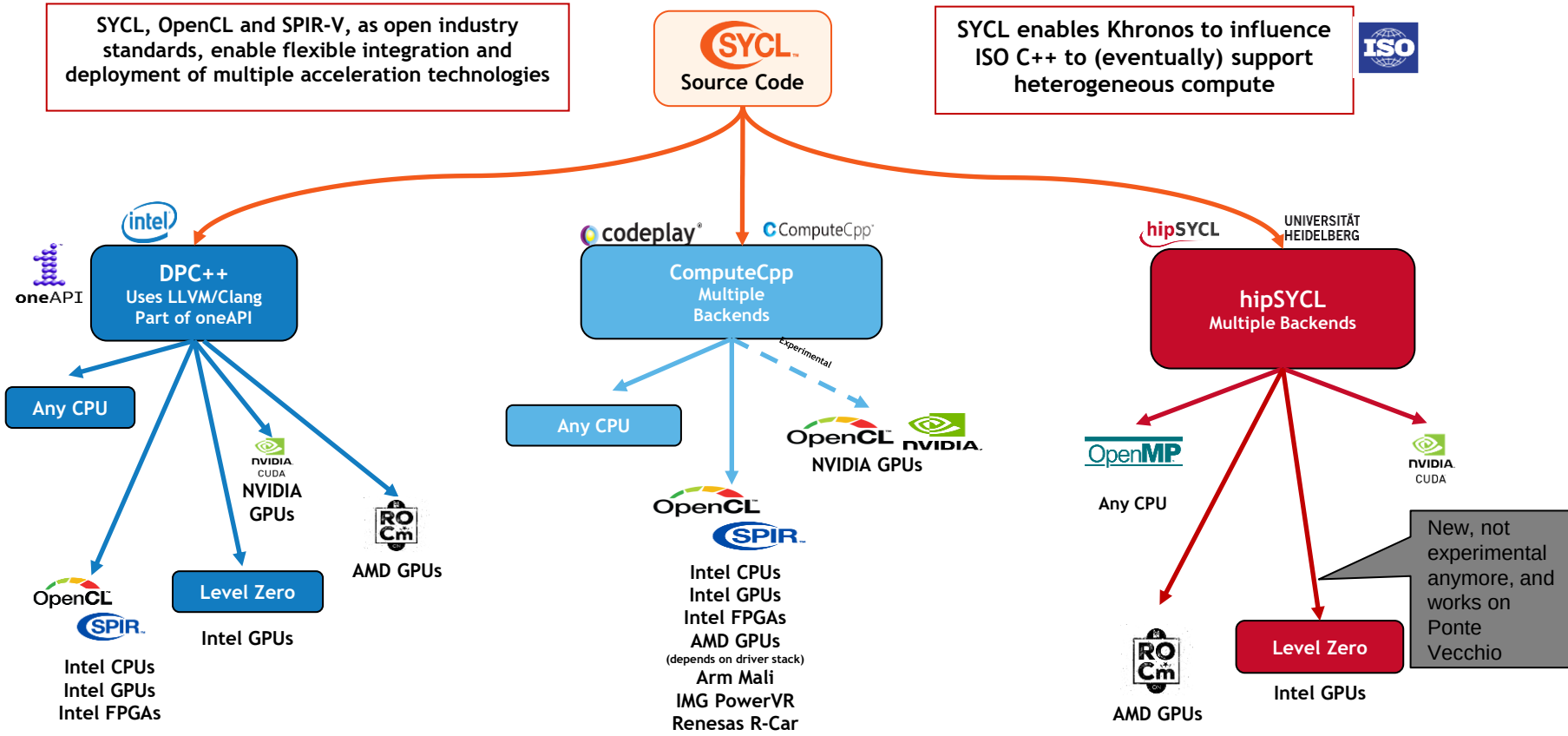


OpenCL 2.2

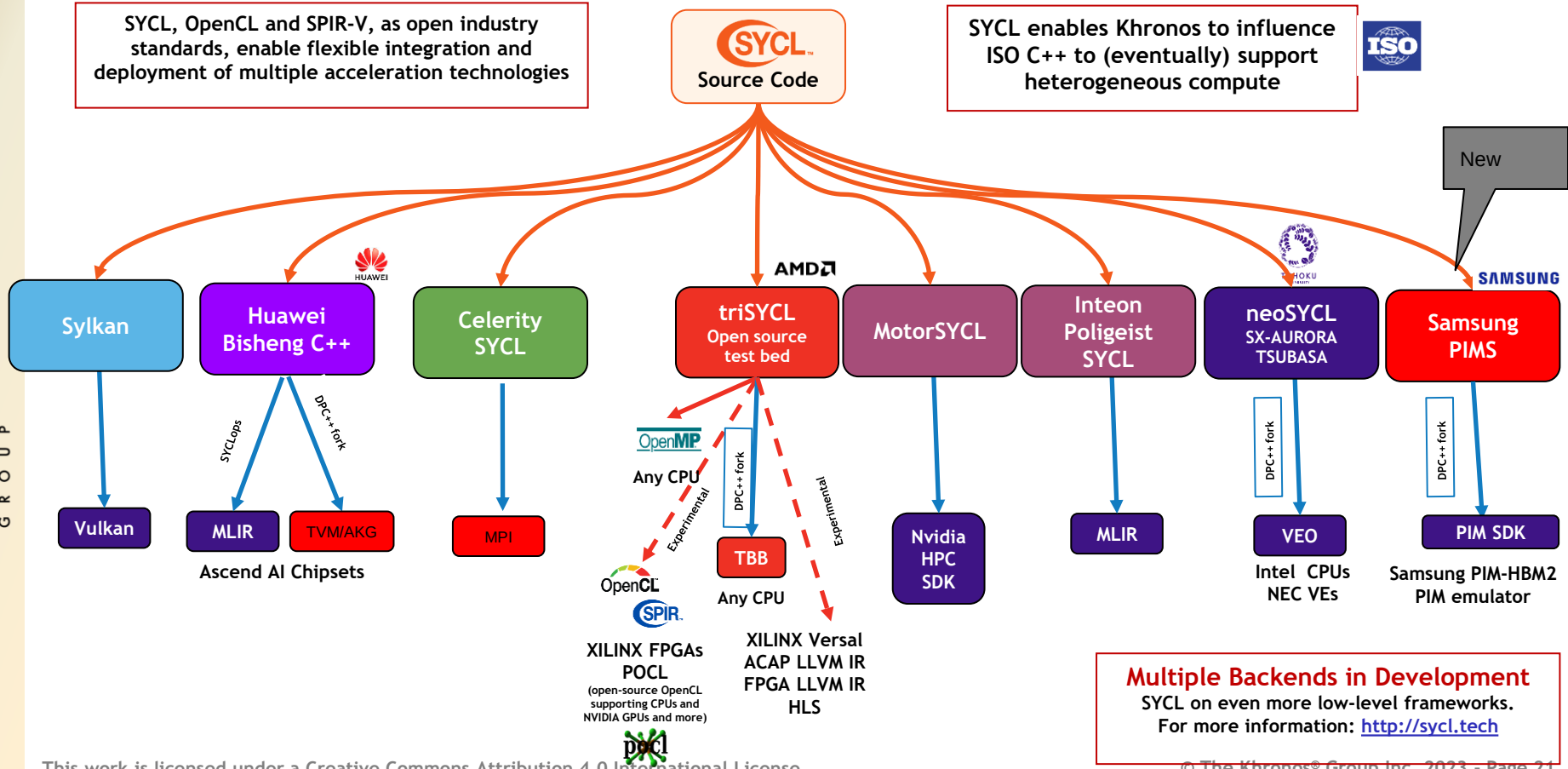
OpenCL 3.0

2011 2015 2017 2020 202X

SYCL Implementations in Development (2023/04/18)



SYCL Experimental Development (2023/04/18)



