

# SYCL 2020 and the future

Michael WOng

Acknowledgements:

SYCL WG  
Rod Burns



One-MKL  
 One-DNN  
 OneDPC  
 SYCL-BLAS  
 SYCL-Eigen  
 SYCL-DNN  
 SYCL Parallel STL  
 ...

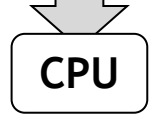
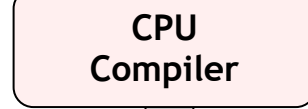
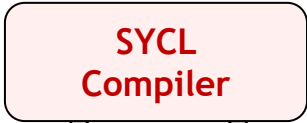


Complex ML frameworks can be directly compiled and accelerated

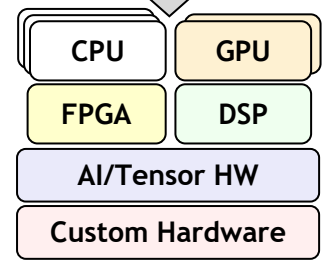
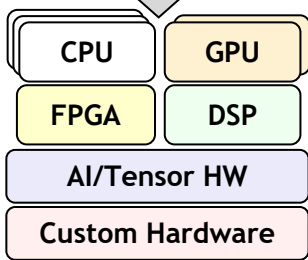


C++ templates and lambda functions separate host & accelerated device code

C++ Kernel Fusion can give better performance on complex apps and libs than hand-coding



Accelerated code passed into device OpenCL compilers



**SYCL is ideal for accelerating larger C++-based engines and applications with performance portability**

# SYCL 2020 is here!

## Open Standard for Single Source C++ Parallel Heterogeneous Programming

- SYCL 2020 is released after 3 years of intense work
- Significant adoption in Embedded, Desktop and HPC markets
- Improved programmability, smaller code size, faster performance
- Based on C++17, backwards compatible with SYCL 1.2.1
- Simplify porting of standard C++ applications to SYCL
- Closer alignment and integration with ISO C++
- Multiple Backend acceleration and API independent

SYCL 2020 increases expressiveness and simplicity for modern C++ heterogeneous programming





# SYCL Academy SYCL 2020 Industry Momentum



**THE NEXT PLATFORM**

HOME COMPUTE STORE CONNECT CONTROL CODE AI HPC ENTERPRISE

LATEST > Can SYCL Slice into Broader Supercomputing? > CODE

HOME > CODE > Can SYCL Slice into Broader Supercomputing?

## CAN SYCL SLICE INTO BROADER SUPERCOMPUTING?

February 3, 2021 Nicole Hennessy

**Embedded COMPUTING DESIGN**

TECHNOLOGY APPLICATION EMBEDDED WORLD RESOURCES

HOME > TECHNOLOGY > OPEN SOURCE > RISC-V & OPEN SOURCE IP

## NSITEXE, Kyoto Microcomputer, Codeplay Software Are Bringing Open Standards Programming to RISC-V Vector Processor for HPC and AI Systems

By Tiera Oliver  
November 03, 2020

**intel**

Intel® oneAPI DPC++: Kernel and API interoperability with C++ technology

By Michael R Carroll  
Published: 03/11/2020 Last Updated: 03/11/2020

### Introduction

This article discusses:

- OpenCL-C kernel ingestion and execution within
- Differences in the pure SYCL\* analogous single source
- Tips including: interoperability features, error handling, and instrumentation

development tools and documentation

**UNIVERSITY OF THE WEST OF SCOTLAND UWS**

Home Profiles Research Units **Research Output** Activities Press / Media

## triSYCL for Xilinx FPGA

**Argonne Leadership Computing Facility**

ALCF Resources Science Community and Partnerships About Support Center

HOME / SUPPORT CENTER / AURORA / SYCL AND DPC++ FOR AURORA

## SYCL and DPC++ for Aurora

Support Center Search

**RENASAS** BE IDEAS FOR EVERY SPACE

PRODUCTS APPLICATIONS DESIGN BUY ABOUT

About Renesas > Press Room > News > Renesas Electronics and Codeplay Collaborate on OpenCL™ and SYCL™ for ADAS Solutions

## Renesas Electronics and Codeplay Collaborate on OpenCL™ and SYCL™ for ADAS Solutions

Open Standard Software Frameworks Facilitate Development Using Renesas' R-Car SoCs to Deliver Computer Vision and Cognitive Processing

September 12, 2017

**Imagination**

**PRESS RELEASE**

23 OCTOBER 2019

## TensorFlow™ gets native support for PowerVR® GPUs via optimised open-source SYCL™ libraries

Open source SYCL neural network libraries optimised for PowerVR, with Codeplay making it easier for developers to port existing code

**NERSC** Powering Scientific Discovery Since 1974

HOME ABOUT COVID-19 RESEARCH SCIENCE SYSTEMS FOR USERS NEWS R&D EVENTS LIVE STATUS

Home > News > Center News > NERSC, ALCF, Codeplay Partner on SYCL for Next-generation Supercomputers

## NERSC, ALCF, CODEPLAY PARTNER ON SYCL FOR NEXT-GENERATION SUPERCOMPUTERS

FEBRUARY 3, 2021  
Contact: cscot@sls@lbl.gov

The National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory (Berkeley Lab) in collaboration with the Argonne Leadership Computing Facility (ALCF) at Argonne National Laboratory, has signed a contract with Codeplay Software to enhance the LLVM SYCL™ GPU compiler capabilities for NVIDIA A100 GPUs.

This collaboration will help NERSC and ALCF users, along with the high-performance computing community in general, produce high-performance applications that are portable across compute architectures from multiple vendors.

Codeplay is a software company based in the U.K. that has a long history of developing compilers and tools for different hardware architectures. The company has been the lead implementer of SYCL compilers and a main contributor to the existing open source support for NVIDIA V100 GPUs through the DPC++ project. NVIDIA A100 GPUs are available in the ThetaGPU extension of A Theta and will power NERSC's next-generation supercomputer, Perlmutter.

