Is the comparison between run times a fair one?

No, the comparison is not fair. First off, the thread sorter is required to resort (or place into a sorted data structure) on top of the original sorting. The fork sorter must also create new files and use IO to output all sorted data to X amount of files. The 2 are not comparable, depending on the amount of data given per CSV and how many CSVs are given one or the other might net a higher speed.

For example when run 1 CSV vs 1 CSV the thread sorting was faster than fork sorting. But when compared at 256 CSVs for both fork sorting became faster.

What are some reasons for the discrepancies of the times or for lack of discrepancies?

Mostly the dataset and the extra requirement of the threadsorter to keep every acquired CSV sorted in one main dataset. It adds extra overhead that the fork sorter does not have to deal with. Also as discussed above the time difference would be larger but the threadsorter only has to output to and create a single CSV file while the fork sorter would have to create X amount of CSV files to output to based on what was in the directory.

Furthermore, fork sorter does not have to wait when touching some data leading to a closer design of parallelism where the thread sorter sometimes needs to wait for other threads to stop using critical data before it can access it. Such as when they are inputting into the BST, slowing the process significantly .

If there are difference, is it possible to make the slower one faster? How? If there were no differences, is it possible to make on faster than the other? How?

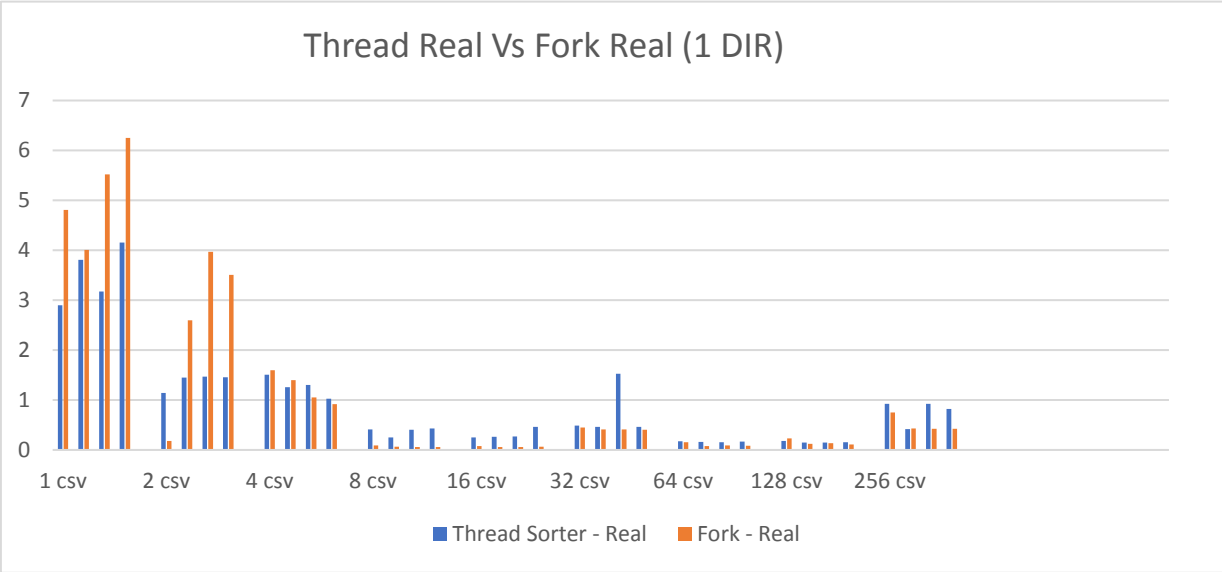
The slower one is dependent on the directory and CSV make-up. To make one faster one end would mean making it slower on another. Overall though our thread sorter could be made significantly faster if we were to forgo keeping every CSV sorted and instead follow the fork sorting and output them to their relative file sorted.

