SYCL – Introduction and Hands-on

Thomas Applencourt + people on next slide - apl@anl.gov

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

Get the example and the presentation:

1. `git clone https://github.com/alcf-perfengr/sycltrain`
2. `cd sycltrain/presentation/2020_07_30_ATPESC`

People who are here to help:

- Collen Bertoni - bertoni@anl.gov
- Brian Homerding - b homerding@anl.gov
- Nevin ":)" Liber - n liber@anl.gov
- Ben Odom - benjamin.j.odom@intel.com
- Dan Petre - dan.petre@intel.com
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
</tr>
<tr>
<td>2.</td>
<td>Theory</td>
</tr>
<tr>
<td>3.</td>
<td>Hands-on</td>
</tr>
<tr>
<td>4.</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
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 projects)
What is SYCL™?¹

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 compilations passes)
4. **High level data-transfer**
5. SYCL is a Specification developed by the Khronos Group (OpenCL, SPIR, Vulkan,OpenGL)

¹SYCL Doesn’t mean “Someone You Couldn’t Love”. Sadly.
SYCL Implementations

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.

Multiple Backends in Development
SYCL beginning to be supported on multiple low-level APIs in addition to OpenCL e.g. ROCm and CUDA
For more information: http://sycl.tech

Credit: Khronos groups (https://www.khronos.org/sycl/)
Goal of this presentation

1. Give you a feel of SYCL
2. Go through code examples (and make you do some homework)
3. Teach you enough so that can search for the rest if you interested
4. Question are welcomed! ³

³Please just talk, or use slack
Theory
A picture is worth a thousand words\textsuperscript{4} and this is a UML diagram so maybe more!

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

---

5 similar to \textit{MPI\_rank}
6 similar to \texttt{pragma omp simdlen/safelen}
7 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
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
```

Using chunking / tiling / vectorization technique
Hands-on
Two Cluster are available for you:

1. **cooley.alcf.anl.gov**. Nvidia GPU (Tesla K80)
2. **devcloud.intel.com**. Intel iGPU (Intel Iris "Gen9")

If you feel adventurous, install a SYCL compiler in your machine:

And if you are using windows you shouldn’t!
What do I run?

1. After logged to you the cluster, get the example and the presentation:
   1. `git clone https://github.com/alcf-perfengr/sycltrain`
   2. `cd sycltrain/presentation/2020_07_30_ATPESC`

2. Get a node
   1. `./fetch_a_node.devcloud.sh`
   2. `# ./fetch_a_node.cooley.sh`

3. Source the correct environment
   1. `cd sycltrain/presentation/2020_07_30_ATPESC`
   2. `source env.devcloud.rc`
   3. `#source env.cooley.rc`

4. Compile and run examples
   1. `cd 9_sycl_of_hell`
   2. `make run_0_tiny_sycl_info`
1. Examples are available in the _9_sycl_of_hell_ folder
2. Exercises are available in the _exercise_ folder
   2.1 We will do the One Atom together \(^{12}\)
   2.2 Harder problem will be covered at the end if we have time\(^{13}\)

\(^{12}\) != "watching me"

\(^{13}\) But you are encouraged to do them by yourself... And send me the solution!
1. Examples are available in the 9_sycl_of_hell folder
2. Exercises are available in the exercise folder
   2.1 We will do the One Atom together \textsuperscript{14}
   2.2 Harder problem will be covered at the end if we have time\textsuperscript{15}

\textsuperscript{14} \texttt{!="watching me"}
\textsuperscript{15}But you are encouraged to do them by yourself, and send me the solution!
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


Examples

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

Documentations

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