CS 421: OutLab: Back To Basics: Odd-Even with CUDA

- Released: Sunday March 4 7:30 AM. Due: Thursday March 8 10:00 PM
- Please do not forget the honor code.

Overview

This is an individual assignment. Any discussions should be constrained to public Piazza postings. You are not to discuss with ANY other person (same class, same batch, different batch, unknown person, ...). The only exception is your professor.

Important Note: Since you know the algorithm inside out, you are not expected to take code from the Internet. Any links considered must be cited as usual.

Task A

We are going to work with odd even sort again, this time with CUDA Please note the following constraints

1. The program should adhere to the general style of openMP odd even sort that has been given to you earlier. For example, you should be ready to either generate the test input, or read from standard input.
2. In particular, you will implement the function `Odd_even` as a CUDA kernel as follows in increasing order of complexity (each of them will result in a file called `ansx.cu`. Each of them will result in elapsed time being printed on the standard output in milliseconds. Round time to the nearest integer. Debug output should go to stdout so that we can compare the answers. Please ask your friends on Piazza on how to do this if this is not clear.
   (a) The number of elements equal to the number of blocks. (Recall that the code fragment provided asks for the number of threads as input. For simplicity the user will simply repeat the number to indicate number of blocks, which is the number of elements.)
   (b) The number of blocks is one (Recall that the code fragment provided asks for the number of threads. For simplicity the user will simply repeat the number to indicate number of threads.)
   (c) No (so-called) shared memory was used in the previous two cases. Now use `__shared__` using one block
   (d) Now, use 128 blocks and 1024 threads per block. Set your data size so that the number of data items is one per thread. You are free to use or not use shared memory. (Recall that the code fragment provided asks for the number of threads. Use the product of the numbers in the input for thread count.)
   (e) Assume 16 blocks and 1024 threads per block. Assume data size is large but will fit in GPU memory. Set chunksize=1024, and let each thread process the data in chunks. You are free to use or not use shared memory but the goal is to get the best possible time, with the restriction that we are going to use odd-even sort everywhere. (Recall that the code fragment provided asks for the number of threads and the size of the input. Use the product of blocks and threads for thread count. Number of elements is similarly computed.)

Task B

This is the reflection essay. Explain your method and difficulty faced in each of the above cases. Also compare your time with the best MPI version and the best openMP version available to you on the web page.

How We will Grade You

- (a) through (d) will be based on correctness
- (e) will be based on correctness and quality (time taken). This is 30%. In particular, we will take the times from the submissions and do some sort of binning (after a linear scale). The best submission will get a bonus.
- The reflection essay carries 20% marks
- Code quality is also important. Make sure to document your code and always use cuda error checking (see this link). You don't have to worry about the code being general to handle arbitrary (but be nice to use macros instead of placing hard coded numbers inside code).

Submission

You can ask for hints from the teaching team if you get stuck. As always, stay tuned on Piazza for clarifications Your group number is the last two digits of your roll number with some exceptions.

1. Do include a readme.txt (telling me whatever you want to tell me). Do include group members if any (name, roll number), group number, honour code, citations, etc.
2. As mentioned, you should periodically commit to GitHub.
3. As usual, submit to Moodle. Grading is based on Moodle submission. The folder and its compressed version should be similarly named. (For example: the folder is `lab02_group07_outlab` and the related `tar.gz` is `lab02_group07_outlab.tar.gz`
4. Your submission should look something like

```
cudaOutlab_groupXY
├── ansA.cu
├── ansB.cu
├── ansC.cu .2 ansD.cu .2 ansE.cu .2 reflection.txt
├── makefile
└── readme.txt
```