CS3210 – Parallel Computing (AY 2017/2018 Sem 1)

Assignment 3

CUDA Matrix Multiplication (5 marks)

Individual Submission due on **Fri, 17 Nov, 2pm**

You are given a GPU-enabled matrix multiplication program written in CUDA (Compute Unified Device Architecture). Using your knowledge of memory access and the GPU organization, you will work towards reducing the execution time.

Before working for this assignment, you should revise CUDA Programming from Lecture 11 and Lab 6.

Performance Improvements on Jetson TK1 GPU

You need to use the **Jetson TK1** node for this question. You can ssh into the node from either ***node1*** or ***node2***.

Compilation and Execution

Get the "**`mm-seq.c`**" and "**`mm-cuda.cu`**" from the web:

```
$wget www.comp.nus.edu.sg/~ccris/cs3210/lab6/mm-seq.c
$wget www.comp.nus.edu.sg/~ccris/cs3210/lab6/mm-cuda.cu
```

Compile the sequential version:

```
$ gcc -O3 mm-seq.c -o mm-seq -lrt
```

Run the sequential version with any matrix size, e.g. 512

```
$ ./mm-seq 512
```

Take note of the sequential timing $T_{seq}(512)$.

Compile the GPU (CUDA) version:

```
$ nvcc –arch=sm_32 mm-cuda.cu –o mm-cuda –lcuda -lcudart
```

- `-arch=sm_32` – specify for which GPU architecture to generate the code; in our case, Jetson TK1 has a Kepler GPU with compute capability 3.2.
- `-lcuda –lcudart` – link with CUDA libraries.

You may use other compilation flags, as you see fit.

Run the cuda version with the same matrix size, e.g. 512

```
$ ./mm-cuda 512
```

Take note of the parallel execution time $T_{par}(512)$.

Your Task

- Optional: optimize the cuda version of the matrix multiplication for better execution time without changing the memory management (i.e. continue to use device global memory). Name your program **mm-optimized.cu**
- Optimize the cuda version for better execution time by using device shared memory instead of global memory. Name your programs **mm-sm.cu** and **mm-banks.cu**
- Derive the speedup achieved and explain your optimizations. **Marks will not be awarded if the performance results obtained are not explained**.

Note:

- Use a matrix size of **at least 512 x 512.**

Evaluation

Marks:

- Up to 3 marks are awarded if you reduce the execution time by using device shared memory. The program should be named **mm-sm.cu**
- Up to 2 marks are awarded if you reduce the execution time by considering the memory banks. The program should be named **mm-sm-banks.cu**

Bonus:

- 1 bonus mark for the two best submissions in terms of performance subject to reducing the execution time.

Deliverables and Submission

Prepare a **zip file** using your **student id (matric no), e.g.** A01234567X.zip, which contains:

`mm-sm.cu`
`mm-banks.cu`
`mm-optimized.cu (optional)`
	Your optimized versions with appropriate comments, modularization and indentation. Please **remove all debug messages**.

`makefile / readme`	For compiling your cuda programs.
`assignment3_report.pdf`	Contains: 1. Explain your optimizations. 2. Speedup achieved. **Include details on how to reproduce your result, e.g. matrix size, execution time measurement etc.** 3. Compare results of the sequential, cuda and optimized implementations. 4. Any special consideration or implementation detail that you consider non-trivial.

The zip archive must be uploaded to IVLE in the workbin folder "Assignment2" by **Fri, 17 Nov, 2pm. Penalty of 10% per day for late submissions will be applied.**

References

1. CUDA C Best Practices Guide: http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html

2. http://cseweb.ucsd.edu/classes/wi12/cse260-a/Lectures/Lec09.pdf