SYCL – Introduction and Best Practices

Thomas Applencourt - 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.

---

1Credit: 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! ²

²Please use google-doc or chat. Will be answers during section breaks or by email if we are short on time
Theory
A picture is worth a thousand words\(^3\)

---

OpenCL Class Diagram

The figure below describes the OpenCL specification as a class diagram using the Unified Modeling Language\(^1\) (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

<table>
<thead>
<tr>
<th>Relationships</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract classes</td>
<td>{abstract}</td>
</tr>
<tr>
<td>aggregations</td>
<td><img src="image" alt="association" /></td>
</tr>
<tr>
<td>inheritance</td>
<td><img src="image" alt="inheritance" /></td>
</tr>
<tr>
<td>relationship navigability</td>
<td><img src="image" alt="navigability" /></td>
</tr>
</tbody>
</table>

### Cardinality

| many        | * |
| one and only one | 1 |
| optionally one       | 0..1 |
| one or more         | 1..* |

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

\(^3\) and this is a UML diagram so maybe more!
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
- local work size Work items are grouped into a work group
- The total number of all work items is specified by the global work size

---

4 similar to MPI_rank
5 similar to pragma omp simdlen/safelen
6 Credit The OpenCL Programming Book by Fixstars
Implicit Loop

```
1 global_work_size = 24; local_work_size = 8

SYCL / Opencl / CUDA:

1 parallel_for<global_work_size,local_work_size>(mykernel);

Explicit loop\(^7\)

```
Live-demo
Bookkeeping

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. Lost of Vendors / Hardware supported (Intel, nvidia, AMD / CPU, GPU, FPGA)
3. Implicit data-movement by default (Buffer / Accessors concepts)
Lot of goods resources online

Spec


Examples

2. https://github.com/alcf-perfengr/sycltrain

Documentations

1. https://sycl.tech/
Thanks you! Do you have any questions?