Tyler DeFoor

CS 415

Parallel Computation

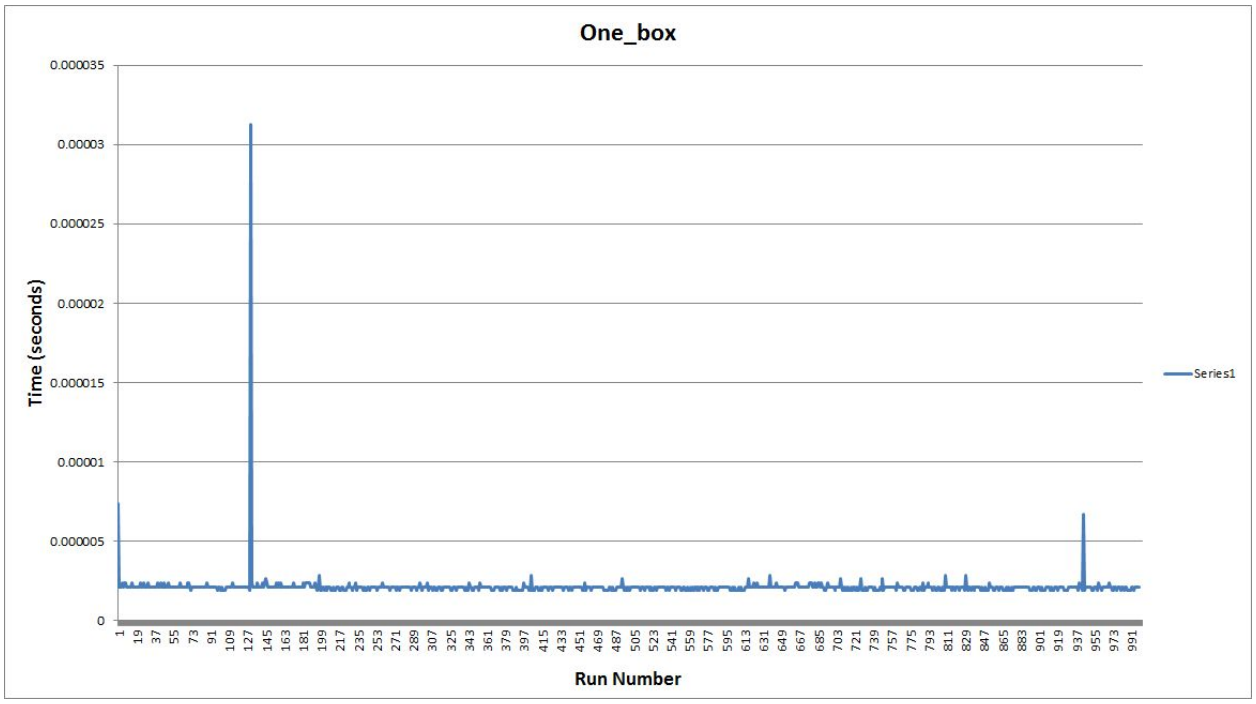
Dr. Harris

PA1 - Ping Pong Report

Part 1: Ping Pong in One Box

The purpose of this part of the assignment is to gauge how long it takes to pass one int between a process and back again in the same node. This means that instead of using network protocols, it would use shared memory. The average time was 2.15E-06 seconds or 2.15 microseconds. This was obtained through using Wtime() before the initial send to start the “timer”, then using Wtime() again after the final receive to end the “timer”.

Overall the graph is fairly consistent. There are some obvious outliers, which are most likely due to network congestion or some hardware fault. This data is the aggregate of 1,000 runs.