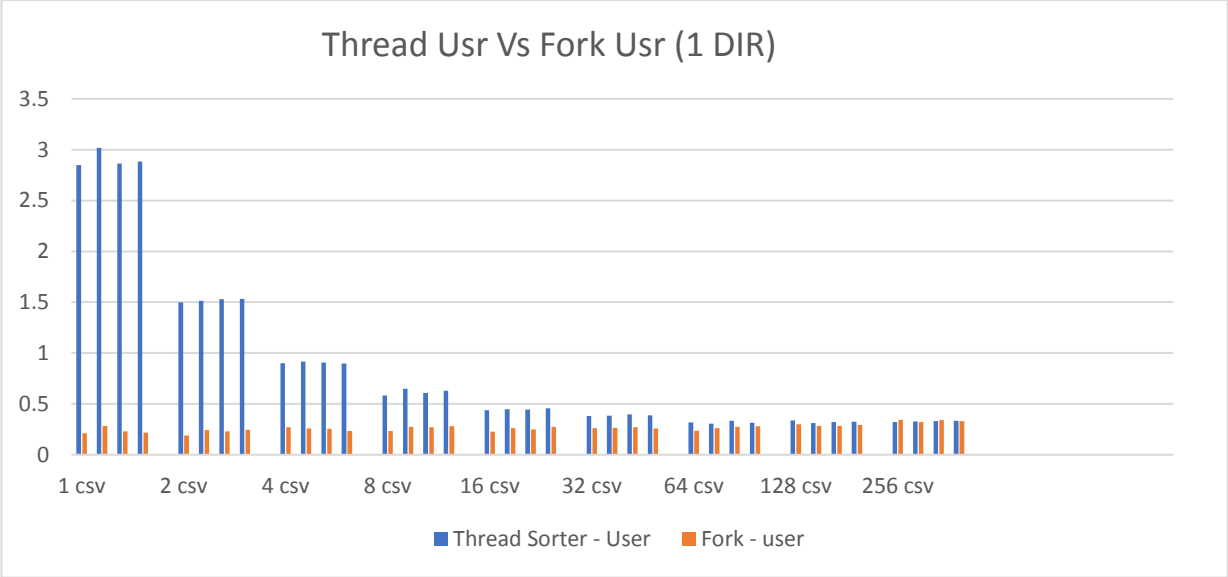
Is mergesort the right option for a multithread sorting program? Why or why not?

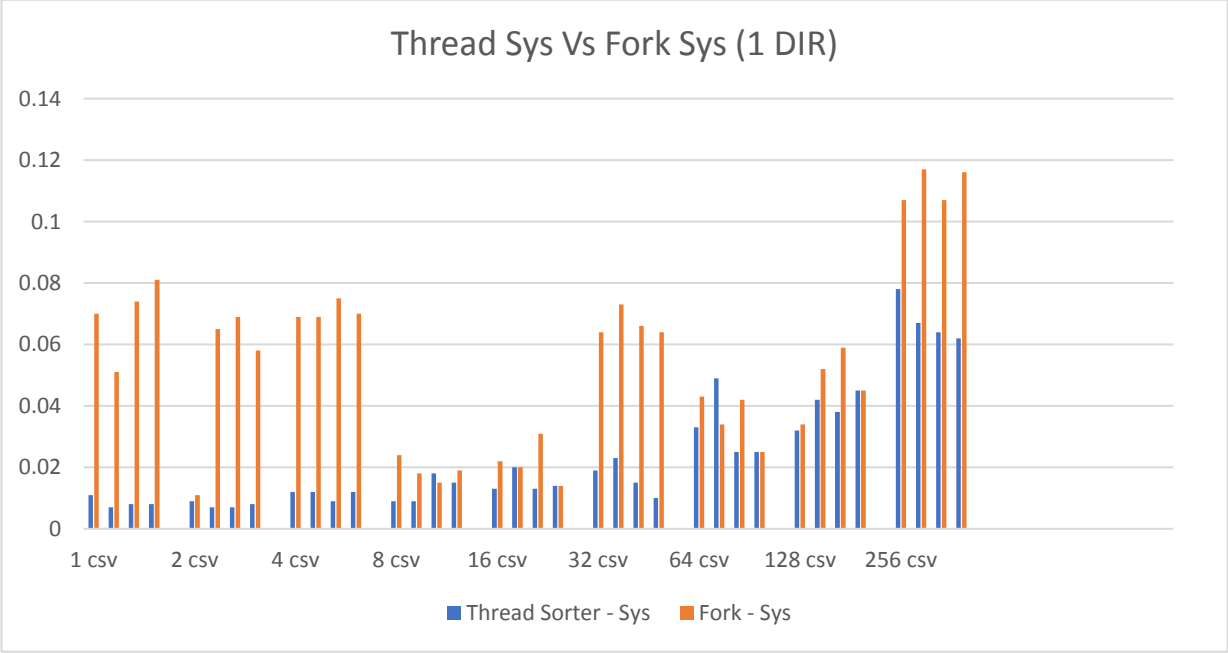
Mergesort is a divide and conquer algorithm. This allows it to be a great candidate for a multithread sorter. Currently the implementation we are using in thread sorter where every single CSV is sorted and then sorted again when combined is essentially a larger scale merge sort where each “chunk” is broken into a new thread and then is reassembled much like how mergesort works per file.

