|  |  |
| --- | --- |
| A picture of a winding road and trees  PDC – Assignment # 03  OPEN-CL | Abstract  This Assignment Focusses on the usage of Open CL to execute Code on CPUs and GPU. Sequential codes have also been written in order to compare and visualize the execution times of same code with different modes.  Muhammad Huzaifa  20I-0604 |

Contents

[OpenCL-Kernel-Execution 1](#_Toc150634809)

[Setting up OpenCL on Docker 1](#_Toc150634810)

[Task # 01 Matrix Multiplication Using CPU 1](#_Toc150634811)

[Graphic Visualization: 6](#_Toc150634812)

[Task # 02 Merge Sort Using CPU 7](#_Toc150634813)

[Visualizing the Results 8](#_Toc150634814)

# OpenCL-Kernel-Execution

This repository contains multiple codes executing using OpenCl on CPU & GPU. Sequential Codes for each program executed on CPU and GPU has also been written to compare the execution time of all from the same problem category.

# Setting up OpenCL on Docker

- docker-compose up -d master --build  
- docker exec -it dockercl-master-1 bash

* or just run script.bat in terminal and get started

After getting a nice bash shell to the container traverse into the folder you need to check for executon and follow the following commands.

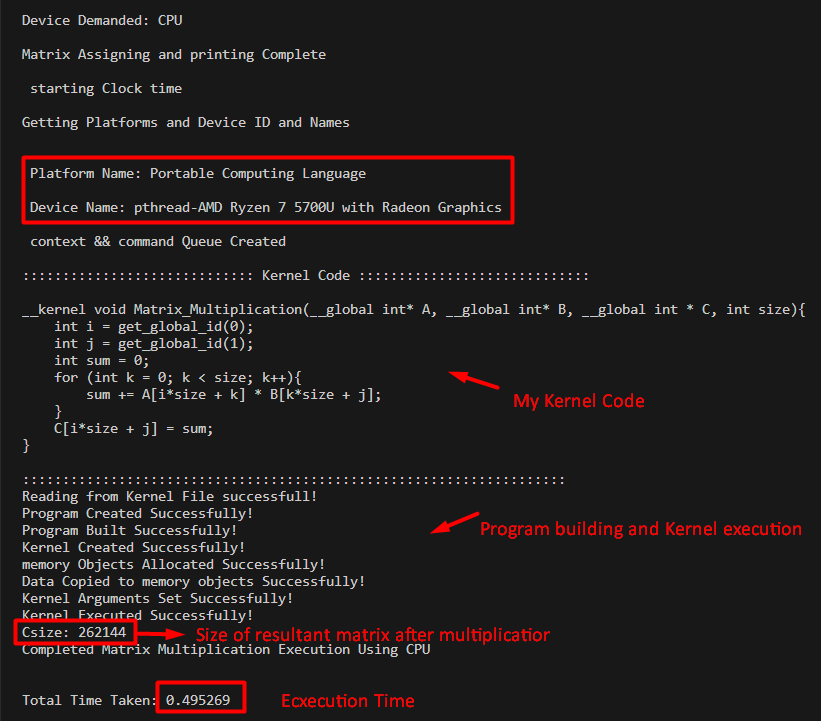
## Task # 01 Matrix Multiplication Using CPU

Assuming Matrix Multiplication is the desired folder

cd Matrix Multiplication  
gcc Matrix\_Mul\_CPU.c -o matmul -lOenCL  
./matmul <mode of execution> i.e., ./matmul CPU

After running the executable via accurate command line arguments you will get something like this:

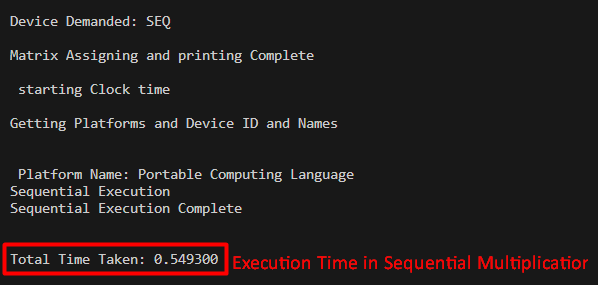
#### CPU based Kernel Execution



As you can see in the screenshot above, the name of the platform and the name of the CPU device used for the execution of kernel code responsible for the multiplication of 2 matrices.

#### Sequential Code Execution

Similarly, Let’s try and run treh sequential code and check for its execution time.

