CS 4380 / CS 5351
Parallel Programming / Parallel Processing
Fall 2016

Project 5

Due: *Thursday, November 17 at 3:30 PM*

All project files including the report have to be submitted using TRACS. Please follow the instructions at http://tracsfacts.its.txstate.edu/trainingvideos/submitassignment/submitassignment.htm. Note that files are only submitted if TRACS indicates a successful submission. See below for what files to submit. The project report has to list and discuss the results of your measurements and provide the answers to the questions. The answer to each question is limited to 50 words. This project has to be done *individually*. You have to be able to explain all of the code that you submit.

5.1 Fractal [45u/40g points]

Parallelize the fractal computation code using CUDA. To obtain enough parallelism, simultaneously parallelize the *for-frame*, *for-row*, and *for-col* loops. In other words, all three of these loops are no longer needed and should be removed as you will have to launch a thread for each iteration of this loop nest (i.e., for each pixel), meaning that only the loop-body code remains. Follow these steps when implementing your code:

1. Base your code on the provided skeleton at /home1/00976/burtsche/Parallel/fractal.cu.
2. Replace all the "…" by inserting the missing code.
3. Remove the loop-carried dependency by computing *delta* as a function of frame.
4. Launch the kernel with 512 threads per block and just enough blocks to create enough total threads. Use the formula from the slides to round up to the right number of blocks.
5. Verify, on small inputs, that the correct fractal is computed.
6. Write a second version of the code that is identical except all "double" types are replaced by "float" (the "runtime" variable can remain a double). Name the new file fractal_float.cu.

On the machines in the parallel lab, compile the CUDA code as follows:

```
nvcc -O3 -arch=sm_20 fractal.cu -o fractal_cuda
nvcc -O3 -arch=sm_20 fractal_float.cu -o fractal_float
```

On Stampede, compile the CUDA code as follows:

```
module load cuda
nvcc -O3 -arch=sm_35 fractal.cu -o fractal_cuda
nvcc -O3 -arch=sm_35 fractal_float.cu -o fractal_float
```

Submit both source-code files on TRACS. Run the code using the provided submission script at /home1/00976/burtsche/Parallel/fractal_cuda.sub. Do not modify this script. Present the runtime for each problem size and executable in a table. Explain and discuss the results. For the 500w 60f input, how does the performance compare to the OpenMP runtime of 0.953 seconds? When you

execute the command "./fractal_cuda 250 30" on one of the Stampede login nodes (after running "module load cuda"), it doesn't work. Why not?

5.2 TSP [55u/60g points]

Parallelize the TSP code using CUDA. To obtain enough parallelism, simultaneously parallelize the *for-i* and *for-j* loops. In other words, both of these loops are no longer needed and should be removed as you will be launching a separate thread for each edge pair. Follow these steps when implementing your code:

1. Base your code on the provided skeleton at /home1/00976/burtsche/Parallel/tsp.cu.
2. Replace all the "…" by inserting the missing code.
3. Verify that the correct tour is computed.

This code cannot be compiled on the machines in the parallel lab as their GPUs are too old. On Stampede, compile the CUDA code as follows:

```
module load cuda
nvcc -O3 -arch=sm_35 tsp.cu -o tsp_cuda
```

Submit the source code on TRACS. Run the code using the provided submission script at /home1/00976/burtsche/Parallel/tsp_cuda.sub. Do not modify this script. Present the best runtime for each problem size in a table. Explain and discuss the results. For the d1291 input, how does the performance compare to the OpenMP MIC runtime of 0.91 seconds? Explain. More than half of the running threads in the BestMove kernel do not get to evaluate any edge pair. Why not?

[***CS 5351 students only***] Why is it necessary to use an atomicCAS in this code to atomically compute a global minimum even though CUDA also has an atomicMin function?

Code Requirements

- Make sure your code is well commented.
- Make sure your code does not produce unwanted output such as debugging messages.
- Make sure your code's runtime does not exceed the specified maximum.
- Make sure your code is correctly indented and uses a consistent coding style.
- Make sure your code does not include unused variables, unreachable code, etc.

Code Submission

- Delete all files that you do not need anymore such as *.o and *.bmp files.
- Make sure your code complies with the above requirements before you submit it.
- Any special instructions or comments to the grader should be included in a "README" file.
- Upload all the files you need to submit onto TRACS. The report has to be in PDF. All other files, including source code, have to be plain text files (e.g., *.cpp files).
- Upload each file separately and do not compress them.
- Do not submit any unnecessary files (e.g., provided or generated files).

You can submit your file(s) as many times as you want before the deadline. Only the last submission will be graded. Be sure to submit at least once before the deadline.

November 10, 2016