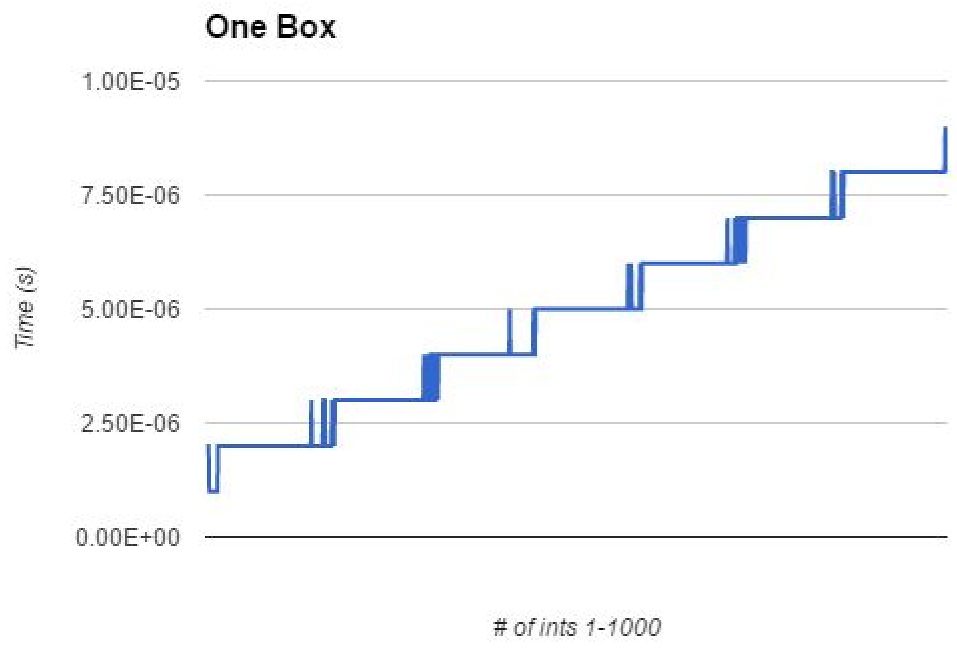
Jake Wheeler
CS415
02/21/2017
PA1 - Ping Pong

Jake Wheeler
CS415
PA1
02/21/2017

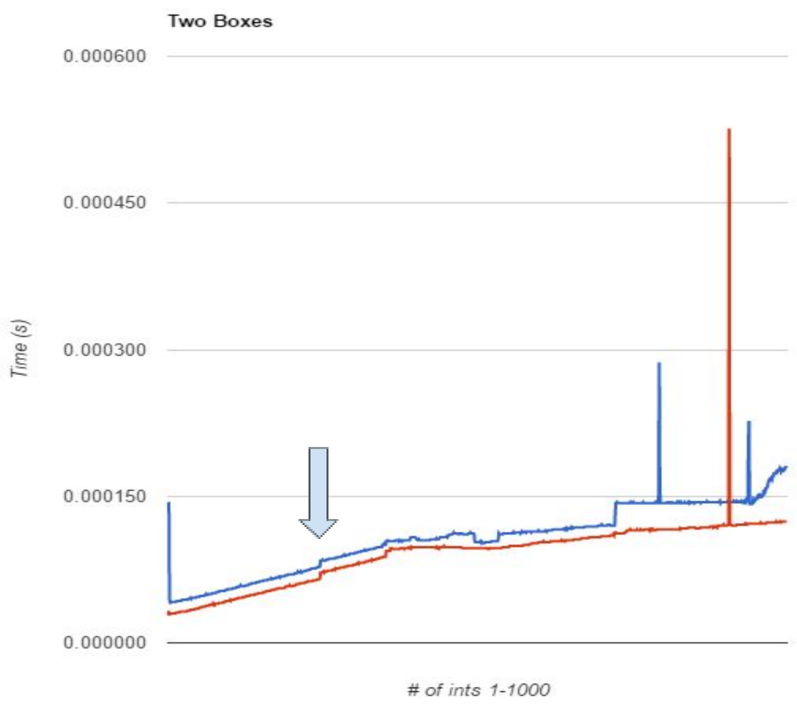
One Box

The timing between processes on one box was achieved by manipulating the command line arguments that srun accepts. By running "srun -n2 -N1" I specified to run two tasks, on one box. The time it took to send integers between processes on one box was very quick, taking on average 2E-6 s for one int, and 8E-6 s for 1,000 ints. When graphed, some interesting things can be seen. Adding integers doesn't seem to increase the time until about 150 integers are added, then the time jumps up consistently by 1E-6 s. This could be due to various factors, such as too low of a resolution for timing. I was only able to time up to the microsecond, and sending more ints may very well be growing linearly down to the nanosecond or even faster. Having my maximum resolution at the microsecond would hide those results from me. Another potential (and more likely) reason for this behavior could be the size of MPI's buffer. If it only had room for ~150 ints then it would need to wait for the buffer to clear before adding another 150, causing the jump in timing.

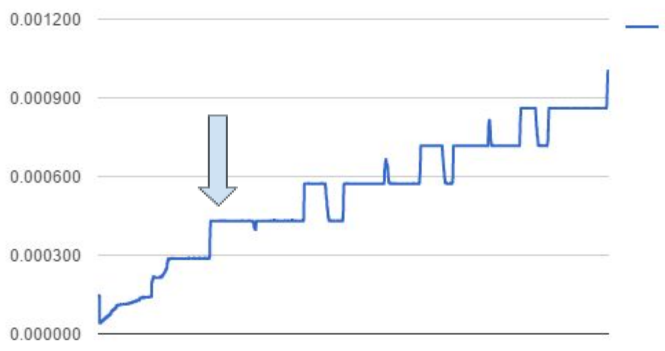


Two Boxes

The timing between processes on two separate boxes was achieved by running "srun -n2 -N2 --ntasks-per-node=1" specifying 2 tasks, 2 nodes, with 1 task per node. The time it took to send integers between processes on one box was still very quick, but much slower than on one box taking on average 1.44E-4 s for one int, and 1.81E-4 s for 1,000 ints. Unlike the test on one box, these numbers fluctuated much more between tests likely due to network traffic. Some significant things can be seen with the graph for these tests as well, although they are bit more difficult to see due to the “noise” in the network. At about 250 ints, the time jumped from a linear growth by about 1E-5s. This consistently happened across multiple tests, so it is not due to the variability of network latency like some of the other spikes in the data. This occurred I believe because of packet size restraints. With the maximum size of a packet being typically 1500 bytes, only so many ints can fit in one packet before a second packet, with its header data must be crafted, explaining the jump in time. The jump can be seen clearly at least two times toward the beginning of the graph, but gets a bit trickier to see in the second half due to network noise.



When increasing the number of ints being sent to a higher number, say 10,000, we start seeing a pattern that is closer to the one box test. This again is likely due to some sort of buffer restriction, in that after a certain amount of ints (in this case, 2,500) are sent, the buffer will be filled and need to be emptied and used a second time for a single message.


0.001200
0.000900
0.000600
0.000300
0.000000