CUDA to SYCL Migration

Rakshith Krishnappa (rakshith.krishnappa@intel.com)

Workshop Agenda

February 15th	Introduction to Using the SYCLomatic Tool and Compiling/Executing SYCL code on Intel Dev Cloud
March 15th	Migrating more complex CUDA source with the SYCLomatic Tool
April 12th	Mini Hackathon and Q&A, Migrating your CUDA Code to SYCL - tips, tricks, and limitations, Migrating CUDA code with libraries.

Session #1 - 02/15/2023, 1:30 – 3:30PM CT

- **Introduction to Using the SYCLomatic Tool and Compiling/Executing SYCL code on Intel Dev Cloud**
  - Installing SYCLomatic tool
  - Understand SYCLomatic tool usage and command line options
  - Migrate a simple CUDA example with just one source file to SYCL
  - Migrate a CUDA example with multiple CUDA source files to SYCL
- In this session we will mainly try to understand how memory allocation and memory copy is accomplished in CUDA versus SYCL, we will also look at how a kernel is offloaded to run on GPU in CUDA versus SYCL.

Session #2 - 03/15/2023, 1:30 – 3:30PM CT

- **Migrating more complex CUDA source with the SYCLomatic Tool**
  - Migrate a CUDA example with multiple CUDA source files to SYCL
  - Optimize Kernel code with SYCL features.
- In this session we will understand how CUDA features like Local Memory, Cooperative groups, warp primitives and atomic operations are migrated to SYCL, we will inspect the CUDA and SYCL source and understand how migration was accomplished using SYCLomatic tool. We will also try to manually optimize the migrated SYCL code for performance using SYCL features.

Session #3 - 04/12/2023, 1:30 – 3:30PM CT

- **Mini Hackathon: Migrating your CUDA Code to SYCL - tips, tricks, and limitations**
  - This session will be a mini hackathon where you can bring your own CUDA source and try to migrate to SYCL, Intel experts will help and answer any questions you may have about the migration process.
  - We will also give an overview of how migration is accomplished when CUDA source use a library like cuBLAS or cuFFT, we will show case other CUDA to SYCL migration projects that are completed and can be used as reference. We will also learn about the current limitation of the SYCLomatic tool, we will learn about some tips and tricks when migrating CUDA to SYCL using SYCLomatic tool.

Pre-requisites

- **CUDA development machine:** Have system ready with CUDA SDK installed, you should be able to compile/run a simple CUDA sample code.

- Sign up for **Intel Developer Cloud** account at devcloud.intel.com/oneapi

Workshop Overview

- These sessions involve **2 steps:**
  1. Migrating the CUDA source on CUDA development machine
  2. Executing migrated SYCL source on Intel CPUs/GPUs on Intel Developer Cloud

- The audience is expected to have a **CUDA development machine** ready for this workshop, we will install SYCLomatic tool on the CUDA development and then migrate the CUDA source to SYCL.
- Once the code migration is complete, we will transfer the migrated SYCL source to **Intel Developer Cloud** to compile, execute and optimize on Intel CPUs/GPUs.

If you do not have a CUDA development machine available, you can just watch the demonstration of step one, CUDA to SYCL migration and then do the step two on Intel Developer Cloud.

Notices & Disclaimers

Reference Material

2023/02/15

Install SYCLomatic Tool

- Go to https://github.com/oneapi-src/SYCLomatic/releases
  - Under Assets
  - Copy web link to linux_release.tgz
- On you CUDA Development machine:
  - In home directory or anywhere: mkdir syclomatic; cd syclomatic
  - wget <link to linux_release.tgz>
  - tar –xvf linux_release.tgz
  - export PATH="/home/$USER/syclomatic/bin:$PATH"
  - c2s --version

SYCLomatic Tool Usage (c2s --help)

- Migrate a single CUDA source file:
  - `c2s test.cu`
- Migrate a single CUDA source to a specific directory name
  - `c2s test.cu --out-root sycl_code`
- Migrate a single CUDA source with source root tree
  - `c2s test.cu --in-root ../`
- Migrate a single CUDA source with custom CUDA installation
  - `c2s test.cu --cuda-include-path /tmp/cuda/include`
- Migrate a CUDA project with makefile:
  - `intercept-build make`
  - `c2s -p compile_command.json`

Compiling SYCL code for Intel and Nvidia targets

- Install oneAPI C++/DPC++ Compiler or Intel oneAPI Base Toolkit
- Install CUDA Plugin for oneAPI from CodePlay
- Link to Installation Instruction

- Compile SYCL for Intel CPUs/GPUs
  - `icpx –fsycl test.cpp`
- Compile SYCL for Nvidia GPUs
  - `clang++ -fsycl -fsycl-targets=nvptx64-nvidia-cuda test.cpp`

Learn Basics of SYCL Programming

- SYCL 2020 Specification
- Data Parallel C++ Book
- Guided learning path with code samples (Jupyter Notebooks):
  - SYCL Essentials
  - SYCL Performance Portability
- C++ SYCL code samples

Optimizing SYCL code for Intel GPUs

- Refer to Intel GPU Optimization Guide
  - Detailed guide explaining how to optimize SYCL code:
    - Thread Mapping and GPU Occupancy Calculation.
    - Memory allocation and transfer optimization when using Buffers or Unified Shared memory.
    - Kernel code optimization – Local memory, Sub-Groups, Atomics, Reduction and more.
    - Using libraries for offload
    - Debugging and Profiling
- Link to Intel GPU Optimization Guide