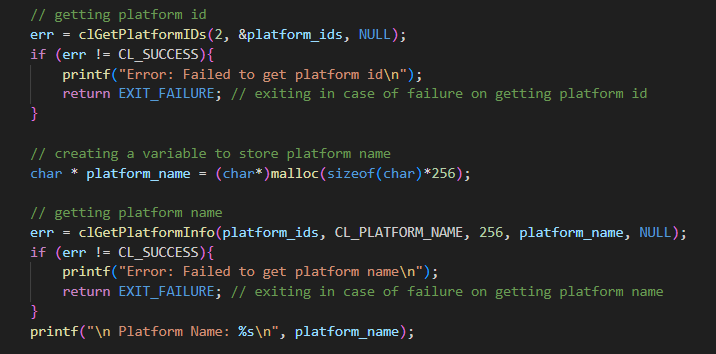


As you would have already notice that the execution time taken by a sequential code is rather high then the execution time taken by the CPU kernel code.

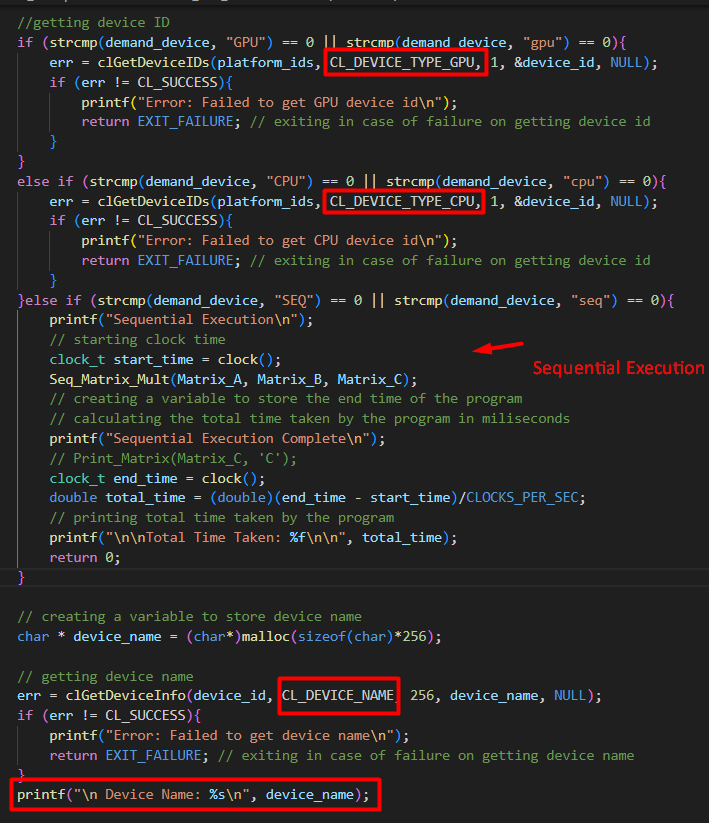
#### Implementation

Foe the implementation of the CPU based kernel execution, thorough steps have been taken to ensure proper and smooth execution. Here are the highlights of the code:

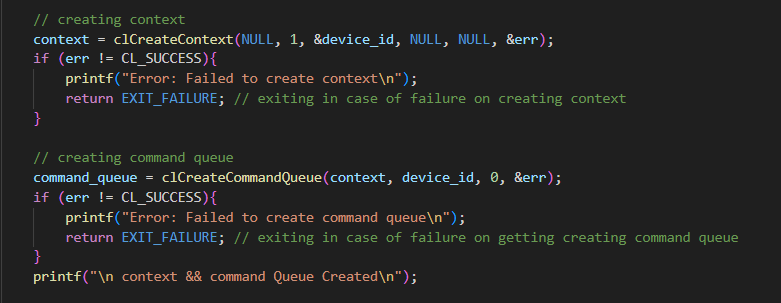
* Getting Platform ID and Platform name



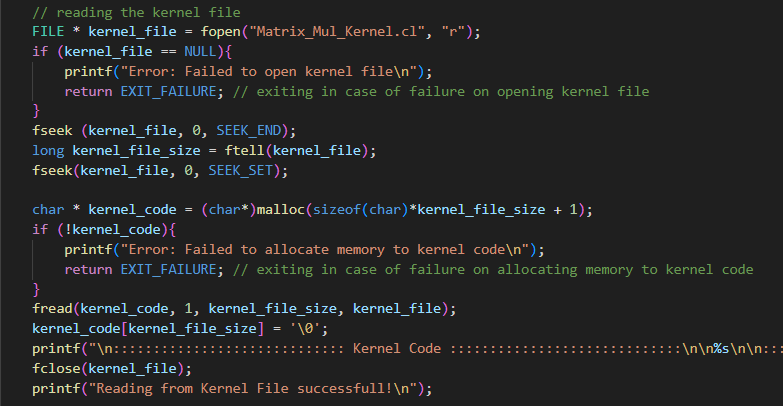
* Setting Device Type, (In case of Sequential) & Getting Device Name