**Perlmutter**

NERSC supercomputers are used for scientific research by researchers working in diverse areas such as alternative energy, environment, high-energy and nuclear physics, advanced computing, materials science, and chemistry. Over the past year, 20

SYCL (pronounced 'sickle') is a royalty-free cross-platform abstraction layer that builds on efficiency of OpenCL. It enables code written in a "single-source" style using C++ (see <https://www.khronos.org/sycl/>). Multiple SYCL implementations exist, including Codeplay, and LLVM and support for

**Argonne and Oak Ridge National Laboratories Award Codeplay® Software to Further Strengthen SYCL™ Support Extending the Open Standard Software for AMD GPUs**

17 June 2021



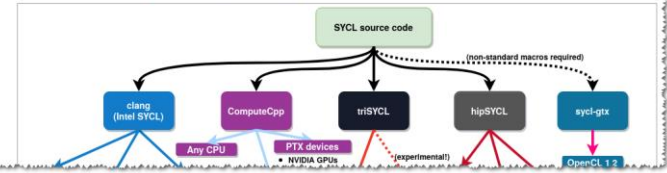
LEMONT, IL, and OAK RIDGE, TN, and EDINBURGH, UK, June 17, 2021 - Argonne National Laboratory (ANL) in collaboration with Oak Ridge National Laboratory (ORNL) has awarded Codeplay a contract implementing the oneAPI DPC++ compiler, an implementation of

**hipSYCL Seeing New Runtime For This SYCL Implementation For CPUs + ROCm/CUDA GPUs**

Written by Michael Larabel in Programming on 24 August 2020 at 08:25 AM EDT. 4 Comments

The hipSYCL effort has been about supporting the Khronos SYCL single-source language built on C++ across any CPU with OpenMP as well as AMD Radeon GPUs via ROCm and NVIDIA GPUs via CUDA. The hipSYCL effort has a new "Lite" experimental runtime under development.

The hipSYCL effort is just one of several SYCL efforts in the ecosystem for running SYCL across CPUs, GPUs, and other accelerators. Here's a look via the hipSYCL project how their effort compares to others.



**SYCL support growing from Embedded Systems through Desktops to Supercomputers**

- <https://www.alcf.anl.gov/support-center/aurora/sycl-and-dpc-aurora>
- <https://www.embeddedcomputing.com/technology/open-source/ip/nsitexe-kyoto-microcomputer-and-codeplay-software-are-bringing-open-standards-programming-to-risc-v-vector-processor-for-hpc-and-ai-systems>
- <https://www.nextplatform.com/2021/02/03/can-sycl-slice-into-broader-supercomputing/>
- [https://www.phoronix.com/scan.php?page=news\\_item&id=hipSYCL-New-Lite-Runtime](https://www.phoronix.com/scan.php?page=news_item&id=hipSYCL-New-Lite-Runtime)
- <https://software.intel.com/content/www/us/en/develop/articles/interoperability-dpcpp-sycl-opencl.html>
- <https://www.renesas.com/br/en/about/press-room/renesas-electronics-and-codeplay-collaborate-opencl-and-sycl-adas-solutions>
- <https://www.ner.gov/news-publications/news-center/news/2021/nersc-alcf-codeplay-partner-on-sycl-for-next-generation-supercomputers/>
- <https://research.pwr.edu.pl/en/publications/trisycl-for-xilinx-fpga>
- <https://www.imaginationtech.com/news/press-release/tensorflow-gets-native-support-for-powervr-gpus-via-optimised-open-source-sycl-libraries/>



## SYCL 2020 Major Features

- **Unified Shared Memory (USM)**
  - Code with pointers can work naturally without buffers or accessors
  - Simplifies porting from most code (e.g. CUDA, C++)
- **Parallel Reductions**
  - Added built-in reduction operation to avoid boilerplate code and achieve maximum performance on hardware with built-in reduction operation acceleration.
- **Work group and subgroup algorithms**
  - Efficient parallel operations between work items
- **Class template argument deduction (CTAD) and template deduction guides**
  - Simplified class template instantiation
- **Simplified use of Accessors with a built-in reduction operation**
  - Reduces boilerplate code and streamlines the use of C++ software design patterns
- **Expanded interoperability**
  - Efficient acceleration by diverse backend acceleration APIs
- **SYCL atomic operations are now more closely aligned to standard C++ atomics**
  - Enhances parallel programming freedom

# Parallel Industry Initiatives



C++11



C++14



C++17



C++20



C++23



SYCL 1.2  
C++11 Single source  
programming



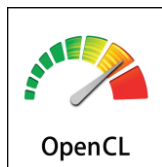
SYCL 1.2.1  
C++11 Single source  
programming



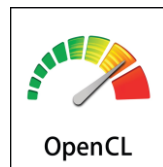
SYCL 2020  
C++17 Single source  
programming  
Many backend options



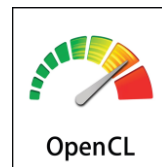
SYCL 202X  
C++20 Single source  
programming  
Many backend options



