Multi-Threaded-Sorter Analysis

Yugant Joshi & Sydney Sigue

The comparison between run times on the Rutgers iLab machines is not a very fair comparison. The run time is heavily dependent on the specs of the machine and the number of other users on the machine and what programs they are running. So depending on when the program is run, different times will be given. However, taking the average provides a good way to measure the time it takes. Therefore while there are discrepancies in the run times, they can be normalized by taking the average. However, there will still remain outliers which skew the run time data.

Our Multi-Threaded-Sorter runs significantly faster than the Multi-Process-Sorter program. We believe this occurs because in the Multi-Process, we have to write a sorted file for every single file we encounter, while for the Multi-Threaded we are only writing a single file at the end. This significantly lowers the cost of writing to the system since it now only occurs once in the entire program. One way to increase the run time would be to do something similar to the Multi-Threaded program – create a single sorted file from all the files rather than sort each individual file. Because we used mergesort which runs in O(nlogn) time, our runtime for sorting all the files would be O(n*(nlogn)) for the Multi-Process sorter since for each of n files, we are calling mergesort. On the other hand, sorting in the Multi-Threaded is done in just O(nlogn) because the sort only occurs at the end.

For our implementation of the Multi-Threaded Sorter, we first read all of the files first into a large array which holds a single row data item. After reading across all of the files, we call mergesort on this final array that needs to be sorted. Because of this implementation (and not one where we sort each file as it comes in and then do a final merge), we believe mergesort is definitely the correct way to go. We found that for small files, quicksort performs faster than mergesort, but since we are reading everything into one very large array, we do not need to worry about run times for small files.