PA 1

CS 615

By Samson Haile

February 23, 2017

Assignment Description

The objective of this assignment is to determine differences in timing when passing messages between processes. This involves comparing the round-trip send time of messages sent on the same computer to messages sent between different computers. Initial examination will compare how long it takes to send an int between processes, evaluating how much longer it takes for messages sent between different computers to that of messages sent on the same computer. Afterwards, the number of ints sent between different computers will be set to gradually increase until noticeable jumps in round-trip travel time for messages are observed. The times will then be plotted against number of ints sent in order to establish an observed pattern in the message travel times. An assessment will then be made to explain trend(s) shown in the graph.

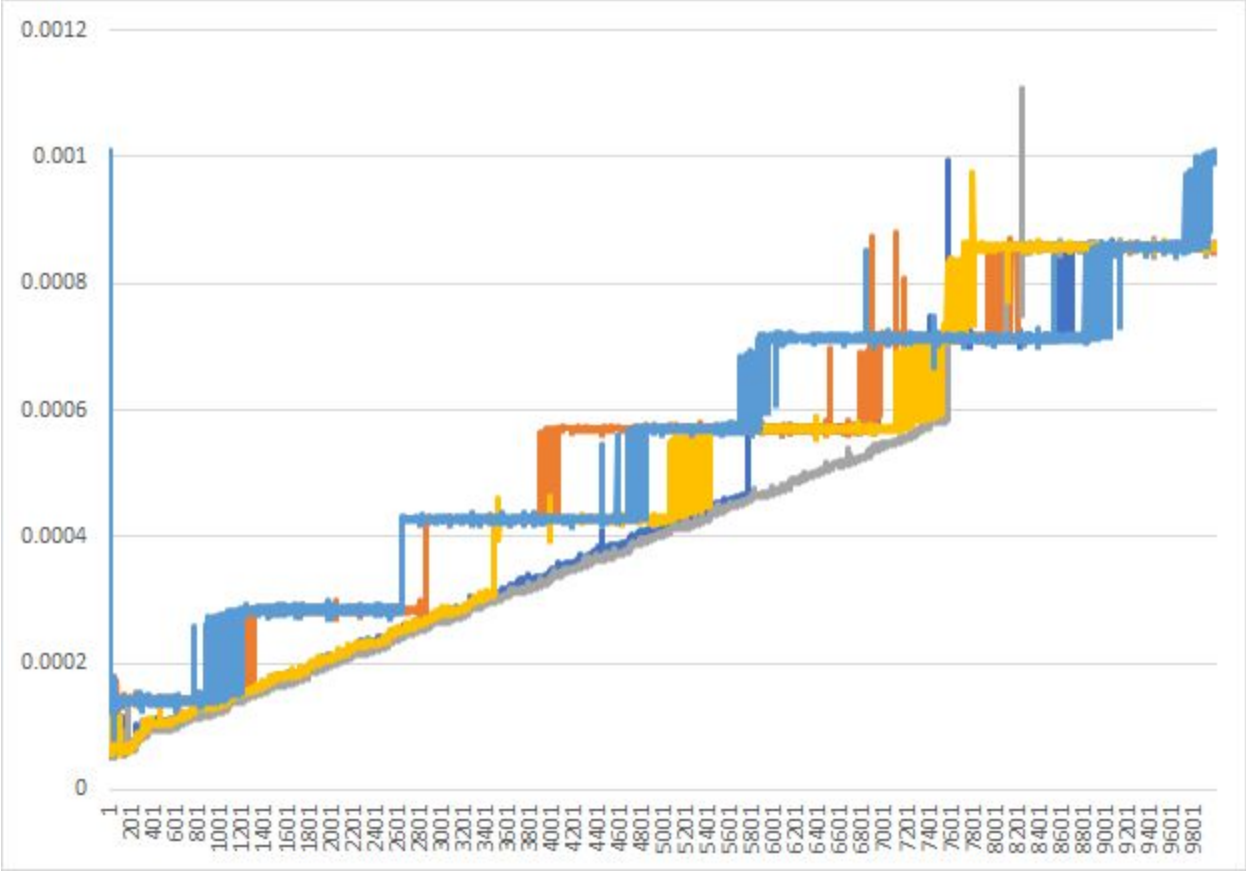
Assignment Methodology

The assignment was implemented in the form of a single c file, 3 batch files, and a makefile. The c file contained the code for initiating round-trip message sends, starting with the initialization of mpi. Then the code is partitioned into two sections, one for the master node initiating the message send and one for a slave node which receives the message and sends another message back to the master node. The master node is responsible for initiating the timer when it sends the first message to the slave node. It is also responsible for stopping the timer when it receives the return message from the slave node. The program holds command line argument functionality to specify a number n such that the program will increment a variable x from 1 to n, timing how long it takes x ints to travel in a message round trip prior to each incrementation.

The c code is compiled through the usage of makefile which is capable of producing an executable to run the code, as well as the ability to remove the produced executable from the directory. Once the executable is produced, one of three batch files can be run in the form of the command sbatch <file_name>. The batch file One_box.sh measures the time it takes to send a single int between processes on the same computer. The batch file Two_box.sh measures the time it takes to send a single int between processes on different computers. The batch file Timing.sh measures how long it takes to send a message of sizes ranging from 1 to 10000 ints.

Data and Analysis

To begin with, the comparison of times between messages sent on the same box and messages sent on different box yielded expected results. The messages sent between different boxes yielded longer message travel time than that of messages on the same box as extra time is needed for messages to travel across the data bus connecting the different boxes. As for the timing of messages carrying different sized buffers, message travel time increased as the message buffer size increased. The graph below illustrates the increase.



Observing the graph, it can be seen that the travel times tended to jump when they hit a certain amount of integers in the buffer. This specifically happened about every 1000 ints added to the packet. The presence of the instantaneous jumps in the graph can be explained by the mechanism packet switching uses when sending messages. In general, packets have a predetermined maximum size when sending data between computers. Therefore, when the maximum size of a packet is exceeded, even by a few bytes, the message requires the addition of the time it would take to send one additional packet.

Considering the form of the graph, it would be reasonable to hypothesize that the maximum packet size is set to carry around 1000 ints. However, some trials show that send times persist over increases of 2000 ints. A possible explanation for this phenomenon could be overlapping

packet sends, i.e. the master node sends the next packet while the slave node is still receiving preceding packets. Another notable aspect of the graph is the linear increase in send times at the beginning of some trials. A side note to this observation is that many of the nodes on the cluster were active during the performance of these trials and so the initial set of small sized packets flooding the cluster may have caused the initial linear increase, which disappeared as packet sizes got larger.