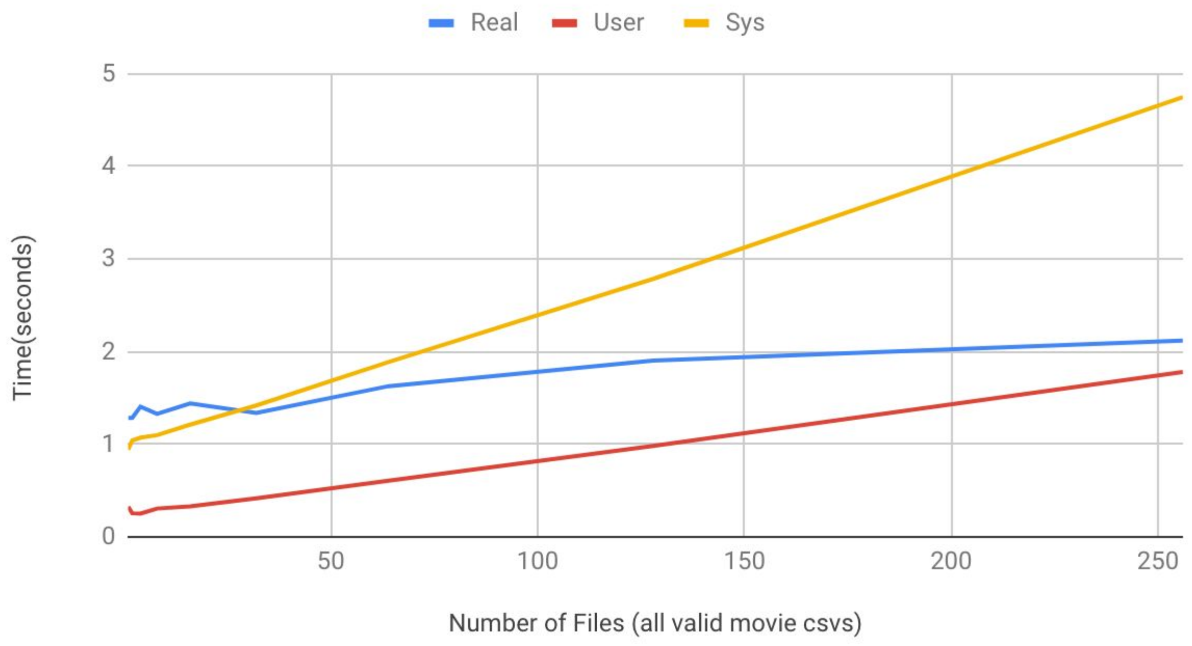
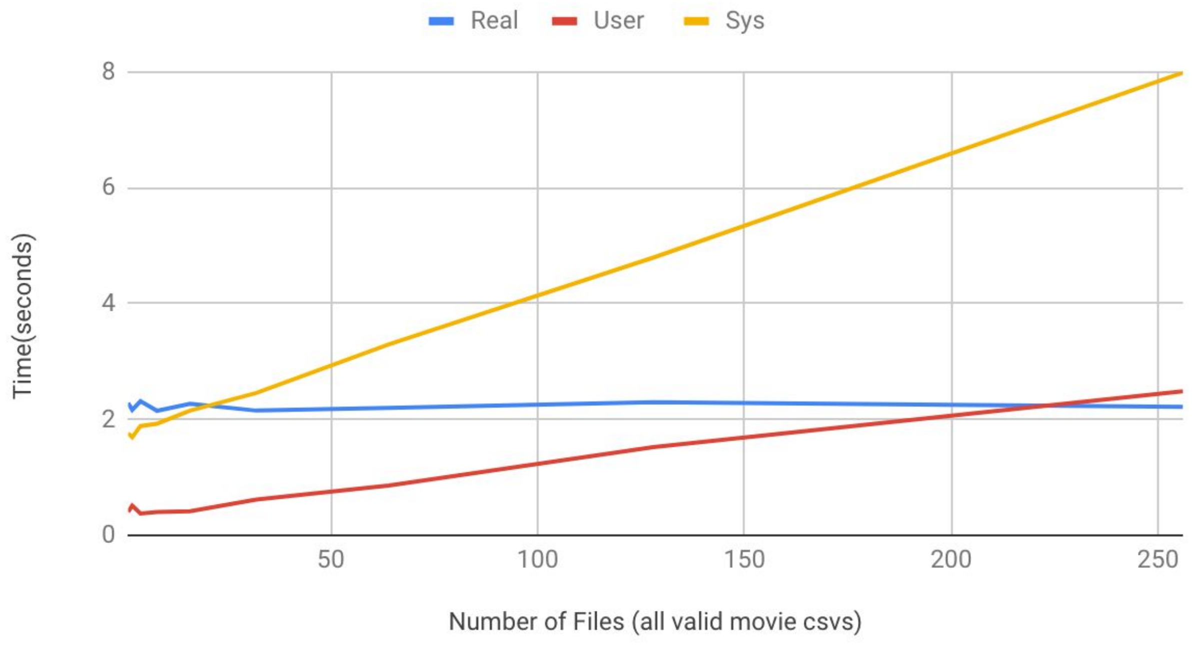
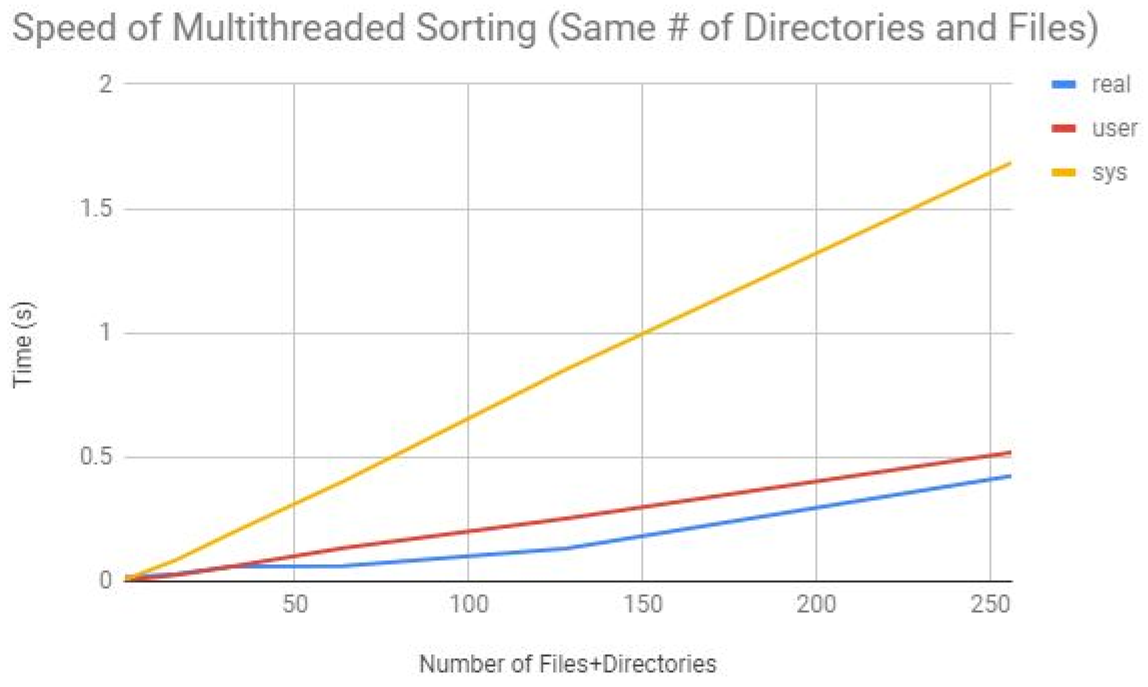
Time Analysis of File IO and Sorting using Multithreading

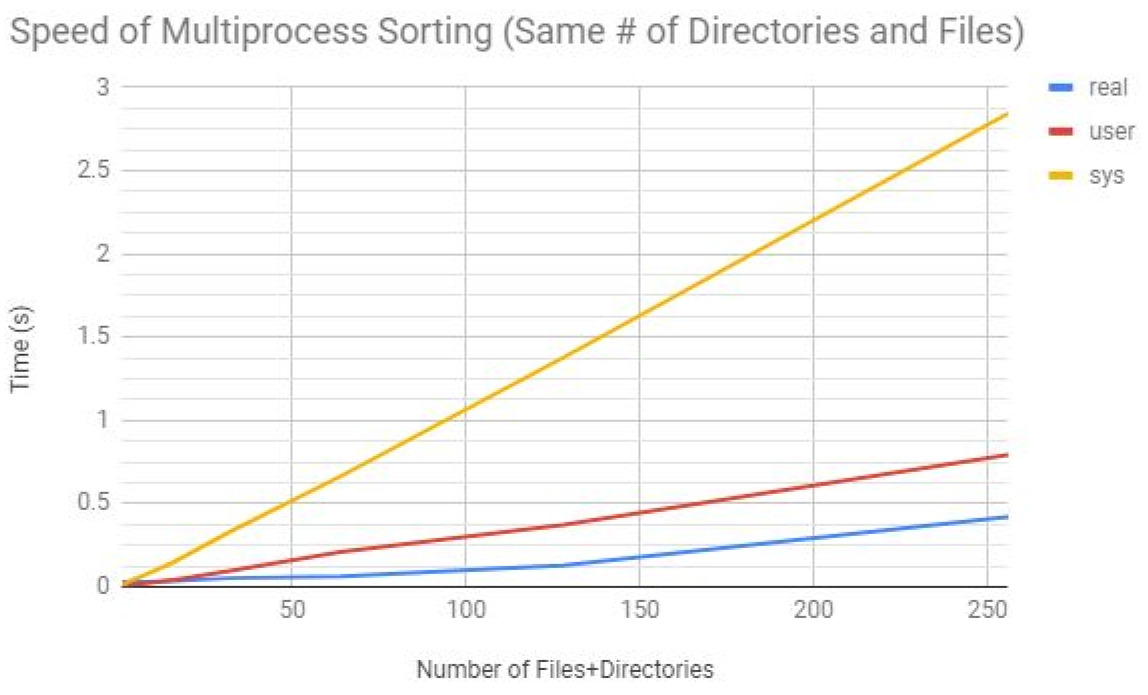


Multiprocessing with csv files Time Analysis



In the graphs above, the multiprocess and multithreaded sorters were run on a single directory with varying numbers of files. The files included 1 large CSV with 5000 rows of data, and many other CSVs with 50 rows of data each such that the total number of files equaled a power of two. It can be seen that the multithreaded version of the sorter has a faster run time. When testing multiple subdirectories, as seen in the graphs below, each folder included another folder, and a valid CSV containing 50 rows of data. Thus in the maximum case of 256, there were 128 folders and 128 files that were searched and sorted. Regardless of the number of files and directories, the multithreaded program ran faster than the multiprocess program. The comparison between run times is not necessarily a fair one. At some moments in time, it is possible that the iLab Machines are under heavy load and thus they process the code slower than normal. At other moments in time, it is also possible that the iLab Machines are under much lighter load and process the results a lot faster. In fact, this is the most likely cause for discrepancies in the program running time for the same program. Depending on how much load the machine is under, we would have different scheduling for the code and thus, have different amounts of time. However, it is clear that threading is a lot faster than multiprocessing. In threading, we have the liberty of concurrently reading files and directories while also sorting whereas in multiprocessing, we operate in a more linear fashion where we can only work on one operation at a time. However, the analysis is not completely fair since the specifications were different for scannerCSVsorter and for multiThreadSorter. For multiThreadSorter, we only had to write to one file at the very end. Whereas, in scannerCSVsorter, we had to write to multiple files. File I/O is a rather slow process and could very likely be why the difference in running time between multiprocessing and multithreading is so large.




Speed of Multiprocess Sorting (Same # of Directories and Files)
Time (s)
0
0.5
1
1.5
2
2.5
3
50
100
150
200
250
Number of Files+Directories
real
user
sys