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



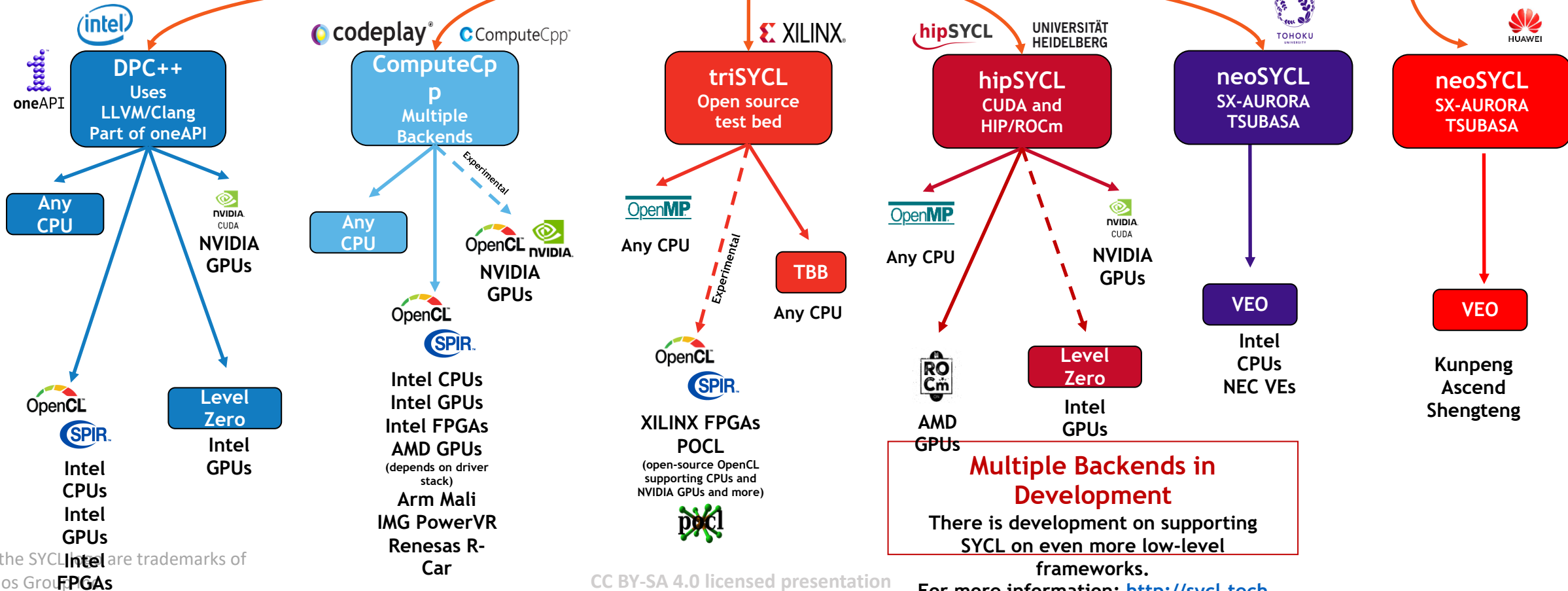
# SYCL Implementations in Development

SYCL, OpenCL and SPIR-V, as open industry standards, enable flexible integration and deployment of multiple acceleration technologies

SYCL enables Khronos to influence ISO C++ to (eventually) support heterogeneous compute



**SYCL**  
Source Code



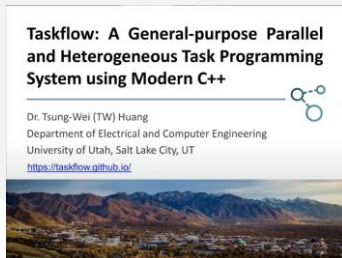
**Multiple Backends in Development**  
There is development on supporting SYCL on even more low-level frameworks.

# SYCL Ecosystem, Research and Benchmarks

## Implementations



neoSYCL  
SX-AURORA TSUBASA



ComputeCpp™

hipSYCL

GROMACS  
FAST. FLEXIBLE. FREE.

alBaka

## Benchmarks/Books

Direct Programming Benchmark



SYCL-Bench

## Linear Algebra Libraries

## Machine Learning Libraries and Parallel Acceleration Frameworks

BLAS	FFT	Math	RAND
SYCLBLAS oneMKL	oneMKL	oneMKL	oneMKL
SOLVER	SPARSE	TENSOR	STL
oneMKL	oneMKL	SYCL-DNN Eigen oneDNN TensorFlow	SYCL Parallel STL oneDPL



Working Group Members

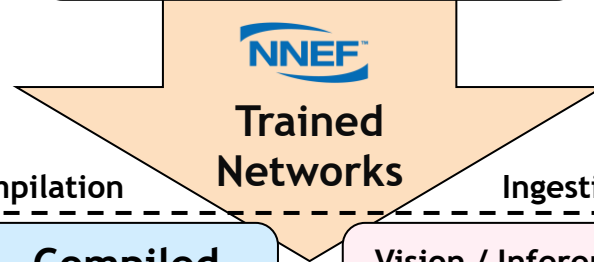


# SYCL in Embedded Systems, Automotive, and AI

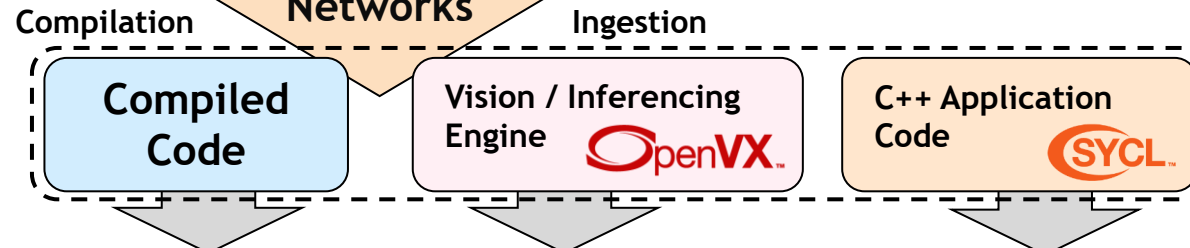
Networks trained on high-end desktop and cloud systems



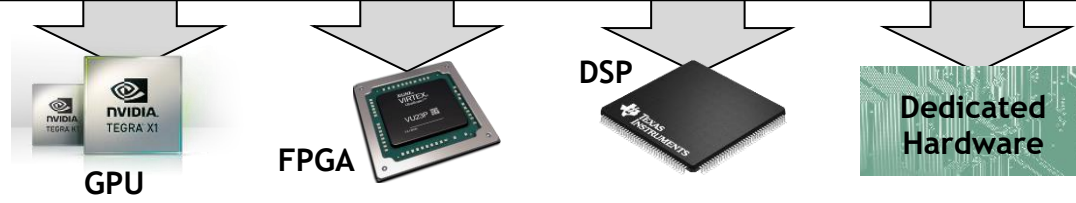
Open industry standards, enable flexible integration and deployment of multiple acceleration technologies