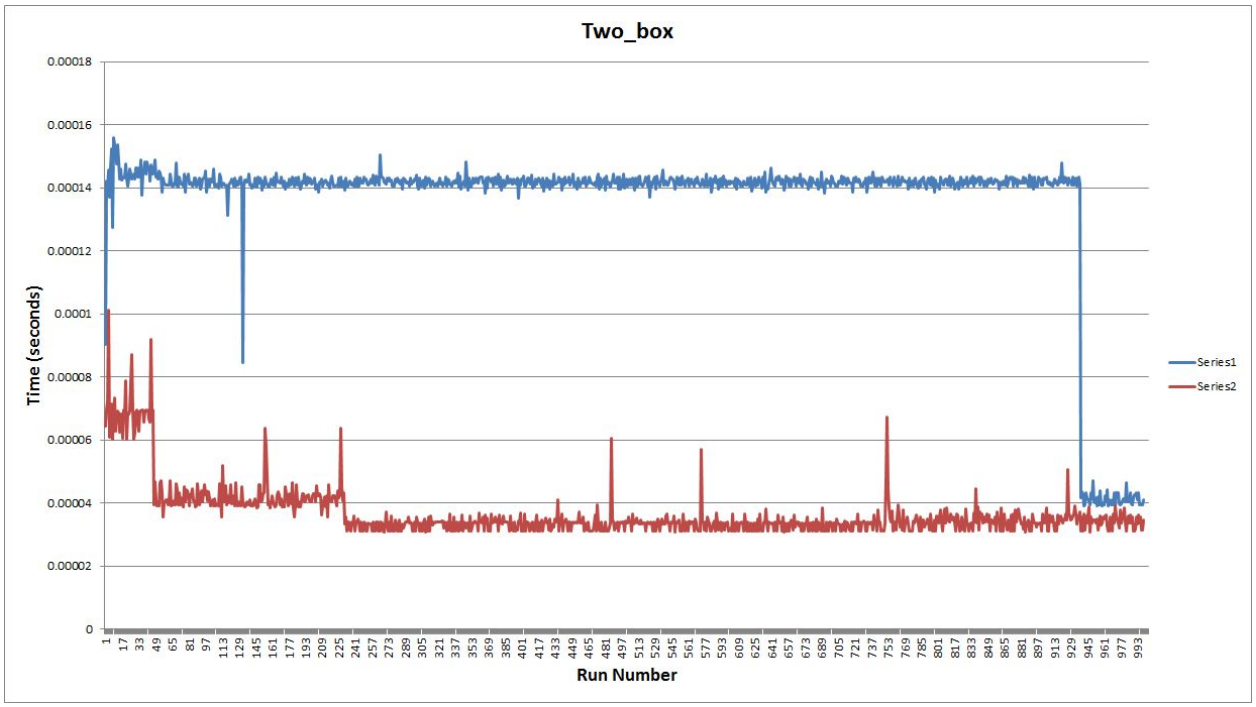
Part 2: Ping Pong in Two Boxes

The purpose of this part of the assignment was to gauge how long it takes to pass an int between two processes on separate nodes. Instead of using shared memory, it would have to go out into the network to send the data.

The average of run one, the blue line, is 2.71E-04 seconds or 271 microseconds. Upon looking at the data, I noticed that the last 100 runs were significantly less than the rest of the data. Thinking this was odd, I ran it again and got run two, the red line. The average for this run is 3.68E-05 seconds or 36.8 microseconds. Run two was much more consistent than run one.

The first run was completed at 6 PM on 2/20/2017. This was a period of time in which there were a lot of jobs running and queued. The second run was at around 2 AM on 2/21/2017, a much quieter time. This should be the cause of the vast time difference between run one and run two, as run one had to deal with other processes on the cluster. For this reason I will take the second run, with an average of 36.8 microseconds, to be the “true” average throughout the rest of the report.