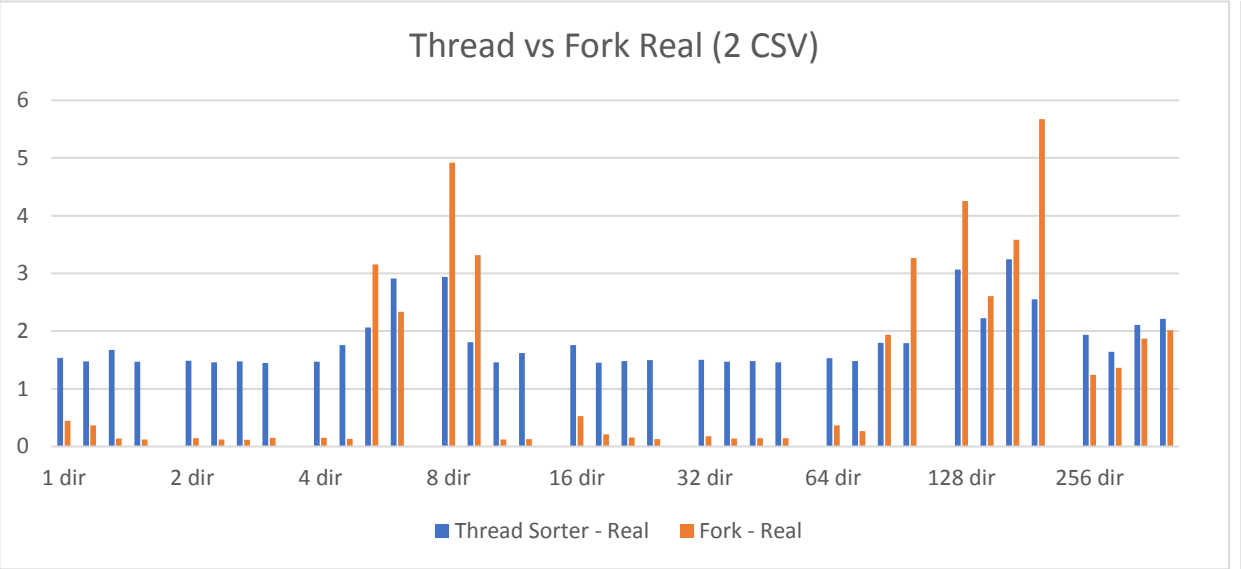
For example. If you had a large file of 5k lines merge sort would first break it down into 2 sets of 2.5k lines, which would be great for threads (one thread sorting 2.5k, one sorting 2.5k and combine when done). Where as currently we have say X amount of CSVs with Y lines each getting its own thread and then when the sorting on each is done they are combined just the like recursive ascent of merge sort.