Building Performance-Portable Software

Starting from scratch

SYCL is the best place to start: open, future-proof, no lock-in, easy to learn

Starting from C++

Easy to add SYCL to existing C++ software

Starting from CUDA

Easy to port from CUDA to SYCL: keep performance on GPUs

Starting from another language

SPIR-V standard enables not just SYCL, but also new languages

Agenda

Amazing Growth

Highlights of last 12 months

Ecosystem and future growth Directions

SYCL News, Ecosystem, Research 2023/04/18

SYCL™ Performance for Nvidia® and AMD GPUs Matches Native System Language

06 April 2023

Ruyman I
CTO

Benchmarks executing workloads using DPC++, oneAPI's implementation of SYCL achieves close to native performance on Nvidia and AMD GPUs, when comparing to the same benchmarks run with CUDA®* and HIP*, respectively.

Bringing Nvidia® and AMD support to oneAPI

16 December 2022

Developers can write SYCL™ code and use oneAPI to target Nvidia* and AMD* GPUs with free binary plugins

GROMACS 2023 Released With Better SYCL For Intel / AMD / NVIDIA

Feb 23, 2023

FAST. FLEXIBLE. FREE. GROMACS

GROMACS as the widely-used molecular dynamics software issued its stable v2023 release this week with improved GPU support via SYCL. Most significant to the GROMACS 2023 feature release is improving its SYCL implementation that provides production-rated support not only for Intel Arc Graphics but also AMD Radeon graphics with ROCm + hipSYCL. There is also non-production-rated NVIDIA SYCL support as an alternative to GROMACS' CUDA support.

STFC to Accelerate Exascale Software in Computational Fluid Dynamics and Code Coupling using SYCL

Jan 5, 2023

Accelerating Made Simpler With Celerity

oneAPI for NVIDIA® GPUs

Add support for NVIDIA GPUs to the oneAPI Base Toolkit using oneAPI for NVIDIA. Develop code using SYCL™ and run on NVIDIA GPUs.

TonyM
Jul 22, 2022 · 6 min read · Listen

Intel Arc GPUs and OneAPI — Do They SYCL?