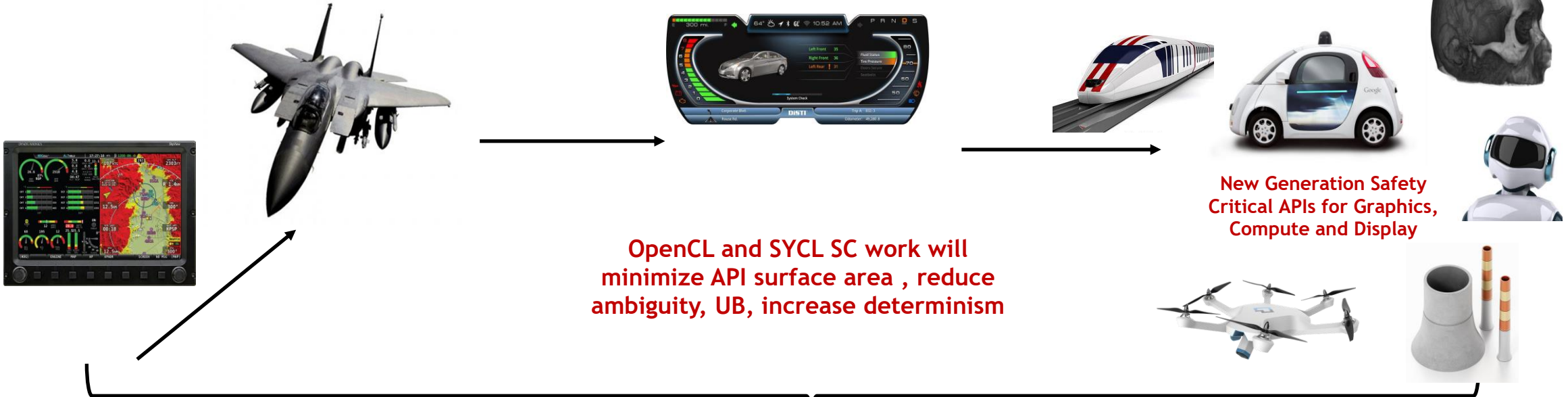
Applications link to compiled inferencing code or call vision/inferencing API



Diverse Embedded Hardware  
Multi-core CPUs, GPUs  
DSPs, FPGAs, Tensor Cores  
\* Vulkan only runs on GPUs

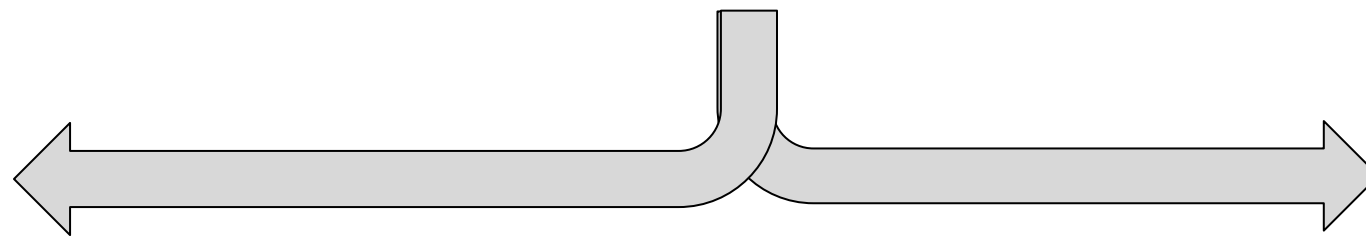


# SYCL Academy Safety Critical API Evolution



OpenCL and SYCL SC work will minimize API surface area, reduce ambiguity, UB, increase determinism

New Generation Safety Critical APIs for Graphics, Compute and Display



Rendering Compute Display



Industry Need for GPU Acceleration APIs designed to ease system safety certification is increasing

ISO 26262 / ASIL-D

CC BY-SA 4.0 licensed presentation



ISO/PAS 21448

UL 4600



ISO/IEC JTC 1/SC 42  
Artificial intelligence

- © ISO/IEC AWG TR 20447-1 (under development) Information technology – Big data reference architecture – Part 1: Framework and application process
- © ISO/IEC TR 20547-2:2015 Information technology – Big data reference architecture – Part 2: Use cases and derived requirements
- © ISO/IEC DIS 20447-3 (under development) Information technology – Big data reference architecture – Part 3: Reference architecture
- © ISO/IEC TR 20547-5:2015 Information technology – Big data reference architecture – Part 5: Standards roadmap
- © ISO/IEC AWG 22989 (under development) 46746766Statistical Intelligence Concepts and Terminology
- © ISO/IEC AWG 23053 (under development) Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)