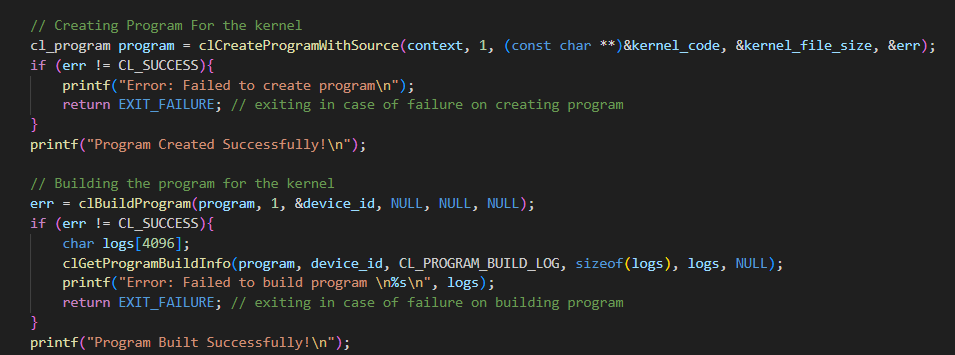
* Creating Context and Command Queues



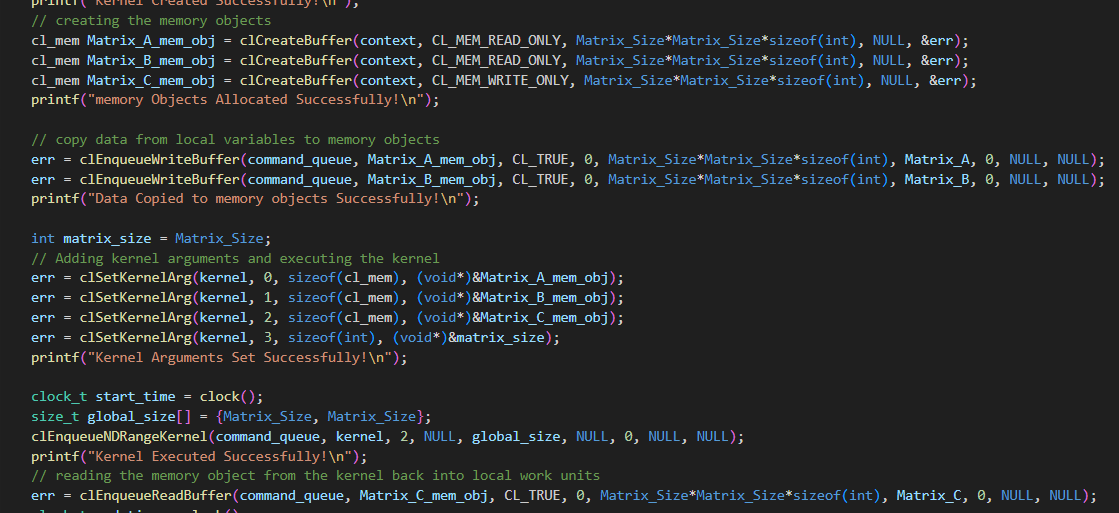
* Reading Kernel Code from Kernel File This has two ways, you can either use a const char \* string to store the code that kernel is going to execute and pass this to create kernel and program. Or you can create a saperate kernel file and read it to pass kernel code to create kernel and program. In my opinion the later is a better option.



* Create & Build Program



* Finally Load Memory arguments, Execute Kernel and Store the Returned global memory object for resultant into your work unit’s local memory object.