Running oneAPI C++ with SYCL code on Intel Arc and Iris Xe GPUs

New

New

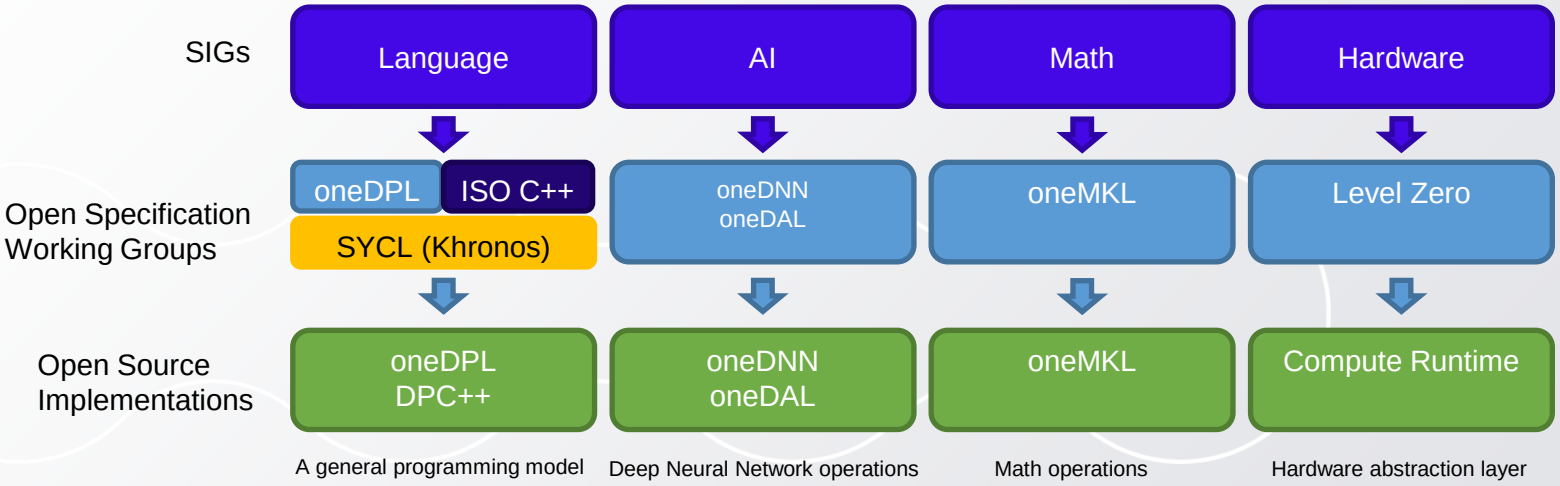
New

New

Working Group Members

Special Interest Groups (SIGs)

Special Interest Groups influence the specifications and implementations



Contribute to the oneAPI Community Forum

- Join and lead SIGs and Working Groups
- Lead technical discussions
- Submit proposals for features and changes
- Vote on proposals

Drive the future of programming for heterogeneous architectures

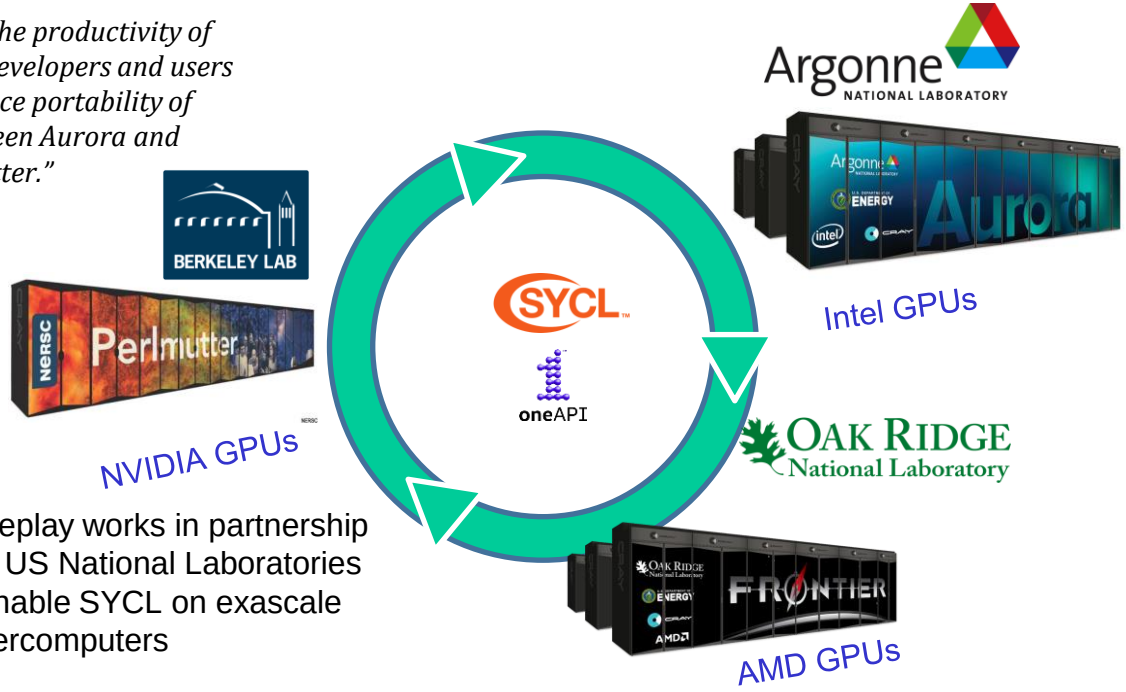
https://oneapi.io/community

oneapi@codeplay.com

SYCL Enables Supercomputers

NERSC National Energy Research Scientific Computing Center

"this work supports the productivity of scientific application developers and users through performance portability of applications between Aurora and Perlmutter."



Codeplay works in partnership with US National Laboratories to enable SYCL on exascale supercomputers

Enables a broad range of software frameworks and applications

kokkos
RAJA
Celerity
alpaka
BabelStream
LAMMPS
GROMACS
fast, flexible & free

KHRONOS GROUP

More workloads need to be SYCL-ready

We have made great inroads with GROMACS, LAMMPS, NWChem workloads.