# SYCL Academy

## SYCL in HPC/Supercomputers

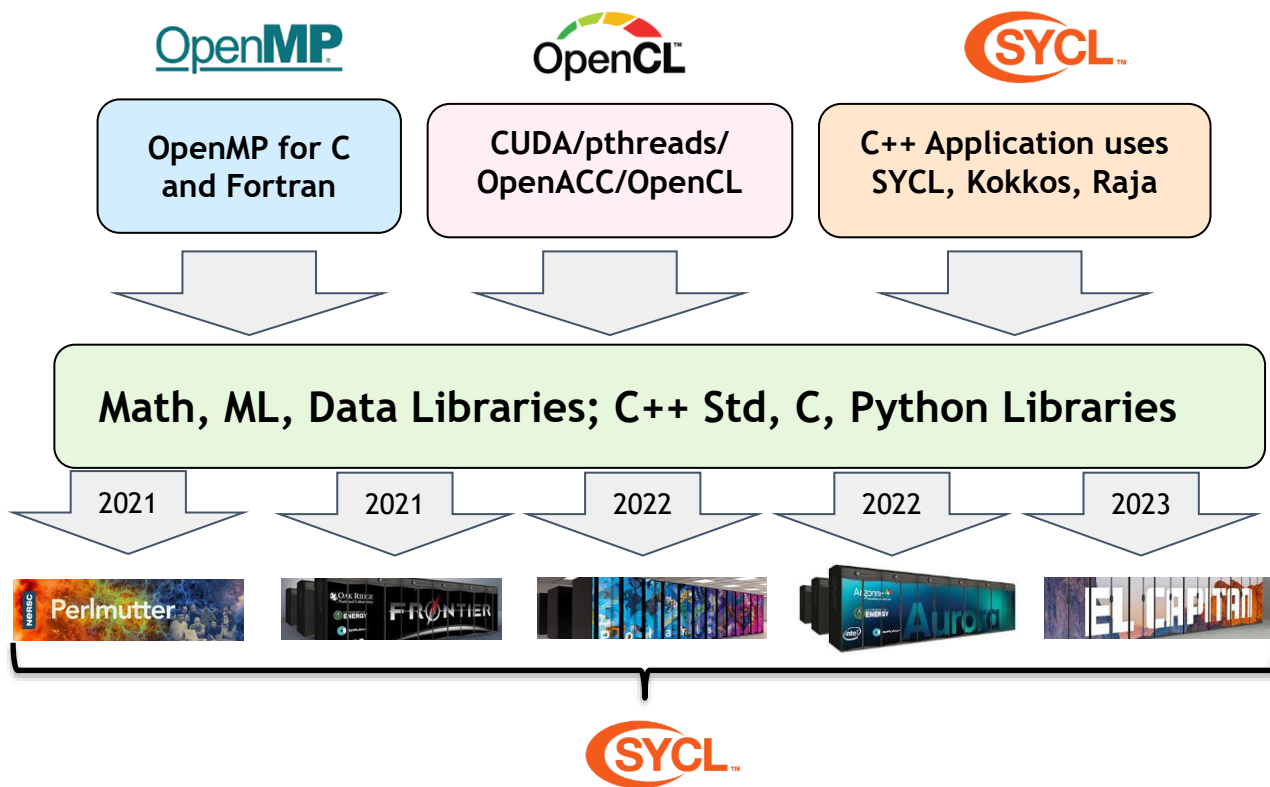


**Simulation**  
HPC Languages  
Solver Libraries, Parallel RT

**Data**  
Productivity Languages  
Big Data Stack, Stats Lib, Databases

**Learning**  
Productivity Languages  
Deep Learning, Linear Alg, ML

Three Pillars of Science Problem

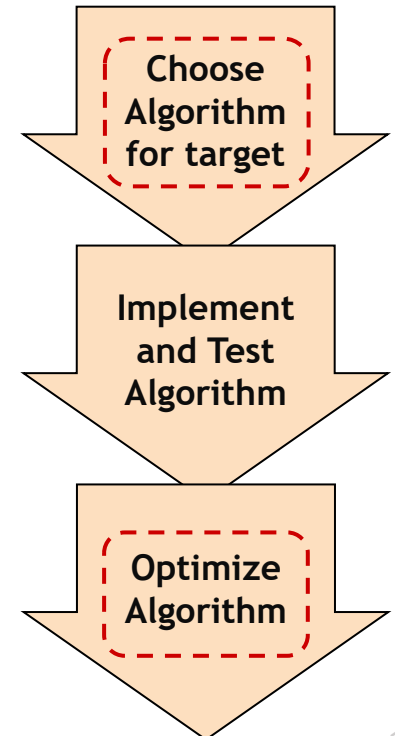


Need Languages that allow control of these Data Issues  
Set Data affinity, Data Layout, Data movement, Data Locality, highly Parameterized Code and dynamically compose the algorithms (C++ templates, parallel STL, inlining and fusion, abstractions)

Libraries augment compiler optimizations for Performance Portable programs

Use open standards to run Performance Portable code on new generation, or different vendor's, hardware with compiler optimization, explicit parametrization and dynamically composed algorithm

Today's Supercomputing Development Workflow needs knowledge of system architecture and tools that control data



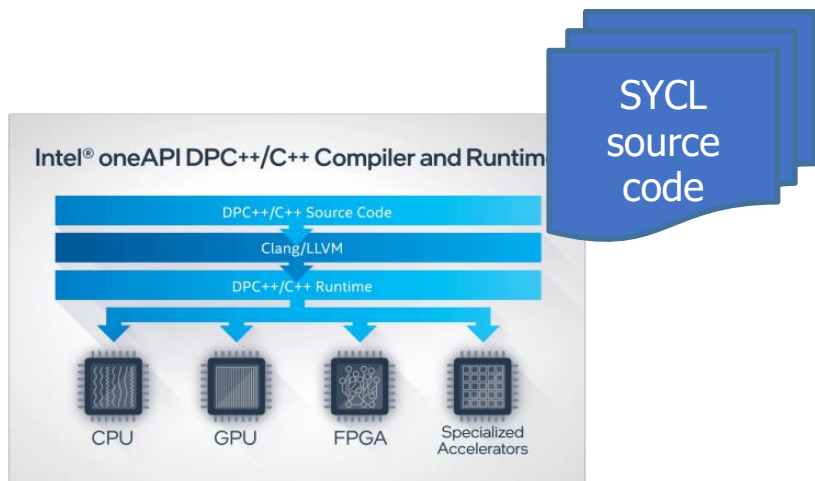
# oneAPI and SYCL



1  
oneAPI



- SYCL sits at the heart of oneAPI
- Provides an open standard interface for developers
- Defined by the industry