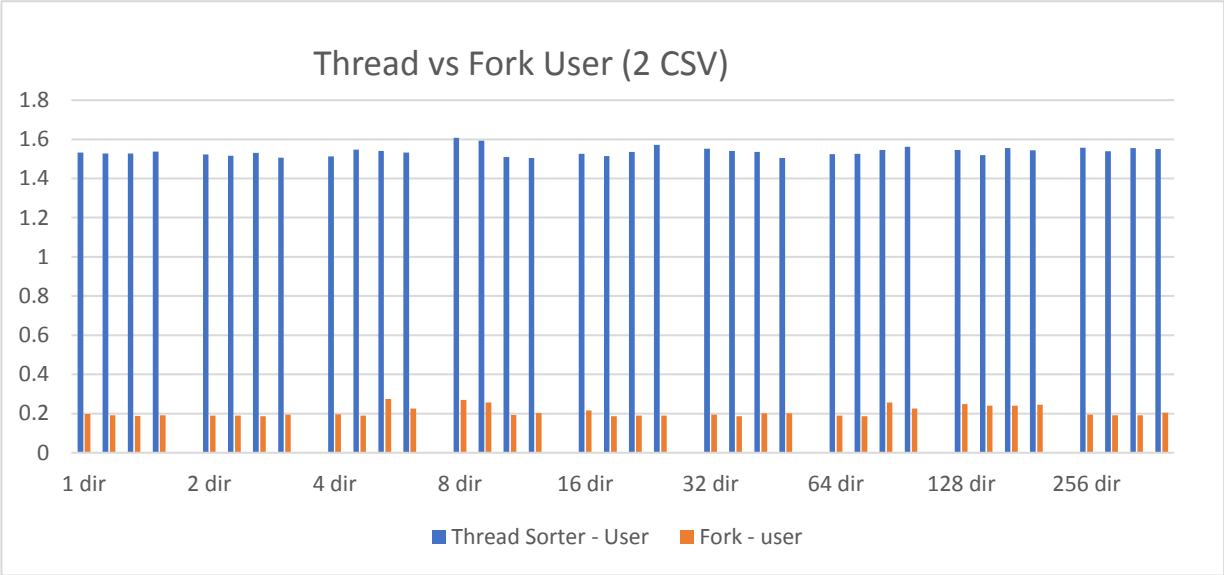
Other algorithms might be able to do it more efficiently sometimes, but due to mergesorts good average case, I think overall this is the best algorithm to be chosen for this situation. The main issue with mergesort is how much memory it uses to sort everything, with a large enough set of data it would become near impossible on some machines.