But we need more!

The Hardware Accelerated Cosmology Code (HACC) framework uses N-body techniques to simulate the formation of structure in collisionless fluids under the influence of gravity in an expanding universe. It depends on external FFT library and is typically compute limited achieving 13.92 Petaflops, 69.2% of machine peak on Sequoia.

QMCPACK is a many-body ab initio quantum Monte Carlo code for computing the electronic structure of atoms, molecules, and solids. It is written primarily in C++, and its use of template metaprogramming is known to stress compilers. When run in production, the code is memory bandwidth sensitive, while still needing thread efficiency to realize good performance.

VPIC (Vector Particle-In-Cell) is a general purpose particle-in-cell simulation code f modeling kinetic plasmas. It employs a second-order, explicit, leapfrog algorithm to update charged particle positions and velocities in order to solve the relativistic kinetic equation for each species in the plasma, along with a full Maxwell description for the electric and magnetic fields evolved via a second- order finite-difference-time-domain (FDTD) solve.

Laghos solves the time-dependent Euler equation of compressible gas dynamics in a moving Lagrangian frame using unstructured high-order finite element spatial discretization and explicit high-order time-stepping. It is built on top of a general discretization library (MFEM) and supports two modes: *full assembly*, where performance is limited by the data

Kripke is a structured deterministic (Sn) transport using RAJA. It contains wavefron algorithms, that stress memory latency and/or bandwidth, and network latency.

Durham: SWIFT, Gadget(v3 and 4), arepo, bam, gizmo, ramses

Others: Grid, sphNG, BAM, Hydra, ATON, Phantom, Fargo3d,Pluto, cosmos++, borg-wl, prompi, GRChombo, swift, gadget, gizmo, ramses, trove, milc, hirep,

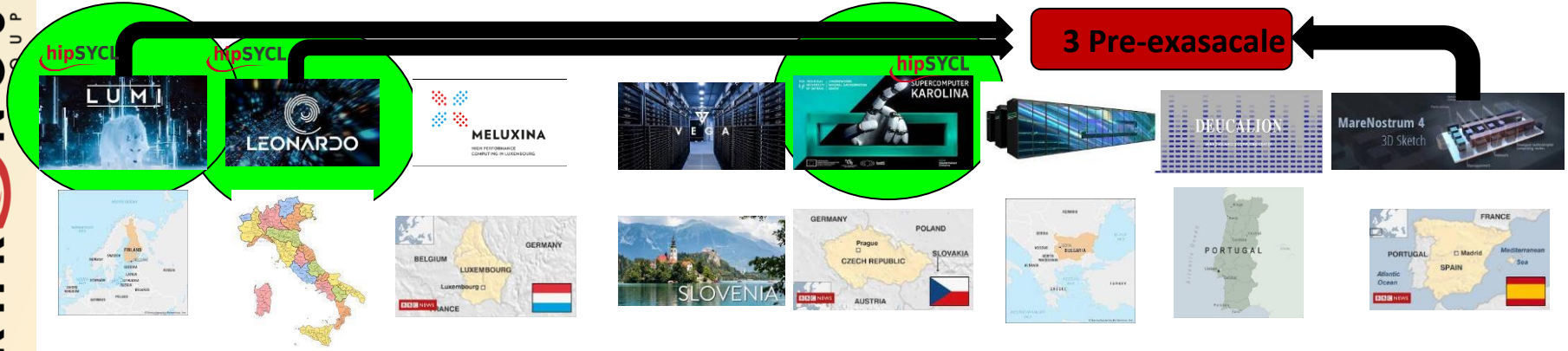
- CASINO(*)
- CASTEP
- CESM2(*)
- Chemshell
- Code_Saturne
- CP2K
- CRYSTAL(*)
- FHI-aims
- GROMACS
- LAMMPS
- MITgcm
- Met Office Unified Mode
- NAMD
- Nektar++
- NEMO
- NWChem
- ONETEP
- OpenFOAM
- ORCA(*)
- Quantum Espresso
- VASP