###### Note

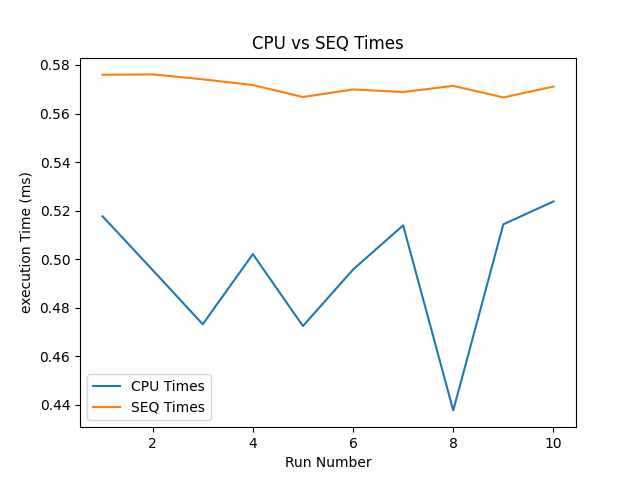
* you can uncomment the matrix printing codes on line # 82, 83, 272 to print the input and resultant matrices to observe the output better.

### Graphic Visualization:

How to Run:

cd Matrix Multiplication  
pip3 install -r requirements.txt (U can search for system specific issues on your own)  
python3 graph.py

To visualize the difference in execution times of both CPU and SEQ matrix multiplication I executed the Matric\_Mul\_CPU.c file with both CPU and SEQ modes using python’s subprocess. Then I extracted the execution times from teh output of both mode executions and stored in 2 different Arrays. Then I used matplotlib to plot the graph using both mode’s execution time and here is teh final result.

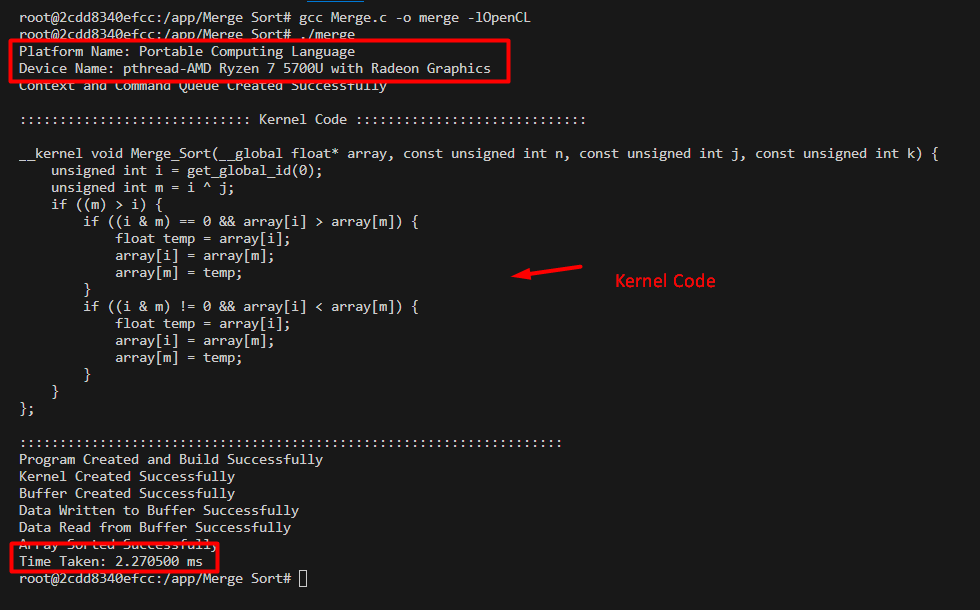


## Task # 02 Merge Sort Using CPU

cd Merge Sort  
gcc Merge.c -o merge -lOenCL  
./merge  
gcc SEQ\_Merge\_Sort.c -o sort  
./sort

After running the executable you will get something like this:

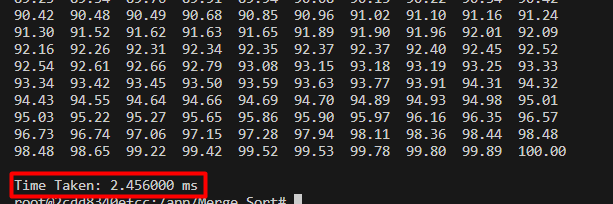
#### CPU based Kernel Execution



As you can see in the screenshot above, the name of the platform and the name of the CPU device used for the execution of kernel code responsible for the multiplication of 2 matrices.

#### Sequential Code Execution

Similarly, Let’s try and run treh sequential code and check for its execution time.



As you would have already notice that the execution time taken by a sequential code is rather high then the execution time taken by the CPU kernel code.

## Visualizing the Results

Following are teh visualized results of the execution times in CPU and Sequential code executions:

