

# SYCL – Introduction and Best Practices

---

Thomas Applencourt - [apl@anl.gov](mailto:apl@anl.gov)

Argonne Leadership Computing Facility  
Argonne National Laboratory  
9700 S. Cass Ave  
Argonne, IL 60349

# Table of contents

1. Introduction

2. Theory

3. Live-demo

4. Conclusion

# Introduction

---

# What programming model to use to target GPU?

- OpenMP (pragma based)
- Cuda (proprietary)
- Hip (low level)
- OpenCL (low level)
- Kokkos, raja, OCCA (high level, abstraction layer, academic project)

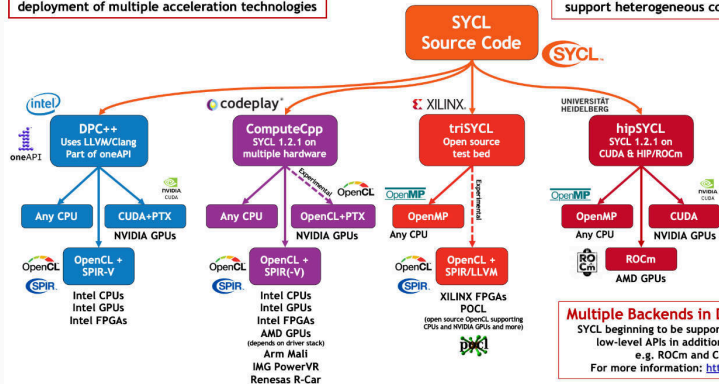
# What is SYCL™?

1. Target C++ programmers (template, lambda)
  - 1.1 No language extension
  - 1.2 No pragmas
  - 1.3 No attribute
2. Borrow lot of concept from battle tested OpenCL (platform, device, work-group, range)
3. Single Source (two compilation pass)
4. **Implicit data-transfer**
5. SYCL is a Specification developed by the Khronos Group (OpenCL, SPIR, Vulkan, OpenGL)

# SYCL Implementation

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



<sup>1</sup>Credit: Khronos groups (<https://www.khronos.org/sycl/>)

# Goal of this talk

1. Give you a feel of SYCL
2. Go through code examples
3. Teach you enough so that can search for the rest if you interested
4. Question are welcomed! <sup>2</sup>

---

<sup>2</sup>Please use the google documents or the chat. Questions will be answered during breaks or by email if we are short on time. Or if I don't know...

# Theory

---



# A picture is worth a thousand words<sup>3</sup>

## OpenCL Class Diagram

The figure below describes the OpenCL specification as a class diagram using the Unified Modeling Language<sup>1</sup> (UML) notation. The diagram shows both nodes and edges which are classes and their relationships. As a simplification it shows only classes, and no attributes or operations.

### Annotations

#### Relationships

abstract classes {abstract}

aggregations ♦

inheritance Δ

relationship navigability ^

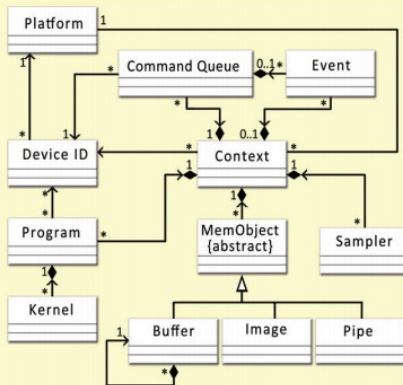
#### Cardinality

many \*

one and only one 1

optionally one 0..1

one or more 1..\*



<sup>1</sup> Unified Modeling Language (<http://www.uml.org/>) is a trademark of Object Management Group (OMG).

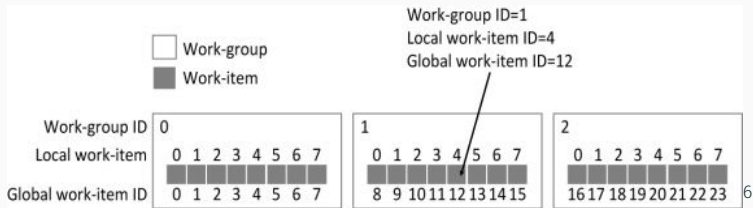
<sup>3</sup> and this is a UML diagram so maybe more!

# Memory management: SYCL innovation

1. Buffers **encapsulate** your data
2. Accessors **describe** how you access those data
3. Buffer destruction will cause **synchronization**

# Implicit Loop

- A Kernel is invoked once for each **work item**<sup>4</sup>
- **local work size** Work items are grouped into a **work group**<sup>5</sup>
- The total number of all work items is specified by the **global work size**



<sup>4</sup>similar to *MPI\_rank*

<sup>5</sup>similar to *pragma omp simdlen/safelen*

<sup>6</sup>Credit The OpenCL Programming Book by Fixstars

# Implicit Loop: Example!

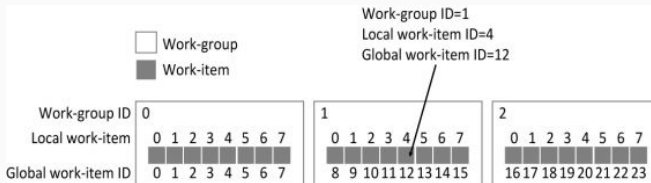
```
1  global_work_size = 24 ; local_work_size = 8
```

SYCL / Opencl / CUDA

```
1  parallel_for<global_work_size,local_work_size>(mykernel);
```

OpenMP<sup>7</sup>

```
1  # wG = work_group ; wC = work_item
2  for (wG_id=0; wG_id++; wG_id < (global_work_size / local_work_size)
3      for (local_wI_id=0; local_wI_id++; local_wI_id < local_work_size)
4          global_wI_id = local_wI_id + wG_id*local_wG_size
```



<sup>7</sup>Using chunking / tilling / vectorization technique

# Live-demo

---

1. Using Argonne Cluster for convenience
2. Using Intel SYCL (DPC++) compiler
3. Running on Intel Integrated Graphic Iris Gen9
4. Example available at  
*<https://github.com/alcf-perfengr/sycltrain>*

## Conclusion

---

# Conclusion

1. SYCL is C++
2. Many vendors (Intel, Nvidia, AMD) and hardware (CPU, GPU, FPGA) supported
3. Implicit data-movement by default (Buffer / Accessors concepts)



# Lot of goods resources online

## Spec

1. <https://www.khronos.org/registry/SYCL/specs/sycl-1.2.1.pdf>
2. <https://www.khronos.org/files/sycl/sycl-121-reference-card.pdf>

## Examples

1. <https://github.com/codeplaysoftware/computecpp-sdk/tree/master/samples>
2. <https://github.com/alcf-perfengr/sycltrain>

## Documentations

1. <https://sycl.tech/>
2. Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL (ISBN 978-1-4842-5574-2)

Thanks you! Do you have any questions?