Kurt Andersen
CS 615
PA1 Ping Pong
February 21, 2017

Introduction:

For this project we were assigned to send messages between separate processors and see how long it took for the cluster to respond. We were required to accomplish 3 tasks for this. One task was to measure the time it took for a message to be sent on the same box and see how long it took to be returned. Another was to determine the time it took for a message to be sent to another box and back. The final one was to determine the spikes in time when larger amounts of data are sent. In order to accomplish this task I had to learn how to use sbatch, as well as gain a better understanding of the MPI library.

Process:

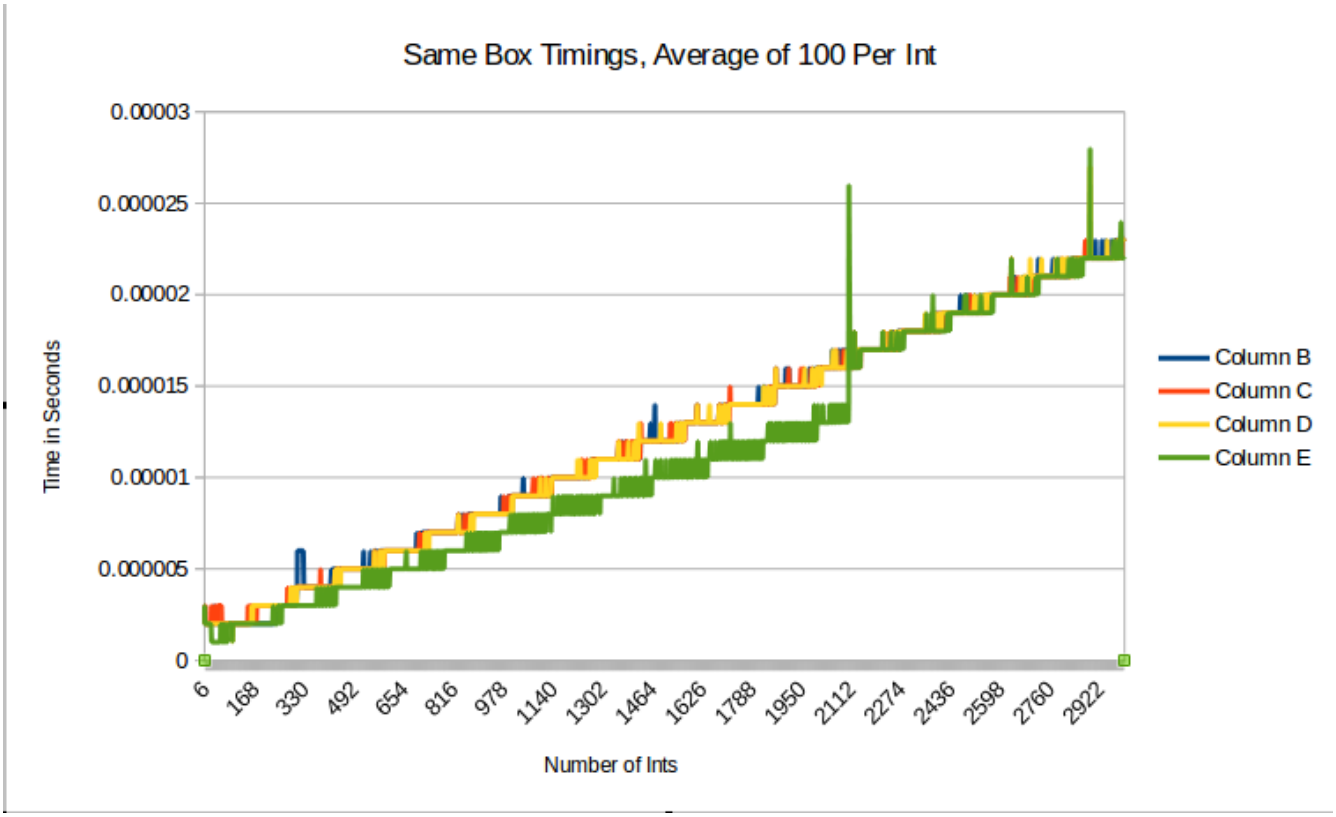
When figuring out the process to accomplish the tasks, as defined in the introduction, I ran into multiple problems right away. Since I am currently in operating systems and have had a tough time wrapping my head around how the parallel process actually work, it took me a few tries to get code giving me the right results. I originally had designed it so it would send one single message and have it sent back. After I got the single message working, I decided to go big with my data. I would send a number of ints, ranging between 1 and 3000, 100 times and take the average messaging time. This allowed for a more accurate reading for the data when it came to analysis.

