Brian Rodriguez
U0853593

Assignment 5

2.

I have access to the CHPC machines including the Kingspeak GPU nodes. I have already submitted a ticket to get Caffe2 on the CHPC machines as well. Although, in order to get Caffe2 on the CHPC machines they had to use a 7-month old Docker image. They have left my ticket open in the event that the 7-month old Docker image is far outdated. I still have to run some tests but I think I will need them to use a more recent version of Caffe2. I have met with Martin Cuma to sort out the details of this process and he will be the one that will aid me if I need a new version of Caffe2. We have already seen a more recent Docker Image of Caffe2 in the Caffe2 github repo but until further notice I will continue to use the older version of Caffe2.

3.

a. The basic constructs at the start of the program are essentially sanity checks to tell the threads whether or not they have valid information. If they do not, then the main thread reports this and all of the threads halt and shutdown. The first AllReduce call, ensures that the Mallocs of the child threads executed correctly. However, this is only a sanity check and is not required for the main program. The next Allreduce essentially does the exact same thing as the previous; checks to see if the malloc has failed. The first notable call is Scatter. This call essentially splits the work and sends a portion of the array to all of the worker threads. Scatter uses the calculated local_n to use as a means of dividing up the work. Scatter is called twice so that it can send both local_x and local_y to the worker threads so that they can also do the work. The next call is to Parallel_vector_sum. This call uses the calculated local_n and all of the input and output vectors that need to be worked on. All of the threads will perform work on the arrays including the main thread. After this call, Gather is called to retrieve all of the results from the threads to be stored in a buffer. Then the main thread prints a sampling of the values that were retirved.

b. The execution time of this program did not differ by a huge amount when using a variable amout of threads. The execution times were 19.056394, 18.788561, 18.550825 for 2, 4 and 8 threads respectively. There was a bit of improvement between the varying thread sizes but I believe that the program could be optimized as mentioned in the homework document. This is not an industry grade MPI program but I look forward to see what such a program looks like.