Dylan Bonsell
Feb. 10, 2014

CS 322 Assignment 3
Writeup

Overview:

The findings in this assignment were insightful, and show a lot about how much messaging and the methods used can affect program run times and efficiency. The programs in this wrieup were run on a intel i7 clocked at 4.9GHz, 16GB DDR3 2133mhz ram, running on an ssd for increased program load time.

Compilation

I used a very basic form of compilation, as described below:

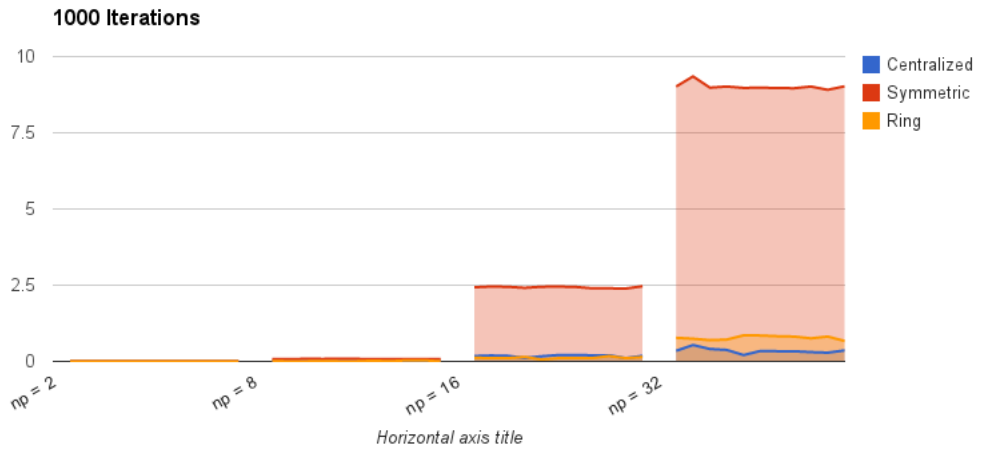
1. To compile, I used: mpicc part.c -o part
2. To run, I used: mpirun -np #processes ./part

Analysis

The way I analized run times is with MPI_Wtime, from the start of the program to the end, to measure in seconds. Then, I took values for 10 iterations for np = 2, 8, 16, 32 and 1000 iterations and 10,000 iterations in the loop.

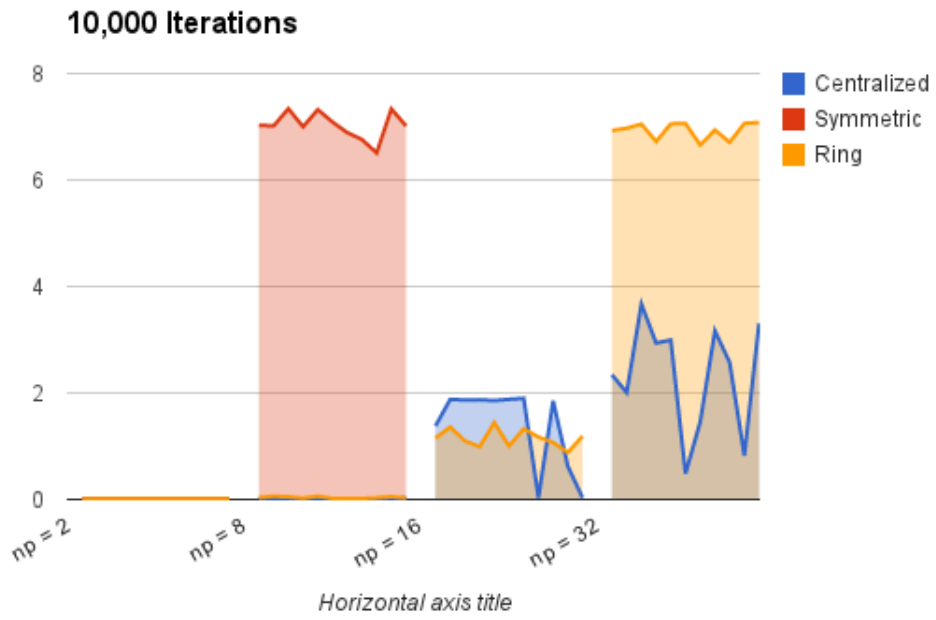
Each data value is run on a system at rest, I.E. no other tasks running.

1. 1000 Loop Iterations



These findings are interesting, but not surprising - Centralized is the most efficient, then Ring, and the worst by far is Symmetric. Centralized is just barely more efficient than Ring because ring requires each processor to send and recieve to 2 processes, whereas all processors that aren't 0 send and recieve to only 1 process in Centralized. Symmetric is, by far, exponantially worse, because it requires so many messages to be passed, for each processor, so it is not surprising that it is exponenitally worse.

2. 10,000 Loop Iterations



For this one, I ran into memory issues at $np >= 16$ for Symmetric, because of overflow errors. Other that that, the analysis is the same as before, except for some small deviations at $np = 16$, to which I accounted for by making the first value an average of all ten values. I believe these outliers to be time specific, perhaps the system had another high-intensity task running.

Sources

All of my data can be found on my Google Drive, with the following link:

Assignment 3 Analysis Data