Nate Stewart
Dr. Jeffrey Jones (TuTh 12:45pm – 2:05pm)
04-19-16
Lab 5 – Ping Pong using MPI

Environment:

All test results presented in this report were run on the Oakley cluster using an interactive node with 1 node, 12 ppn, and a walltime of 59 minutes. All MPI compilation occurred using the default version mvapich2/2.1. The program was written using standard C and compiled using the following command: mpicc -O3 -o a.out stewart_nate_lab5.c. The program should be run with the following command: 'mpirun -np 2 a.out'. Unfortunately, the only timing method used was unix time because ctime provided very inconsistent numbers which made the bandwidth calculations difficult.

Results:

Detailed run time and bandwidth information is available in the table below, Figure 1. The program was run five times and the information available in Figure 1 are averages of the five runs. The runtime was recorded in seconds, and the bandwidth in bytes/second.

Message Size	Runtime (seconds)	Bandwidth (bytes/sec)
32	2.687	381195256.283
256	3.670	2227912691.873
512	4.915	3333680805.718
1024	7.597	4314183915.637
2048	15.712	4171319855.383

Figure 1: Runtime metrics for the five message sizes

As the message size increases, the number of bytes transmitted increases. As the number of bytes transmitted increases, the bandwidth also increases (up to a point) because the program is able to send more and more information without impacting the runtime too extensively. This correlation works up to a message size of 2048 where the amount of information to transmit is too much to complete in a reasonable amount of time. This is consistent with the bandwidth calculations where we went from 381195256 bytes/sec with a message size of 32 up to 4314183915 bytes/sec with a message size of 1024 and then back down to 4171319855 bytes/sec with a message size of 2048. With this information, it seems that the BUS is being fully utilized with a message size somewhere between 1024 and 2048.

Conclusion:

Although this was a very simplistic lab, it had a very powerful message: MPI is extremely powerful. In a hierarchical setup where MPI is used to facilitate message passing between two systems, Pthreads or OpenMP can be used to spawn off threads to complete work. It was also interesting to understand just how many bytes can be sent among processes on a single computer via an I/O BUS.