Name : Sweta Rout
Unity ID : srout
Student ID : 200206715

Performance comparison: a table lists execution time, speedup (using mode 0 as the baseline) of GPU sort the excludes the data initialization and result verification.

I chose to implement merge sort.

Mode	Execution Time	Ratio to the base(mode 0)
0	27046002	1
3	21813746	1.23

What's the complexity of your algorithm?

Complexity of merge sort in GPU is O(log n * n). This is because GPU runs the sorting algorithm for n number of times in parallel by dividing the input data set into those many chunks.

Is the performance improvement as good as you expected? Why or why not?

Ideally, the algorithm should provide much better results as the execution is done in parallel. But, in order to achieve better results or speedup, my code has to make best utilization of the memory units and accordingly select the number of threads and blocks. Moreover, the threads are synchronized so that they all complete before the next step. This hampers the speedup.

Comparing your performance with pthread

Name	Execution Time	Ratio to the base(pthread)
Pthread merge sorting	20271504	1
CUDA merge sorting	21813746	0.92

Performance should be better in openCL than compared to pthread. Because for 2 cores, pthread has two threads for each of the core. And, FPGA/openCL can have a large number of work units to work on a large number of elements in a parallel environment. Here, we are getting little slower performance because of overhead.

The list of students sharing the same board with you :

- Sweta Rout (srout@ncsu.edu)
- Cody Nesbitt (cznesbit@ncsu.edu)

References :

https://arxiv.org/ftp/arxiv/papers/1505/1505.07605.pdf

(For more understanding CUDA parallelism and sorting)