SYCL as a universal programming model for HPC

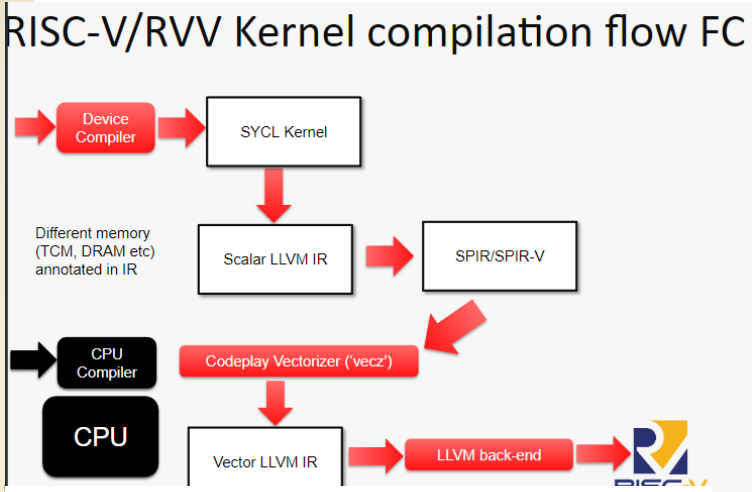
Starting with US National Labs

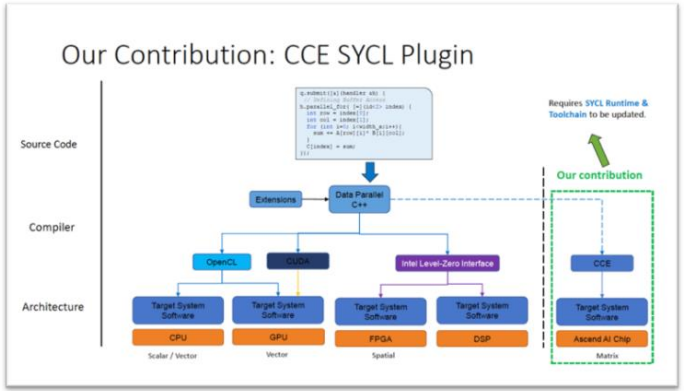
Across Europe, Asia are many Petascale and pre-exascale systems

- With many variety of CPUs GPUs FPGAs, custom devices
- Often with interconnected usage agreements
- Europe EPI: hipSYCL in Leonardo, Lumi and Karolina



SYCL Machine Learning





oneAPI Specification 1.2-rev-1 documentation

oneDPL

The oneAPI DPC++ to support data par implementing data

sycl-blas

An implementation of BLAS using the SYCL open standard for acceleration on OpenCL devices.

C++

sycl-dnn

SYCL-DNN is a library implementing neural network algorithms written using SYCL.

C++

sycl-ml

SYCL-ML is a C++ library, implementing classical machine learning algorithms using SYCL.

C++

oneMKL

oneMKL Interfaces is an open-source implementation of the oneMKL Data Parallel C++ (DPC++) interface.

C++

clvk

clvk is a prototype implementation of OpenCL 3.0 on top of Vulkan using clspv as the compiler.

C++

clspv

Clspv is a prototype compiler for a subset of OpenCL C to Vulkan™ compute shaders.

LLVM

Eigen

Collection of samples and utilities for using ComputeCpp, Codeplay's SYCL implementation.

C++

visioncpp

A machine vision library written in SYCL and C++ that shows performance-portable implementation of graph algorithms.

