Mini-Project 1 – Multicore Programming

Due: Monday, March 28th at 11:59PM EST

The goal of this project is to use your understanding of parallel computing resources in a manycore processor to optimize two fully functional applications. The applications are Matrix Multiple and K-Means Clustering.

For a functional description of the applications, please refer to:

- http://en.wikipedia.org/wiki/Matrix_Multiplication
- http://en.wikipedia.org/wiki/K-Means

The code optimization techniques you may want to consider are explained in Lectures 7, 8 and 9.

Grading Criteria

- 30% - Correctness
  - `matrix_mul/cuda/matrix_mul.cu` - The provided code is implemented for **power of 2** input matrix sizes only. Create a version that works for any matrix up to 2048x2048 in size.
  - `kmeans/cuda_kmeans.cu` - The provided code does not work for two of the provided data sets (`kmeans03.dat` and `kmeans04.dat`). Create a version of kmeans that works for all four data sets. *Hint: check the "compute_delta" function and arguments*

- 30% - Performance
  - `matrix_mul/cuda/matrix_mul.cu` - Achieve an average of **at least 200GFLOPs** of throughput across the 10 testcases in `matrix_mul_03.dat`
  - `kmeans/omp_kmeans.cu` - Achieve **at least a 2x speed up** compared to the provided code (SUM of all tests)

- 30% - Write up - For each performance optimization explored, describe clearly:
  - How the speed up works
  - What is the expected speed up?
  - What is the observed speed up?
  - An explanation of any difference between the expected and observed speed ups

- 10% - Code quality - Good coding practices and well commented code

Guidelines for the write up:

Minimum of one 8.5x11 page write-up for each optimization. The write up should include:

- Optimization goal:
  - Hardware resources being optimized towards? (GPU memory? Shared memory?)
  - What is the specification of the hardware you are optimizing for?

- Optimization process:
  - Data considerations
  - Parallelization considerations

- Optimization results:
  - Performance before optimization
  - Performance after optimization

The three teams with the fastest implementations will present the techniques they attempted in a 10-minute presentation during the project review session.

Mini-Project 1 - Setup

Step 1: Download the initial version of the code

```
$ cd ~/
$ cp /afs/andrew.cmu.edu/course/18/646/MP2/18646_MP2.tar.gz ~/
$ tar xzvf 18646_MP2.tar.gz
$ tree 18646_MP2
18646_MP2/
├── kmeans
│   ├── cuda_io.cu
│   ├── cuda_kmeans.cu        <<== To optimize (K-Means)
│   ├── cuda_main.cu
│   ├── cuda_wtime.cu
│   ├── kmeans.h
│   └── Makefile
└── matrix_mul
    ├── cuda
    │   ├── Makefile
    │   ├── matrix_mul.cu      <<== To optimize (Matrix Multiply)
    │   ├── matrix_mul.h
    │   └── tests.cpp
    ├── matrix_mul_03.dat
    └── tests
        └── testutil.h

4 directories, 12 files
```

Step 2: Compile the code, by running "`make`" in the appropriate project directory.

Set up the CUDA Environment as described in Homework 2 (Task 2)

Compile the provided **matrix_multiply** code:

```
$ cd ~/18646_MP2/matrix_mul/cuda
$ make
$ ./matrix_mul -i ../matrix_mul_03.dat -o
 Test Case 1     0.00644 Gflop/s
 Test Case 2     0.01286 Gflop/s
 Test Case 3     0.41478 Gflop/s
 ...
```

Compile the provided **K-Means** code:

```
$ cd ~/18646_MP2/kmeans
$ make
$ ./cuda_main -i ~/18646_MP1/data/kmeans02.dat -n 64 -o
 Writing coordinates of K=64 cluster centers to file ...
 Writing membership of N=7089 data objects to file ...

 Input file:    ~/18646_MP1/data/kmeans02.dat
 numObjs        = 7089
 numCoords      = 4
 numClusters    = 64
 threshold      = 0.0010
 Loop iterations    = 73
 I/O time           = 0.0113 sec
 Computation timing = 0.6274 sec
```

Step 3: Optimize your code

- For the project "**matrix_mul**", please apply your optimization only to the file `matrix_mul/cuda/matrix_mul.cu`.
- For the project "**kmeans**", only make changes to the file `kmeans/cuda_kmeans.cu`.

Note: DO NOT change the function interface. Any changes in the interface could result in your work not working in our test infrastructure and you will receive no credit.

Step 4: Submit your optimized code (`matrix_mul.cu` and `cuda_kmeans.cu`) and project write up to gradescope

Submit your optimized version of `matrix_mul.cu` to the Matrix Multiply programming assignment on gradescope: https://www.gradescope.com/courses/357643/assignments/1881328

Submit your optimized version of `cuda_kmeans.cu` to the K-Means programming assignment on gradescope: https://www.gradescope.com/courses/357643/assignments/1881329

Submit your team project writeup to:
https://www.gradescope.com/courses/357643/assignments/1881332