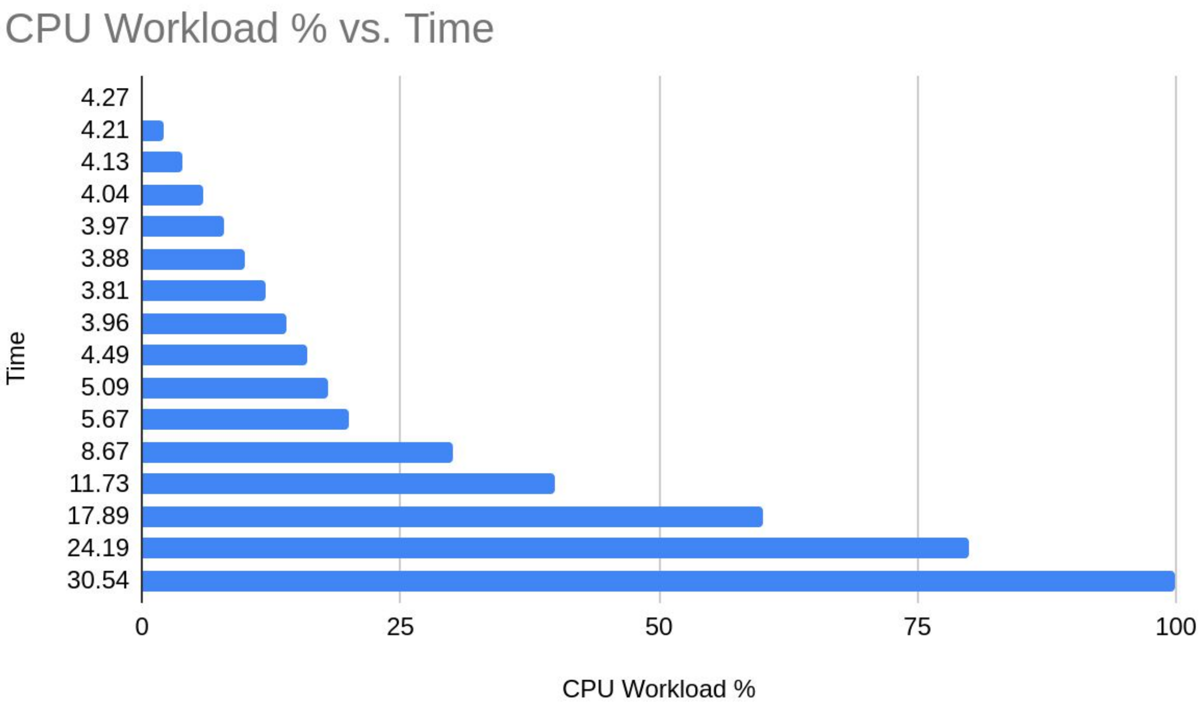
Victor Calixtro
CS 4380 Parallel Programming

6.1 OpenMP + CUDA Collatz

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

When the CPU workload is small, it can handle it. As the CPU workload increases, the GPU workload decreases there isn't enough work for the GPU to do so it isn’t as efficient and the CPU is given a larger percentage of work which is better optimized for the GPU.

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

The cpu percentage with the best time is 12%.

Question 6.1d) How much faster is the best hybrid execution relative to just using the GPU?
4.27/3.81 = 1.12 So twelve percent.

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

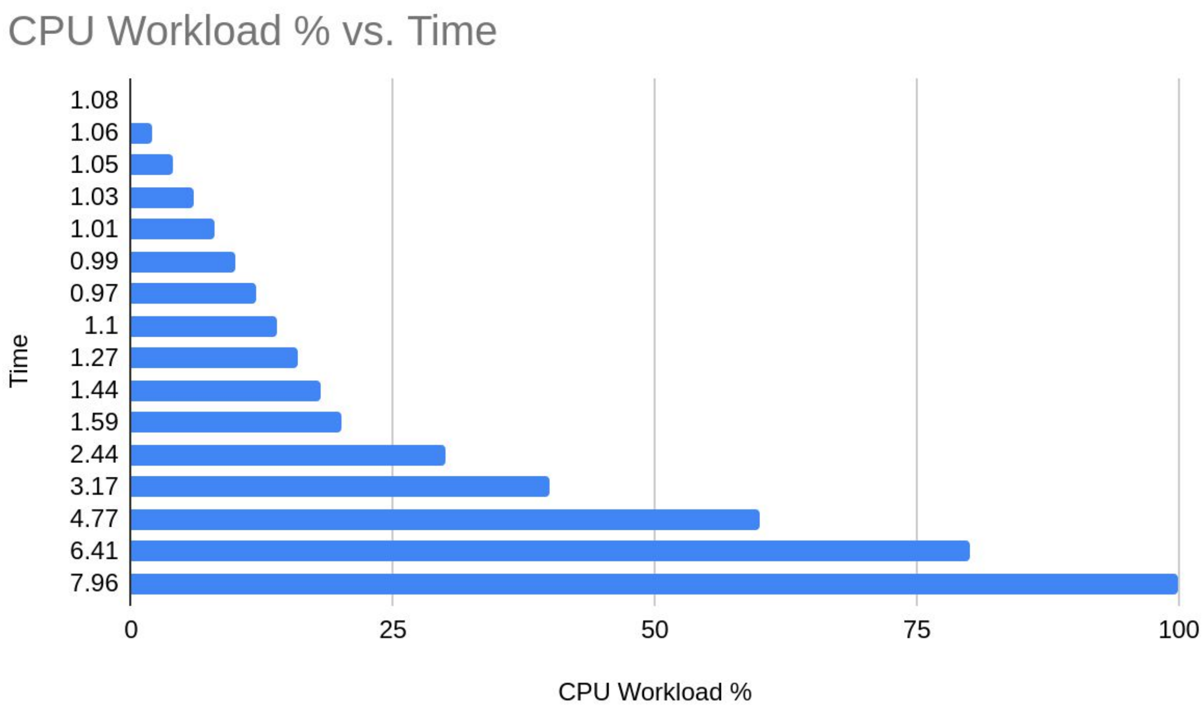
There are 20 threads, its in the sub file.

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

According to the lonestar website its 5 sockets and 10 cores. Hyperthreading is enabled.

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.



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

12% results in 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)?

3.81/0.97 = 3.92 So it is almost 4 times faster.

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

The problem is most likely load balancing, The compute nodes might be getting a number that is not divisible by 4.

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.

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