***Project Report***

***Submitted by: Vidhyashree Nagabhushana***

***NETID: v\_n63***

**6.1 OpenMP + CUDA Collatz**

**Question 6.1a) The compute times (in seconds) in a bar graph with the CPU workload percentage along the x-axis and the time along the y-axis labelled for each bar showing the compute time with two digits after the decimal point.**

**Question 6.1b) Explain the compute-time behavior, especially why it first drops and then increases when increasing the CPU percentage (and decreasing the GPU percentage).**

As we increase the workload on CPU and reduce the workload on GPU, since the workload increases on CPU and, also CPU has lesser compute nodes than that of GPU, the compute time for the increased workload on CPU increases.

**Question 6.1c) Which CPU percentage results in the highest performance?**

CPU percentage 12 results in the highest performance.

**Question 6.1d) How much faster is the best hybrid execution relative to just using the GPU?**

The best hybrid execution is 11.2% (1.12 times) faster than just using the GPU.

**Question 6.1e) How many OpenMP threads are used in the parallel CPU code section and where is this number specified?**

20 OpenMP threads are used in the parallel CPU code section and this number is specified in the submission file as stated: “export OMP\_NUM\_THREADS=20”

**Question 6.1f) How many CPU sockets and cores do the GPU compute nodes of Lonestar5 have? Is hyperthreading enabled on those cores?**

The GPU compute nodes of Lonestar 5 has one socket and 10 cores. Hyperthreading is enabled with 20 threads per node.

**6.2 OpenMP + CUDA + MPI Collatz**

**Question 6.2a) Present the compute times (in seconds) in a bar graph with the CPU workload percentage along the x-axis and the time along the y-axis. Include a label for each bar showing the compute time with two digits after the decimal point.**

**Question 6.2b) Which CPU percentage results in the highest performance?**

CPU percentage 10 results in the highest performance.

**Question 6.2c) How much faster is the fastest hybrid execution running on 4 compute nodes relative to the fastest hybrid execution running on 1 compute node (Part 1)?**

The fastest hybrid execution running on 4 compute nodes is 73.9% (3.83 times) faster than the fastest hybrid execution running on 1 compute node.

**Question 6.2d) What is a likely problem with the blocked distribution of work across the compute nodes? Explain**

If we distribute the workload in a blocked fashion, this might lead to load imbalance which leads to some threads having to take more workload.

**6.3** **OpenMP + CUDA + MPI Fractal**

**Question 6.3a) Present the compute times (in seconds) in a bar graph with the number of CPU frames along the x-axis and the time along the y-axis. Include a label for each bar showing the compute time with two digits after the decimal point.**

**Question 6.3b) Do these results mean that running no frames on the CPU is always fastest? Explain.**

Yes. In the code, we notice that the computations that take place are ideal for a GPU as they are large computations that the GPU can handle.

**Question 6.3c) For inputs with just a few CPU frames, it is better to parallelize the middle (for-row) loop than the outer (for-frame) loop. Why is that and under what condition?**

We know that smaller inputs have lesser issues of load imbalance. Hence, the middle (for-row) loop has a lesser load imbalance issue when compared to the outer (for-frame) loop. Hence, it is better to parallelize the (for-row) loop.