Charles Arnaudo

Prof. Mohammed Anwaruddin

Parallel Computing

10 August 2018

**Odd-Even Transposition Sort Using MPI**

I designed the program with uniformity in mind. I also adapted it from examples in the book. The only time the code performs different actions is when the process running is the root process. I also leveraged C’s qsort function to sort each local array during each MPI process. I wrote a function called compare which is what qsort uses to sort the array items. Each process creates an array and populates it with a random value between 0 and 2048. We then loop through the available processes and check their ranks, specifically that of its neighbor. Once we return to the root process, we get the time taken to gather all the processes and print the time.

As the chart below shows, computational time decreases as we increase the number of nodes used. For each individual node, however, increasing process count does not generally decrease computational time. This discrepancy is apparent when using an odd number of nodes. Because the work is not distributed evenly, a lot of time is spent waiting for the one node with the most work to gather.

With this data in mind, in general, to decrease computational time increase the number of nodes used and pick a number of processes so that nodes mod process equals zero, as long as process is less than node.