Erikson Sodergren

Concepts of Parallel and Distributed Systems – Project 2

For this assignment I quickly found a working method to parallelize a bitonic sorting algorithm using shared memory and pthreads in C. The program makes efficient use of the number of threads it is told to use to split up the work, initially dividing each block of sortable elements up for multiple threads, but as the algorithm progresses, giving each thread many blocks to work with individually. For the sake of simplicity and minimizing the size of the input file, I wrote the program to work only with integers and always sort in ascending order, however very minor changes to the soruce code could change this and allow input to specify ascending/descending, and use doubles.

Once again, due to Kraken running slowly for initial testing, all recorded test data was obtained using a weaker machine with only 2 cores, in this case chianna was used. As a result, the runtime for more than 2 threads was terrible. Speedup obtained with the algorithm was 1.2 for 2 threads, .8 for 4 threads, and .3 for 8 threads on chianna, with slightly worse performance with the N=1024 dataset for 4 and 8 threads. In addition, some speedup calculations may be off because these tests were run over the course of an hour, but with short individual runtimes and other users may have slowed the machine down during some portions.

Below is a diagram illustrating how the program divides responsibilities among processors, with 4 threads and 16 data elements. An arrow represents a comparison with the larger number place where the arrow points. Each color is a separate thread doing its own work. Once the 3rd phase in any execution using 4 threads begins, as in this example, each thread continues as long as its current data needs, without care for the others or further apportioning of work. In this example they only continue for one more phase before the sort is complete.