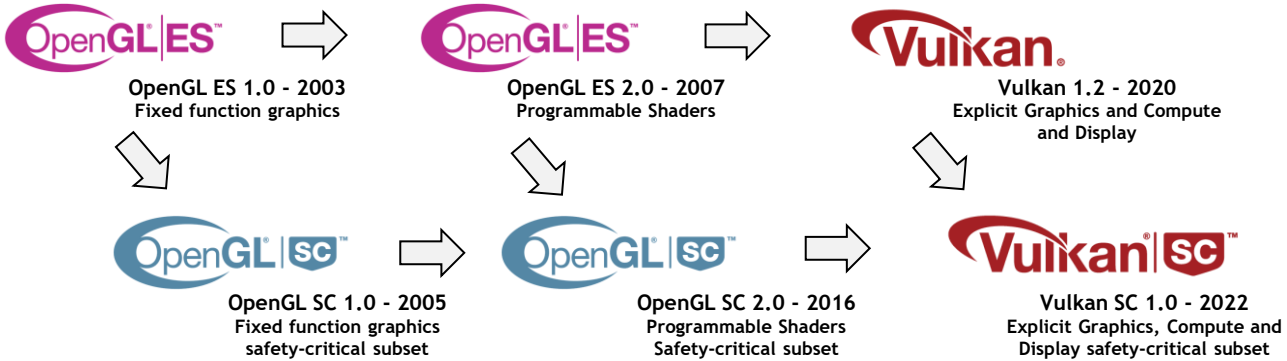
C++

TensorFlow™

Collection of samples and utilities for using ComputeCpp, Codeplay's SYCL implementation.

C++

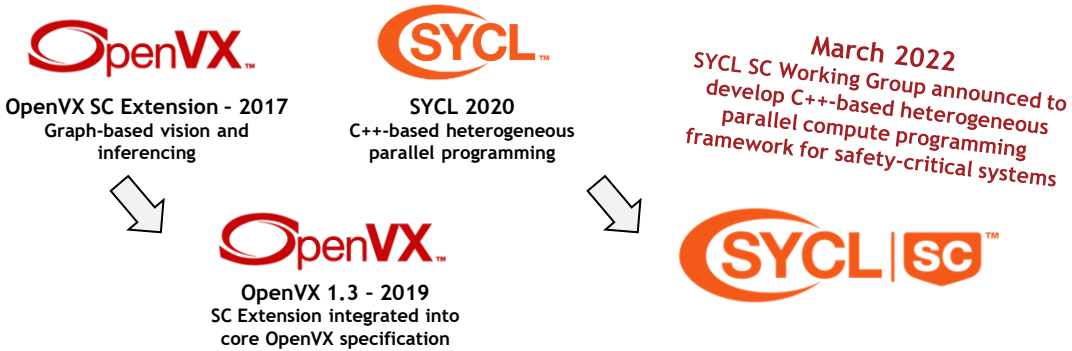
Khronos Safety Critical Standards Evolution



Khronos has 20 years experience in standards for safety-critical markets

Leveraging proven mainstream standards with shipping implementations and developer tooling and familiarity

A choice of abstraction levels to suit different markets and developer needs



Khronos AUTOSAR Liaison: SYCL Demonstrator

Motivation

Currently there is no native AUTOSAR functionality to utilize hardware accelerators for high performance computation.
Only way is to integrate 3rd party libraries which can affect safety.

At the same time there is a challenge for AUTOSAR Adaptive Platform to cover cutting-edge functionality like:

- AD/ADAS systems
- Performing heavy algorithms
- AI
- etc.

Thank you to AUTOSAR and Intellias

intellias
Intelligent Software Engineering

AUTOSAR

The main goal of this concept is to enable parallel heterogeneous programming, using standardized C++ based API, for solving issue of high performance computing.

Important part of the concept is to consider ISO-26262 Standard without sacrificing of performance.

ISO 26262

Final words

Programming Models Must Persist but also be high quality and portable with conformance tests

SYCL 2020 Launched February 2021

SYCL user and developer Phenomenal Growth

Easy to build SYCL on any device

SYCL is mainstream

Market needs SYCL to Evolve with more workloads

SYCL thriving community is our most important asset

Future SYCL: Emerging transformative technologies

SYCL can be a part of a standard programming model for all HPC, Embedded AI/ML, and Automotive

SYCL is an open standard with multiple company contributions, lots of European/Asia projects

Enabling Industry Engagement (2023/04/18)