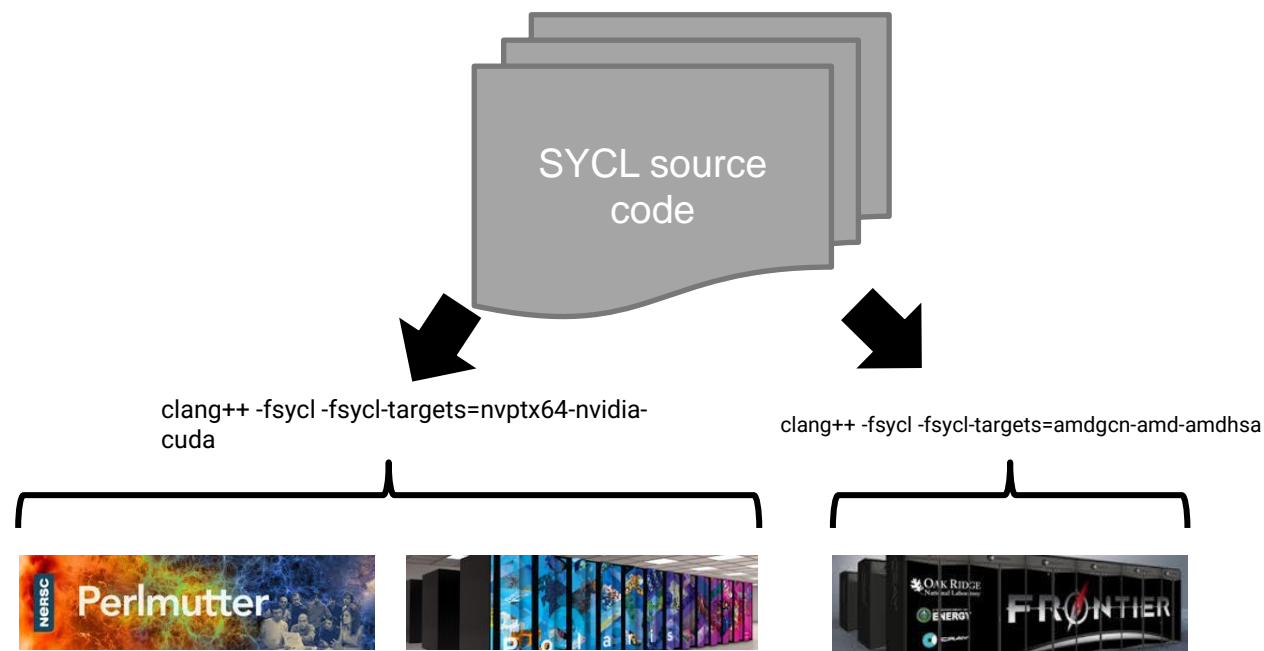


## Nvidia and AMD Support in oneAPI



- Extending DPC++ to target Nvidia and AMD GPUs
- Supporting Perlmutter, Polaris and Frontier supercomputers
- Open source and available to everyone

Different targets using a simple compiler flag

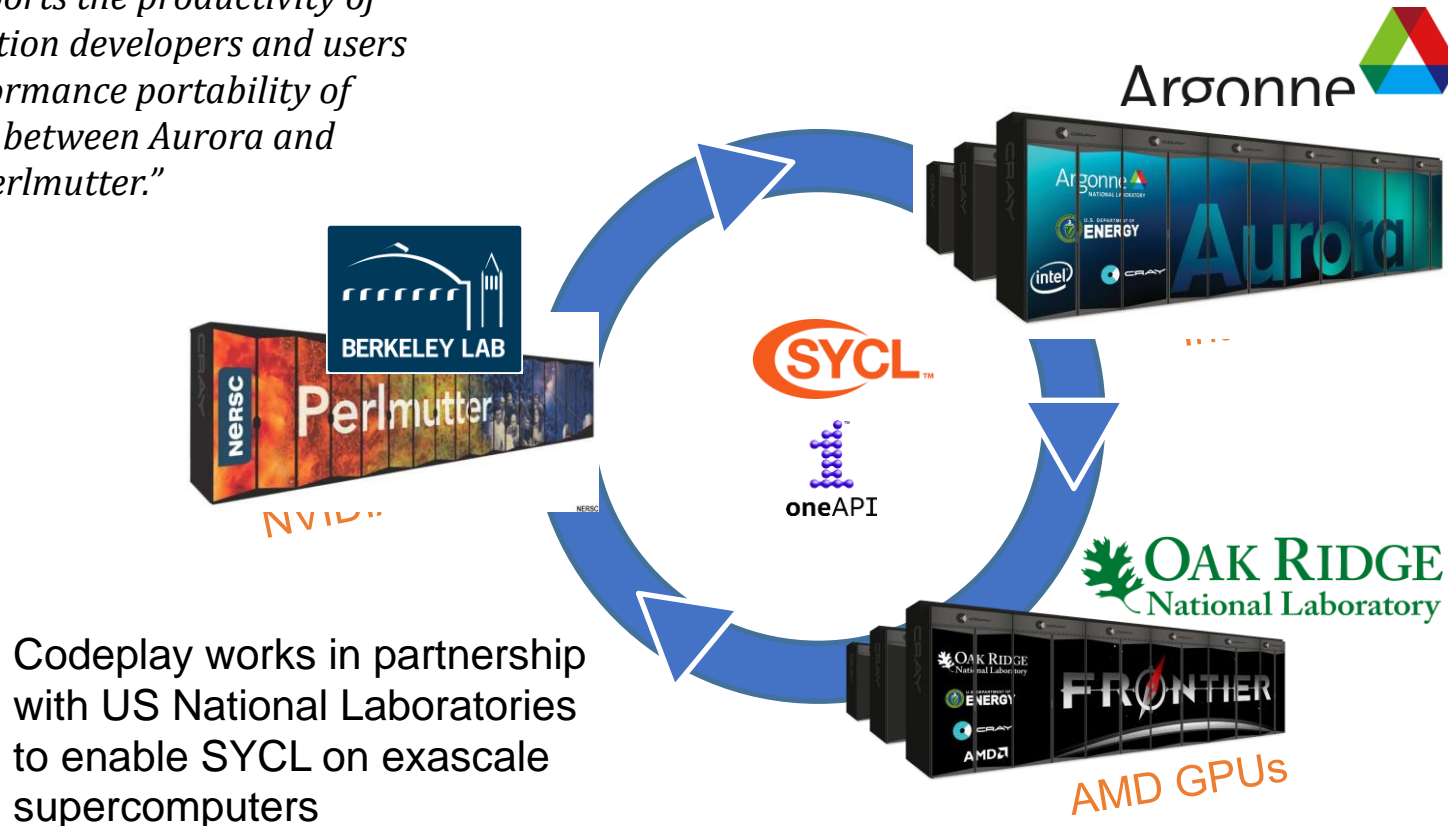


<https://www.codeplay.com/oneapiforcuda>  
Resources for AMD coming soon



# SYCL Enables Supercomputers

*“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



# SYCL Academy

## SYCL Future Evolution



### SYCL 2020 compared with SYCL 1.2.1

- Easier to integrate with C++17 (CTAD, Deduction Guides...)
- Less verbose, smaller code size, simplify patterns
- Backend independent
- Multiple object archives aka modules simplify interoperability
- Ease porting C++ applications to SYCL
- Enable capabilities to improve programmability
- Backwards compatible but minor API break based on user feedback



