Project Assignment 01 Report

CS415

2-22-2017

By: Jervyn Suguitan

Description of Assignment

Project 01 main purpose was to teach students how to do message passing (ping-pong) between two processes and see how the timings of this ping-pong message passing changed with different variables such as hardware and variables. This was accomplished by using two functions, MPI_Send and MPI_Recv. The project was split into 3 parts a) message passing between two processes in 1 box, b) message passing between two processes in 2 boxes, and c) finding how many integers are needed to create some a jump in the timing. The timing function that was used to time the message passing for this project was MPI_Wtime.

Part 1: One Box

The first part of the project tasks students in figuring out the time it takes to pass an integer from two different processes on the same box. To find the most accurate time, each time the program would run, a single integer would ping-pong between the processes 100 times. After passing the integer 100 times back and forth, we get the times of all these runs, average them, and print out the average overall time, The algorithm used to find a single time (a single ping pong message) was using MPI_Send and MPI_Recv. The code uses a master process and slave process. The master sends an integer to the slave process using MPI_Send. The slave process waits for a message by using MPI_Recv. Once the slave receives the sent integer, the slave process changes the value of the integer (I did this to show that it was actually sending back and forth an integer correctly) and immediately sends back the integer to the master process using MPI_Send. The master process now receives the changed integer using MPI_Recv and the time for all this occurs is held in a variable and later averaged for the 100 runs.

Problems that I have encountered during this part of the project is that I didn't understand how MPI_Send versus MPI_Isend worked. Another problem I had was that my timing algorithm wasn't the most efficient so it each timing period was slightly slowed due to poor optimization of my code. Also I did not declare a value of my variable "msgtag", which is the fifth argument of MPI_Send and MPI_Recv, this made the timings for both this part of the project and the next part of the project exactly the same.

Figure 1 shows the average times of 10 different full program trials. The average of these 10 different trials is 2.20E-06 seconds. The 3 largest outliers are seen in trials 1,3, and 5. These outliers are probably due to network congestion since many people were using the h1 node as these trials were running.

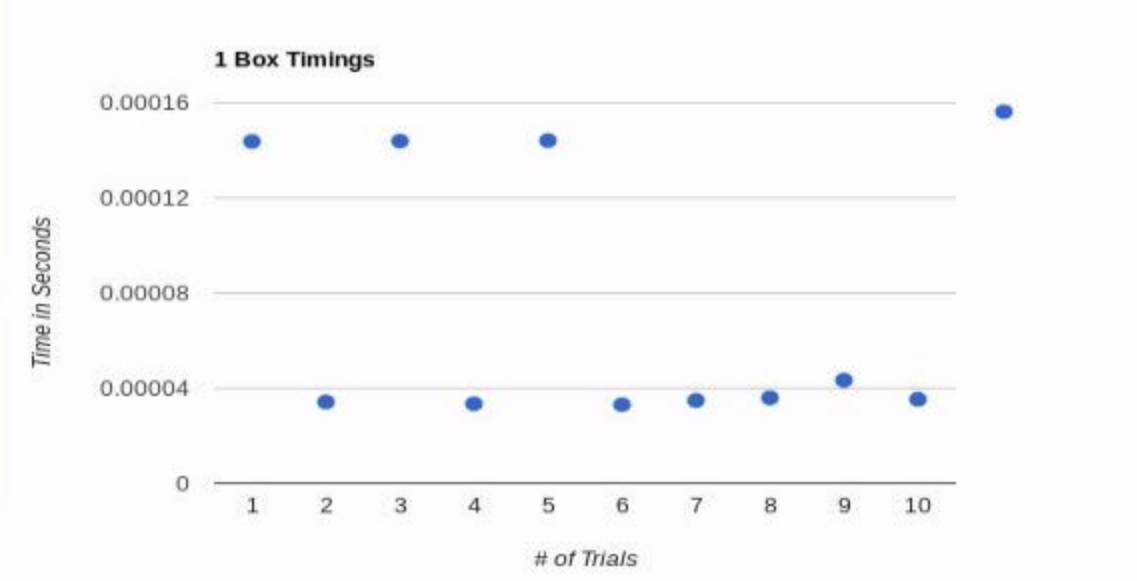

Fig. 1: A Scatter plot that shows the average times of different trials for message passing between a single boxes. The average time of these trials is 2.20Ee-6 seconds. The most likely reason for the outliers is due to many people using the the h1 node at the same time.

