Parallel Sorting Algorithms in C Using Open Multi-Processing

Zietlow, Nathaniel

The purpose of the first part of this research is to determine the fastest algorithm to sort a random list of numbers into sequential order. I predicted that bubble sort would be the fastest because it has a very simple design and few lines of code. The second phase of the project covers how parallel processing will affect sorting algorithms. My hypothesis was that the more processors that were dedicated to the task, the faster the task would be completed. The data did not support the first hypothesis. When Sorting an array that is 100,000 numbers long, Shellsort was faster than Zietlow Sort by an eight second difference because it would move large numbers across the array more quickly. Whenever a list of numbers needs to be sorted into the correct order, the results from this project should be considered in order to optimize the efficiency of the sort. Using the most efficient algorithm will save 525 kWh of energy or $53 each year. In addition, it would save carbon dioxide emissions equivalent to the amount absorbed by eight trees. The data supported my second hypothesis that the more threads that were dedicated to a task the quicker it would go. The algorithm could save 3.5 minutes when sorting a list of about 250,000 items when it used four threads instead of one.