SYCL 2020

### Over 40 Selected Features for SYCL 2020

- Unified Shared Memory)
- Parallel Reductions adds a built in reduction operation
- Work-group and sub-group algorithms
- Improvements to atomic operations
- Class template argument deduction (CTAD) and deduction guides
- Simplification of accessors
- Expanded interoperability with different backends
- Extension mechanism
- Address spaces
- Vector rework
- Specialization Constants

### SYCL Future Roadmap (MAY CHANGE)

#### Improving Software Ecosystem

Books, Tutorials, Tool, libraries, GitHub

#### Expanding Implementations

DPC++  
ComputeCpp  
triSYCL  
hipSYCL  
neoSYCL

#### Regular Maintenance Updates

Spec clarifications, formatting and bug fixes

<https://www.khronos.org/registry/SYCL/>

Repeat The Cycle every 1.5-3 years



NEXT

#### Conformance Tests

Working on Implementations

Future SYCL NEXT Proposals

Integration of successful Extensions plus new Core functionality

Converge SYCL with ISO C++ and continue to support OpenCL to deploy on more devices

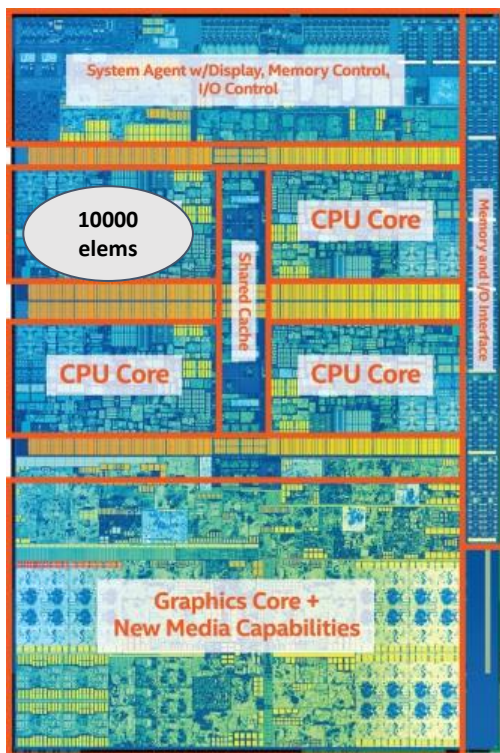
CPU  
GPU  
FPGA  
AI processors  
Custom Processors

...

# A Demo with C++ Parallel STL



## What can I do with a Parallel For Each?



Intel Core i7 7th generation

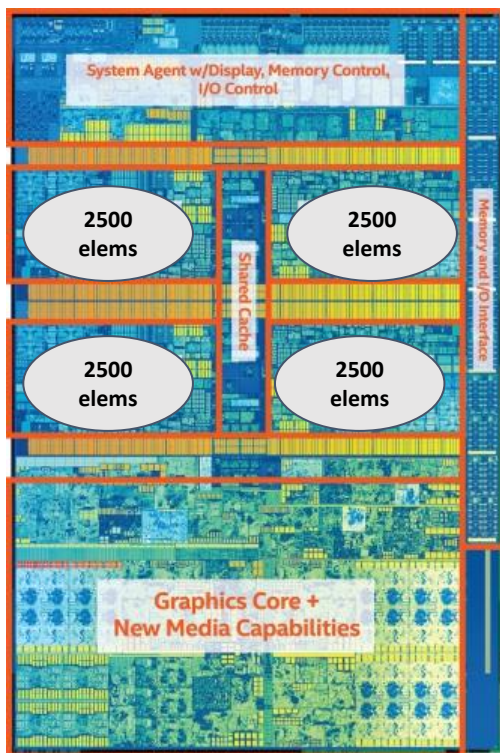
```
size_t nElems = 1000u;
std::vector<float> nums(nElems);
```

```
std::fill_n(std::begin(v1), nElems, 1);
```

```
std::for_each(std::begin(v), std::end(v),
              [=](float f) { f * f + f });
```

**Traditional for each uses only one core,  
rest of the die is unutilized!**

## What can I do with a Parallel For Each?



Intel Core i7 7th generation

```
size_t nElems = 1000u;
std::vector<float> nums(nElems);
```

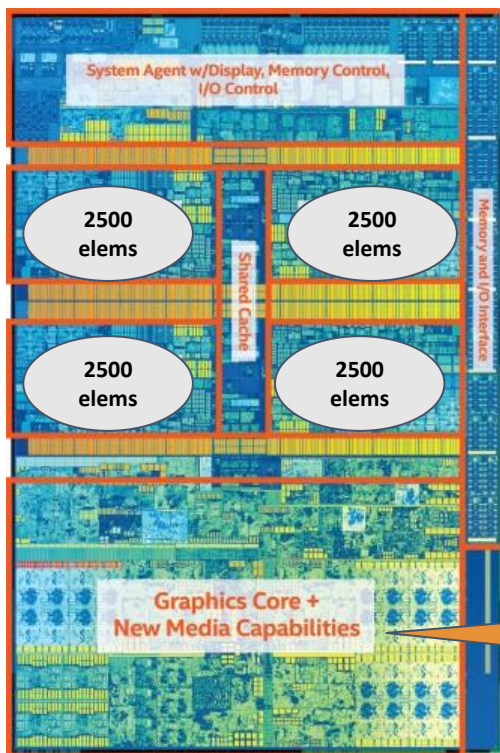
```
std::fill_n(std::execution_policy::par,
            std::begin(v1), nElems, 1);
```

```
std::for_each(std::execution_policy::par,
              std::begin(v), std::end(v),
              [=](float f) { f * f + f });
```