- SYCL working group values industry feedback
  - https://community.khronos.org/c/sycl
  - https://sycl.tech
- SYCL Academy
  - https://github.com/codeplaysoftware/syclacademy
- SYCL FAQ
  - https://www.khronos.org/blog/sycl-2020-what-do-you-need-to-know
- SYCL CTS
  - https://github.com/KhronosGroup/SYCL-CTS

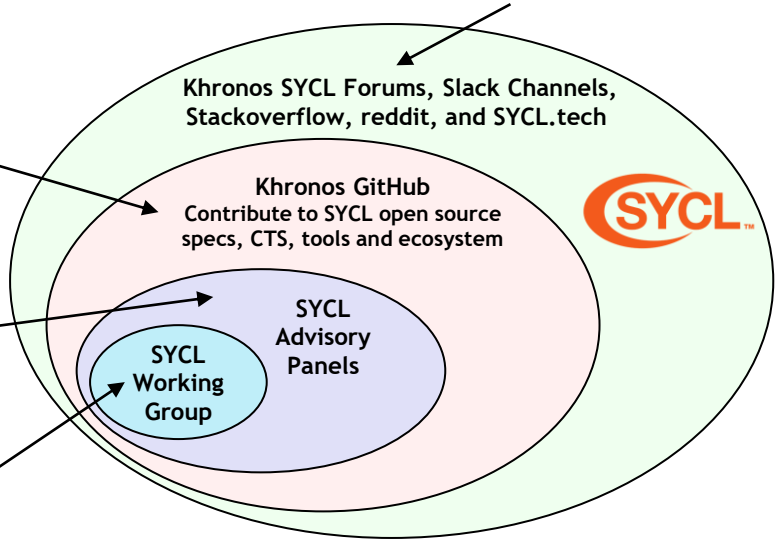
- Advisory Panel Chaired by Tom Deakin of U of Bristol
  - Apr 25, 2023 4pm UK
- Regular meetings to give feedback on roadmap and draft specifications

Public contributions to Specification, Conformance Tests and software
https://github.com/KhronosGroup/SYCL-CTS
https://github.com/KhronosGroup/SYCL-Docs
https://github.com/KhronosGroup/SYCL-Shared
https://github.com/KhronosGroup/SYCL-Registry
https://github.com/KhronosGroup/SyclParallelSTL
https://github.com/intel/llvm

Invited Experts
https://www.khronos.org/advisors/

Khronos members
https://www.khronos.org/members/
https://www.khronos.org/registry/SYCL/

Open to all!
https://community.khronos.org/www.khr.io/slack
https://app.slack.com/client/TDMDFS87M/CE9UX4CHG
https://community.khronos.org/c/sycl/
https://stackoverflow.com/questions/tagged/sycl
https://www.reddit.com/r/sycl
https://github.com/codeplaysoftware/syclacademy
https://sycl.tech/



Enjoy the Conference

Tutorial 1: Introduction to SYCL [1996

Course Leaders: **Christopher Edsall, Univer**

09:15 – 17:00 GMT

▸ show / hide abstract

Tutorial 2: SYCL Techniques and B

Tutorial Lead: **Rod Burns, Codeplay Softwa**

of Heidelberg. Ronan Keryell, AMD. Igor Vora

09:15 – 17:00 GMT

▸ show / hide abstract

Aksel Alpay
University of Heidelberg

Andrey Alekseenko
KTH Royal Institute of Technology

Bartosz Sobol
Jagiellonian University

Dmitrii Tolmachev
ETH Zurich

Edward Mascarenhas
Intel

Erwan Fabiani
Université de Bretagne Occidentale

Ewan Crawford
Codeplay Software

Gregor Daiß
University of Stuttgart

John Pennycook
Intel

Leonardo Solis-Vasquez
Technical University of Darmstadt

Lorenzo Carpentieri
University of Salerno

Marcel Breyer
University of Stuttgart

Salvatore Cielo
Leibniz Supercomputing Centre

Sergi Siso
Hartree Centre STFC UKRI

Shufan Yang
Edinburgh Napier University

Wenju He
Intel

KHRONOS GROUP