When I was still coding though I ran into multiple errors. One of the biggest ones I ran into was when I was trying to print the data to a file for the differences between the messages. I forgot to tell only the master node to print, so at first I was getting a lot of NULL printing to my files. Once I figured out how to resolve this problem everything went a lot smoother. Another issue I ran into was other students hogging the resources on the cluster. There were multiple times when people when running the same process for over 40 minutes. This made it more difficult to run my processes because other students were attempting to run theirs as well. Which in turn ended up taking all of the resources

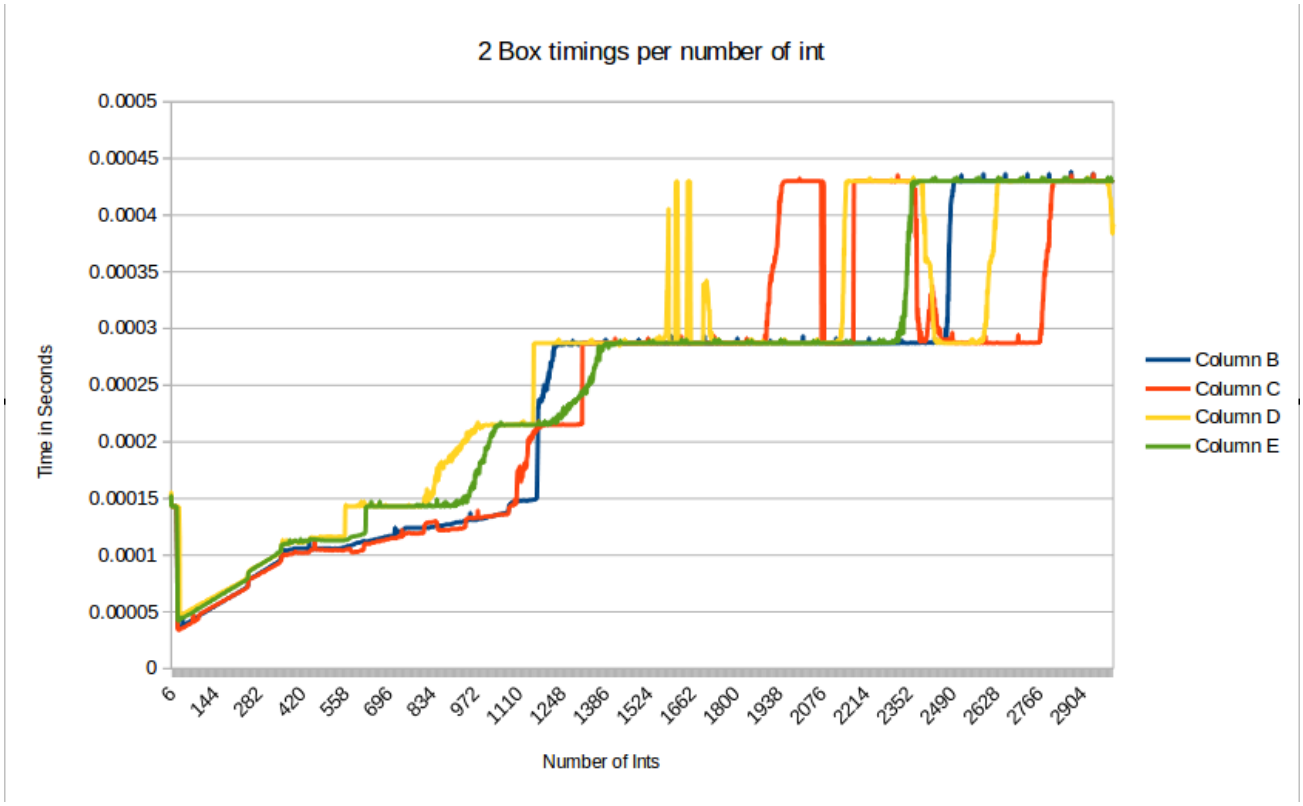
on the cluster. I am hoping in the future other students learn to put a time limit on their processes in order to allow other students to take full advantage of the resources available.

