Assignment 1 : MPI

Kat Young
MS Computer Science
Richmond, VA
youngk6@vcu.edu

I. Small Dataset

For the small dataset, using a single-thread ran in 56 ms. When parallelized, 2 processes took 266 ms to run, 4 processes took 288 ms to run, 8 processes took 313 ms to run, and 16 processes took 344 ms to run. I calculated speedup based on each number of processes, dividing the serial run time by the parallel run time. For 2 processes, the speedup was 0.21. For 4 processes, the speedup was 0.19. For 8 processes, the speedup was 0.18. For 16 processes, the speedup was 0.16. As processes were added, the run time continued to increase, starting with a large initial jump at running this dataset on two processes. This is because of the overhead costs of running multiple processes and the fact that there isn't enough work to be done on each process to make it worth while. Thus, no matter how many processes the code is running on, there will not be a speedup where run time decreases.

II. Medium Dataset

For the medium dataset, using a single-thread ran in 9035 ms, 2 processes took 4823 ms, 4 processes took 2610 ms, 8 processes took 1634 ms, 16 processes took 1277 ms. The speedup for 2 processes was 1.87. The speedup for 4 processes was 3.46. The speedup for 8 processes was 5.53. The speedup for 16 processes was 7.08. As processes were added, the run time continued to decrease, becoming noticeably faster. Based on the speedup times, an estimated 93-94 percent of the code is parallel. Using an infinite number of threads and processes, the maximum speedup would be around 14.29!

When the code attempted to run on 2048 processes, it ended up faulting, since having that many processes ended up being too many to perform. While adding processes to a dataset the size of medium.arff can speed up run time, there comes a point where the returns for adding processes diminish. There is overhead cost with creating processes, and if too many processes are created and not given enough work, it isn't up to its potential efficiency.