Parallel Computer Systems, Fall 2022
Instructions for Databar Exercise 6:
OpenCL on GPUs

October 25, 2022

1 Introduction

You will this week explore OpenCL on GPUs.

NB: You share the GPUs with all other students and there are only a few GPUs available. This means that some times there will be conflicts and your programs will not run as the GPUs are occupied. You must log out from the GPU node when you are not running programs on it. Otherwise other students might not be able to run. Be patient!

You report on this exercise with up to a page in the second report.

Read through this document in entirely before starting working on the exercises!

2 Reference material

Before carrying out the exercise study `https://github.com/HandsOnOpenCL/Lecture-Slides/releases/download/v1.2/KITE-OpenCL-course.pptx`.

3 Learning objectives

During this assignment you will be working towards the following learning objectives:

- Write small parallel programs using industry standard parallel programming systems.
- Outline the principles for common parallel programming models.
- Use OpenCL to write parallel GPGPU programs.

4 Setting up the system

You will have to use the HPC machines for these exercises just like the previous exercises.

5 Working on the exercises

Challenge yourself throughout the exercises. Try your best, and then some, and fully solve the problems and answer the questions before continuing!

Download and extract the archive `Exercises.tar.gz` from DTU Learn. You will need these files for the exercises.

We will be using parts of the exercises from: `https://handsonopencl.github.io/`. The code in the archive has been adapted to run on the HPC system.

6 Exercises

First, setup the environment to run OpenCL on GPUs using the following commands:

```
voltash
module add cuda/11.1
```

Run these commands to set up the environment when you want to compile and run OpenCL programs on the GPUs.

To run on CPUs only do:

```
voltash
```

Adding the Cuda module will disable CPU support. To have CPU support in OpenCL only run `voltash`.

You must start a new terminal then setup the environment each time you compile and run programs. Once you have run your programs copy the output to an editor and immediately afterwards close the terminal. This way you minimize the time you use the GPU node and will allow others to run.

Sometimes when running programs, there will be a warning about version numbers. Ignore this warning.

However, the commands may complain indicating that GPU nodes are not available. In that case, you should wait a few minutes then rerun the commands. If the problem persist reach out to course staff. We only have a few GPU nodes to share between all students!

Setup the environment to run OpenCL on CPUs using the following commands:

```
source /zhome/bb/5/41921/Public/OpenCL/source.me
export LD_LIBRARY_PATH=/zhome/bb/5/41921/Public/OpenCL/AMDAPPSDK-3.0/lib/x86_64
```

Run these commands in every terminal window you open, to setup the required environment variables.

The following exercises use the files in `Exercises.tar.gz`. To work on Exercise0X go to the directory `Exercise0X/C`. The C++ and Python versions of the exercises have not been adapted to work on the HPC systems.

You may get a compiler warning about a deprecated function (`clCreateCommandQueue`) when compiling the program. Ignore this warning.

6.1 Exercise01

In this exercise you will work with a program that examines the available OpenCL accelerators.

- Read and understand the code in `DeviceInfo.c`.
  - What does the program do?
  - Identify the OpenCL API calls. (The functions starting with `cl`).
  - Find the documentation for each call online. What does it do? Which arguments does it take?
- Compile the program using the makefile.
- Run the program on a CPU and examine the output.
  - What does it mean?
  - Is this what you expected? If not, go back and reexamine the code.

We will now run on GPUs. For that to happen you need to update the `Makefile` which is used for building the program and rebuild the program with `make clean` and then `make`. Before running `make` remove the following lines from the `Makefile`:

```
CL_DIR = /zhome/bb/5/41921/Public/OpenCL/AMDAPPSDK-3.0/include
LIB_DIR = /zhome/bb/5/41921/Public/OpenCL/AMDAPPSDK-3.0/lib/x86_64
```

Run the program and examine the output.

- Run the program and examine the output.
- Is this what you expected? If not, go back and reexamine the code and the commands you executed to run the program.

6.2 Exercise02

In this exercise you will work with an OpenCL kernel that add vectors.

- Read and understand the code in `vadd_c.c`.
  - What does the program do?
  - Where is the kernel program defined?
  - How does it work?
  - Identify the OpenCL API calls. (The functions starting with `cl`).
  - Find the documentation for each call online. What does it do? Which arguments does it take?
- Compile the program using the makefile.
- Run the program on a CPU and examine the output.
  - Is this what you expected? If not, go back and reexamine the code.

6.3 Exercise05

In this exercise you will work with the code of the accelerator kernel. Your task is to experiment with the existing kernel in `Exercises05/C/vadd_c.c`.

- Change the code to run on a CPU. Compile and run the program.
  - Examine the output.
  - Did you get the expected output? If not, go back and fix your code.

Rerun on the GPU. Compare the performance to that of the CPU.

6.4 Calc revisited

We now revisit calc and use OpenCL.

Your task is to:

1. write an OpenCL version of the `calc.c` program
2. measure the performance of your OpenCL implementation on the GPU and compare it to your OpenMP and MPI implementations.

7 Reporting

You report on this exercise with up to a page in the second report. For the calc part of the exercise provide the source code and analyze the performance of the program. For example, how can you measure efficiency and speedup?

For the rest of the exercises, only summarize your experiences.

8 License