Results:

The results I obtained visually made sense to me. When I sent 1 int to a processor in the same box, my average message time was 0.000002 seconds, and when I would sent 1 int to a processor in a separate box, the average message time was 0.000153 seconds. This is obviously due to the processor having to have to put the data into a packet and send it across the network when sending to a separate box. The thing that perplexed me is that when sending multiple messages to another box, is that the time started spiked, and then dropped fairly quick. I am assuming this is due to a latency issue when first contacting the separate node. As shown below are two separate graphs. One showing the average time with 4 separate tests for message sending in the same box. The other graph shows the average time it took to send the same type of data to a separate box.



In this graph the growth was linear because the data was being sent in the same box. The X axis represents the number of integers being passed to another processor and the Y axis represents the total time for the data to send, and be sent back to the master process. Each different colored line represents a separate test. It has a linear growth because the amount of data being added was gradually getting larger and larger. The reason why it is linear is because the processor did not have to store the data being transferred into a packet, it was just shifting the data in the same box. This allows for quicker message passing. There are some random spikes in time, due to what I would presume are other people accessing the cluster at that time causing more traffic on the cluster. The data for this graph can be seen in one of four files. These files are located under the output folder in the project directory and are labeled “oneBox_XXXX.csv” The x's represent the job number from the run.



In this graph, it showed data being sent between two separate boxes. The X axis represents the number of integers being passed to another processor and the Y axis represents the total time for the data to send, and be sent back to the master process. Each different colored line represents a separate test. In this graph there are points where the graph spikes up and then becomes level. This happens because when the master node is communicating to the other nodes, it would have to pack the data into a packet and send it across the network. The points where the graphs spike and become level, is when a packet would be full, and another packet would be added to the message causing the receiving end to take more time when unpacking the data. There are random spikes that I would assume are other people accessing the cluster causing traffic on the cluster. These results I believe are what I was looking for in this project. The data for this graph can be seen in one of four files. These files are located under the output folder in the project directory and are labeled “TwoBox_XXXXX.csv” The x's represent the job number from the run.

Conclusion/Future Work:

Overall I feel as though this project gave me successful results. The data represented in the graphs reflected what I believed would happen with the Ping Pong code. When moving data on the same processor, the time just increases linearly because the total amount of data being moved on the same box. The transition between two boxes was what I assumed would happen as well. However I was unsure when the jump for the second packet would occur, but after running multiple tests, I was able to determine that each packet contains about 1200 integer values. In comparison, sending 3000 integer values on the same box takes approximately the same amount of time sending about 30 integers to a seperate box.

In the case of future work, I believe it is important to have this basic knowledge of message passing and how it can affect the duration of my programs. Understanding when and how long certain messages take to send to other boxes can change how I design parallel programs. I can have more processes sent to the same box, and have fewer sent to outside boxes. This I feel could improve overall speed when processing massive amounts of data.

Special Mentions:

I would like to note that I spoke with Vinh Le, Gunnar Wambaugh, and Jeff Williams in order to figure out a good way to tackle this program. Vinh Le also taught me how to construct a bash script so it would run everything I needed to from it. The information shared between Gunnar and Jeff was different approaches as to how to properly send multiple messages and figure out the timing methods.