**Workload is distributed across cores!**

(mileage may vary, implementation-specific behaviour)

## What can I do with a Parallel For Each?



Intel Core i7 7th generation

```
size_t nElems = 1000u;
std::vector<float> nums(nElems);
```

```
std::fill_n(std::execution_policy::par,
            std::begin(v1), nElems, 1);
```

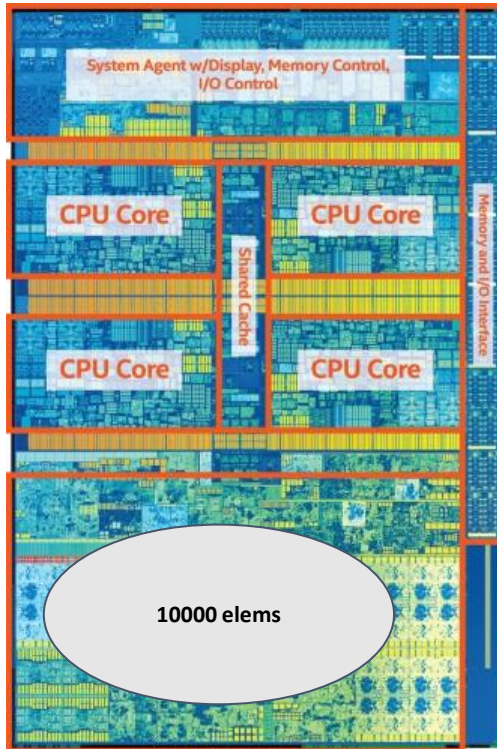
```
std::for_each(std::execution_policy::par,
              std::begin(v), std::end(v),
              [=](float f) { f * f + f });
```

What about this part?

**Workload is distributed across cores!**

(mileage may vary, implementation-specific behaviour)

## What can I do with a Parallel For Each?



Intel Core i7 7th generation

```
size_t nElems = 1000u;
std::vector<float> nums(nElems);

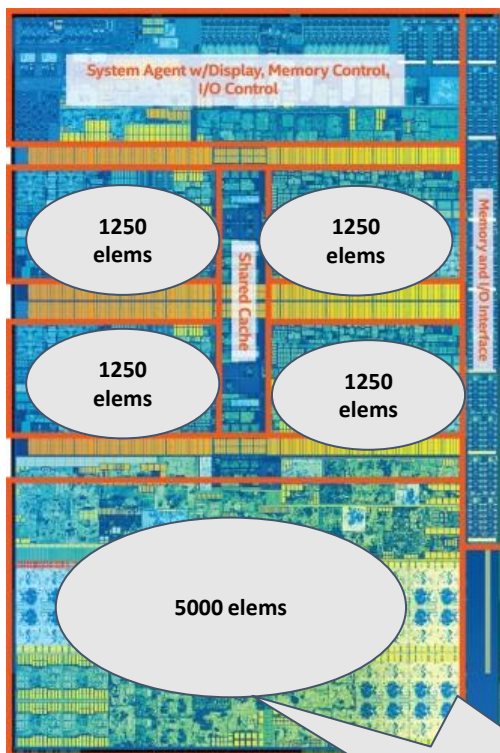
std::fill_n(sycl_policy,
            std::begin(v1), nElems, 1);
```

```
std::for_each(sycl_named_policy
              <class KernelName>,
              std::begin(v), std::end(v),
              [=](float f) { f * f + f });
```

**Workload is distributed on the GPU cores**

(mileage may vary, implementation-specific behaviour)

## What can I do with a Parallel For Each?



Intel Core i7-7700

**Experimental!**

```
size_t nElems = 1000u;
std::vector<float> nums(nElems);
```

```
std::fill_n(sycl_heter_policy(cpu, gpu, 0.5),
            std::begin(v1), nElems, 1);
```

```
std::for_each(sycl_heter_policy<class kName>
              (cpu, gpu, 0.5),
              std::begin(v), std::end(v),
              [=](float f) { f * f + f });
```

**Workload is distributed on all cores!**

(mileage may vary, implementation-specific behaviour)



Demo Results - Running `std::sort`  
(Running on Intel i7 6600 CPU & Intel HD Graphics 520)

size	2 <sup>16</sup>	2 <sup>17</sup>	2 <sup>18</sup>	2 <sup>19</sup>
<code>std::seq</code>	0.27031s	0.620068s	0.669628s	1.48918s
<code>std::par</code>	0.259486s	0.478032s	0.444422s	1.83599s
<code>std::unseq</code>	0.24258s	0.413909s	0.456224s	1.01958s
<code>sycl_execution_policy</code>	0.273724s	0.269804s	0.277747s	<b>0.399634s</b>

# SYCL 2020 is here!

## Open Standard for Single Source C++ Parallel Heterogeneous Programming

- SYCL 2020 is released after 3 years of intense work
- Significant adoption in Embedded, Desktop and HPC markets
- Improved programmability, smaller code size, faster performance
- Based on C++17, backwards compatible with SYCL 1.2.1
- Simplify porting of standard C++ applications to SYCL
- Closer alignment and integration with ISO C++
- Multiple Backend acceleration and API independent

SYCL 2020 increases expressiveness and simplicity for modern C++ heterogeneous programming







- SYCL working group values industry feedback
  - <https://community.khronos.org/c/sycl>
  - <https://sycl.tech>
- SYCL FAQ
  - <https://www.khronos.org/blog/sycl-2020-what-do-you-need-to-know>
- What features would you like in future SYCL versions?

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/>