Part 2: Two Boxes

The next part of the projects tasks students students in figuring out the time it takes to pass an integer from two different processes on two different boxes. Since the two different processes are on two different boxes for this part of the project this will cause a slowdown compared to a single box. This is due to message passing on the same box uses shared memory which is much more faster that message passing between two boxes.

The code for this part of the project is exactly same as first portion. The only thing that changed is the line of code that is used to run the program. Instead of using “-n2” which tells the computer cluster to use two different nodes from the same box, this part of the project makes us use “-N2” which tells the computer cluster to make the program run with two processes from two separate boxes.

Figure 2 shows the averages of times from 10 different trials. The data is a little more spread out as these tests were taken at different times as the middle tests were occurring when many students were using the network which could have slowed down the program and timings. The average time from all 10 trials below is 6.82E-05 seconds. We can clearly see that the average time for two different boxes is much higher compared to the single box message passing. To reiterate the main reason that the single box is faster is due to shared memory.

The problems for this part of the project are also seen in part 1 of this report.

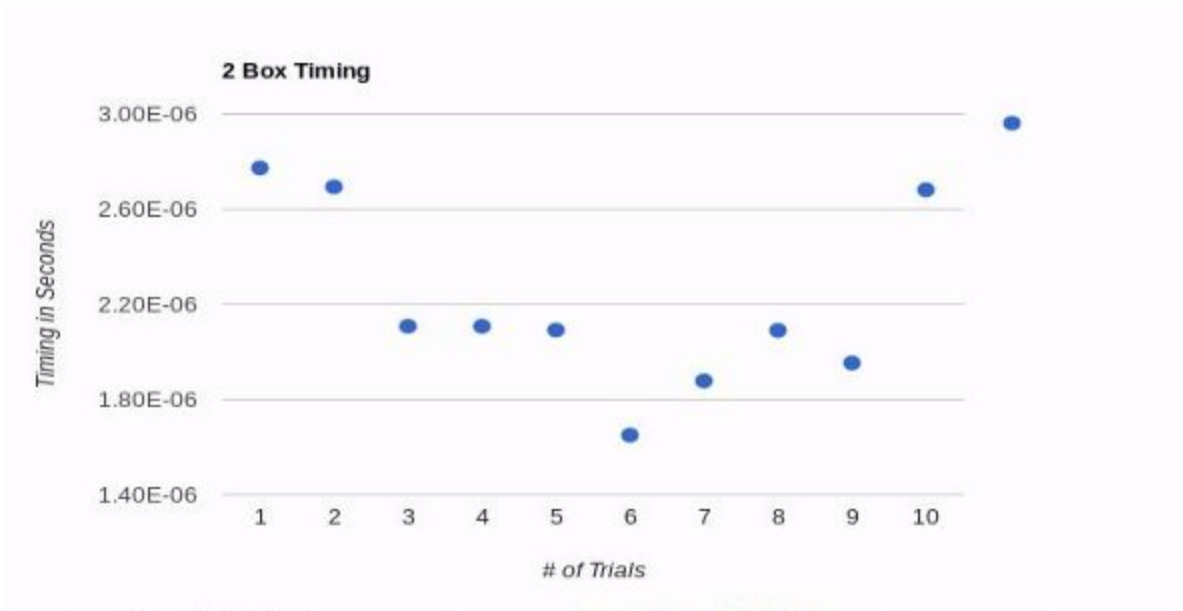

Fig. 2: A Scatter plot that shows the average times of different trials for message passing between two different boxes. The average time of these trials is 6.82E-05 seconds. The most likely reason for the outliers is due to many people using the the h1 node at the same time. The times are much larger (slower) here compared to Fig. 1 due to shared memory.