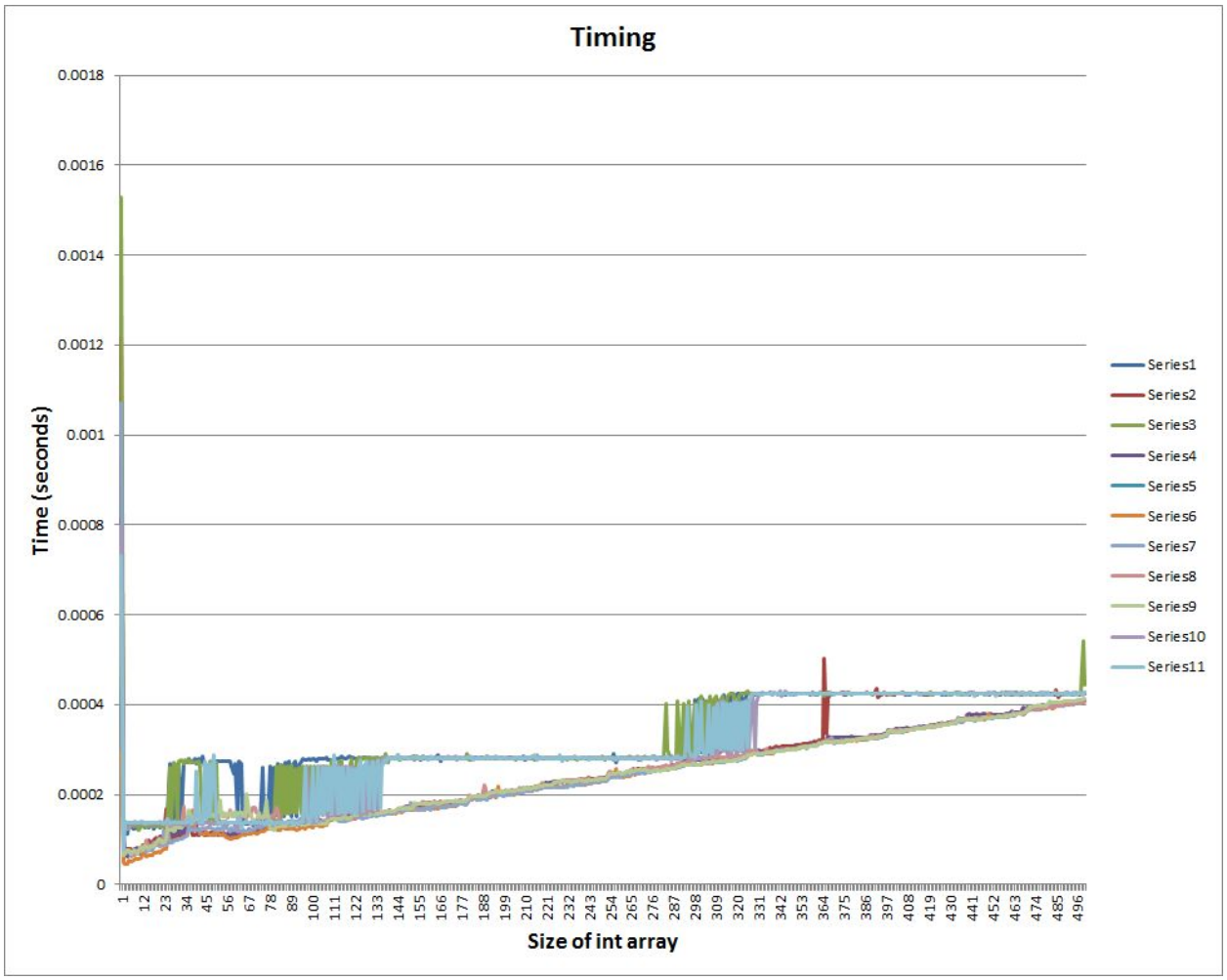
The average for two boxes was 17.12 times longer than the average for one box. This is because the one box run was able to use shared memory, which is a lot faster than the network transfer that had to be done on two boxes.



Part 3: Timing Increasing Ints

The purpose of this part of the projects is to see how many ints we can pass back and forth before there is a large increase in timing. I did this by passing an int array back and forth between processes on separate boxes, forcing it to use the network and a buffer rather than shared memory. The int array started at a size of 10 and went to 5,000, increasing by 10 every time for a total of 500 passes. This was then run 10 times and plotted in the graph below.

There are some outliers, but overall two trends emerged. Three runs stayed relatively linear throughout the entire time. This could be due to reduced network load during these times or some optimization that is done through MPI itself. The other 7 runs spike in roughly the same areas though. They would vary greatly starting at 250 ints, then hit a solid line at 1,500. This jumped again at around 3,000 integers, then seemed to start spiking again right at 4,500. The early spikes can be written off due to network activity, but the spikes at 1,500 and 3,000 are due to the program hitting the maximum amount of data the buffer can pass. The buffer must be at or around the size of 1,500 ints.



Conclusion

The average time for one box was 2.15 microseconds. This is faster than the two box test, which had an average of 36.8 microseconds. The reason for the large difference is because the one box test utilized shared memory to pass back and forth rather than the network connections that the two box test was forced to use.

There seems to be jumps in the message passing between two boxes every 1,500 ints. This is due to the buffer size being at or around 1,500 ints large. Once more than 1,500 ints are passed, the communicators have to wait for the program to buffer all of the data.