Part 3: Timing

The last part of the project tries to make students figure out how many integers being passed would cause the time to increase during a ping pong. For this part of the project student would need to use two different boxes since if the processes were sending between the same box, shared memory would be used and essentially the processes would copy data to other parts of data and not message pass. For my code, I sent different amounts of integers ranging from 1 integer to 10000 integers and after determining how many integers that would be passed, the program sends that amount

of integers 100 times and then find the average time. Figure 3 shows the results of 5 different trials. The differences of the trials can be due to network issues such as many students using the h1 node and multiple programs running at the same time.

Problems I had during this part of the project was figuring out how to change my code from the first project as I did not understand what the second argument of MPI_Send and MPI_Recv did. This argument states how many elements in the send/receive functions can hold. For this part of the project I had to understand that this argument changes so I used a double for loop in which the outer for loop changed the number of integers that could be passed at the time, increasing 5 integers at a time. The inner for loop is similar to the previous program in which it looped this new buffer size being passed between processors 100 times.

Figure 3 has many “jumps”, parts of the line graphs where the line increases very quickly. However these jumps occur around specific period of times, specifically when a number of integers are passed. When one of these processes pass data they send a packet, and based on the graph we see around every 1800-2000 integers a jump occurs. This jumping may occur since the packet size, the amount of data a process can pass. I estimate the packet size can hold around 1800-2000 integers, this explains jumps, since if a processor passes more than 1800-2000 integers, more packets are needed to send the data. Also the jump times can change since the packets of info that are passed between nodes may not arrive sequential, which can create jumps.

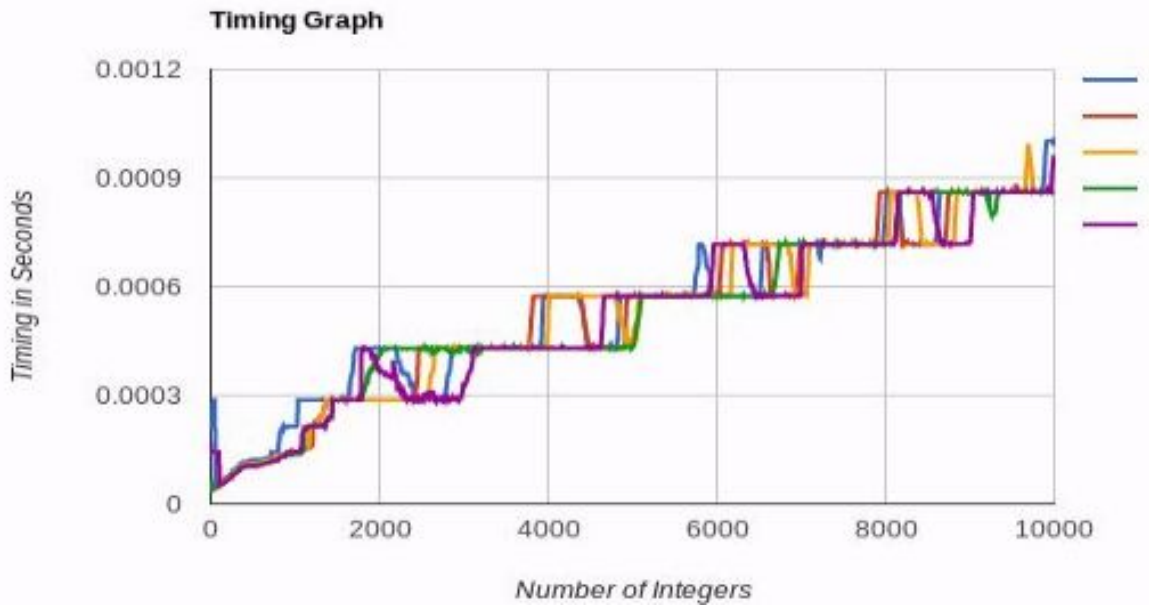

Fig. 3 Line graph that shows an increase in time as more integers are allowed to pass. Jumps may occur due to several reasons like people using the network and packet size.

Conclusion

This project teaches students how to pass data between different processes in the box and using basic MPI functions such as MPI_Send and MPI_Recv. Teaches why certain timings could occur which could be from network congestion, shared memory, many students using the h1 node at the same time, etc. This project also teaches student about packet sizes and their relevance to timing.