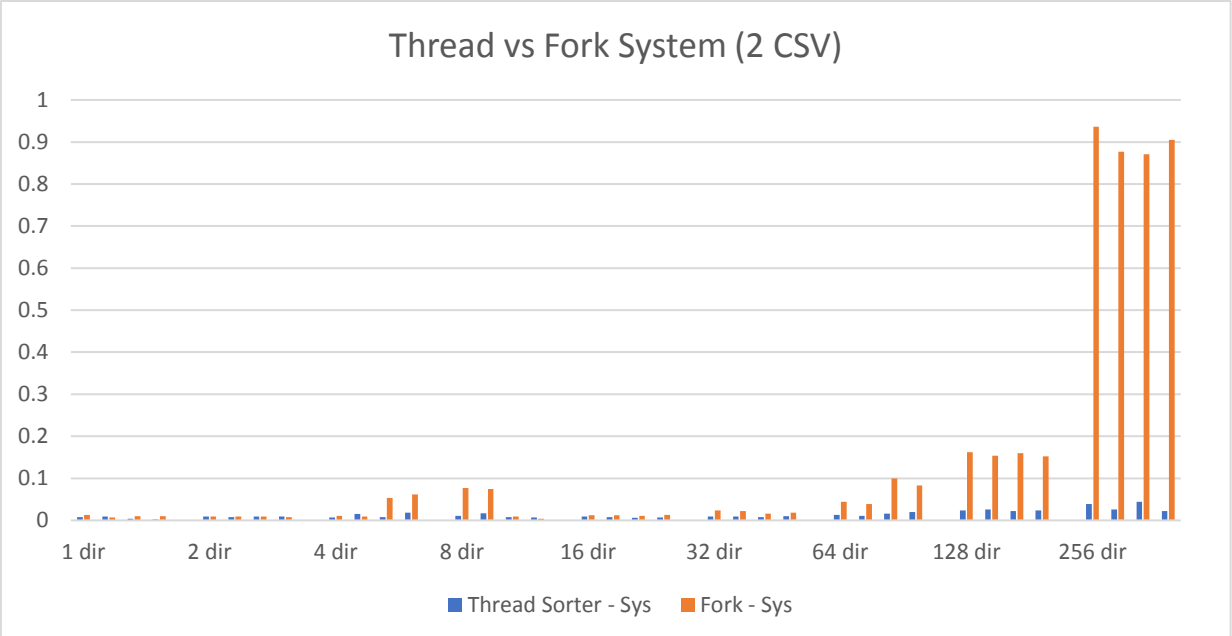

Thread Real Vs Fork Real (1 DIR)
7
6
5
4
3
2
1
0
1 csv
2 csv
4 csv
8 csv
16 csv
32 csv
64 csv
128 csv
256 csv
Thread Sorter - Real
Fork - Real

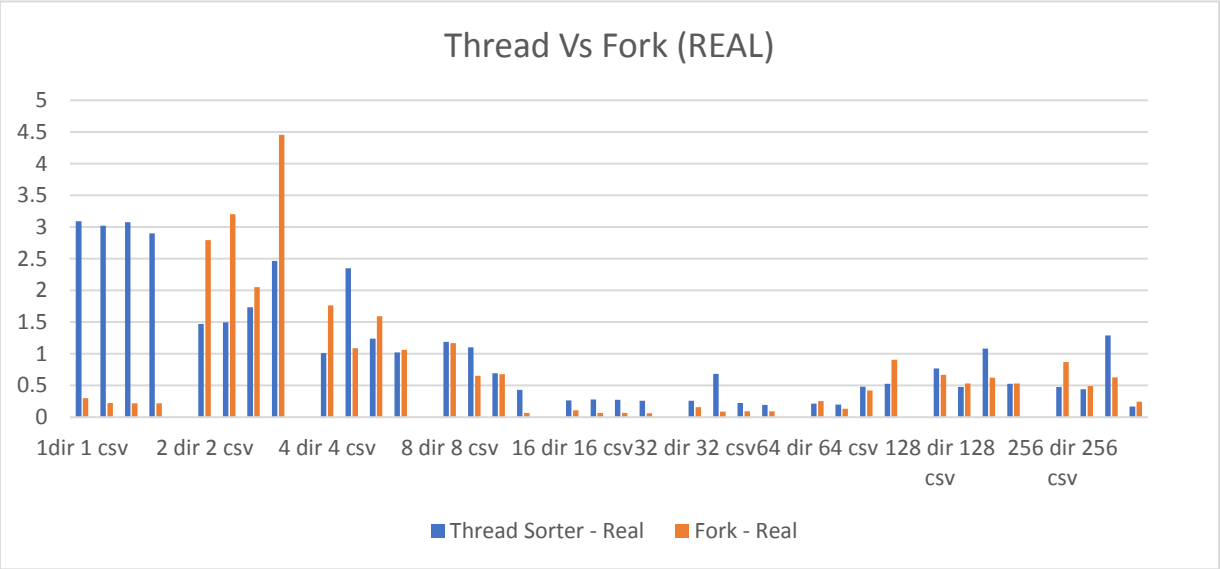

Thread Usr Vs Fork Usr (1 DIR)
3.5
3
2.5
2
1.5
1
0.5
0
1 csv
2 csv
4 csv
8 csv
16 csv
32 csv
64 csv
128 csv
256 csv
Thread Sorter - User
Fork - user

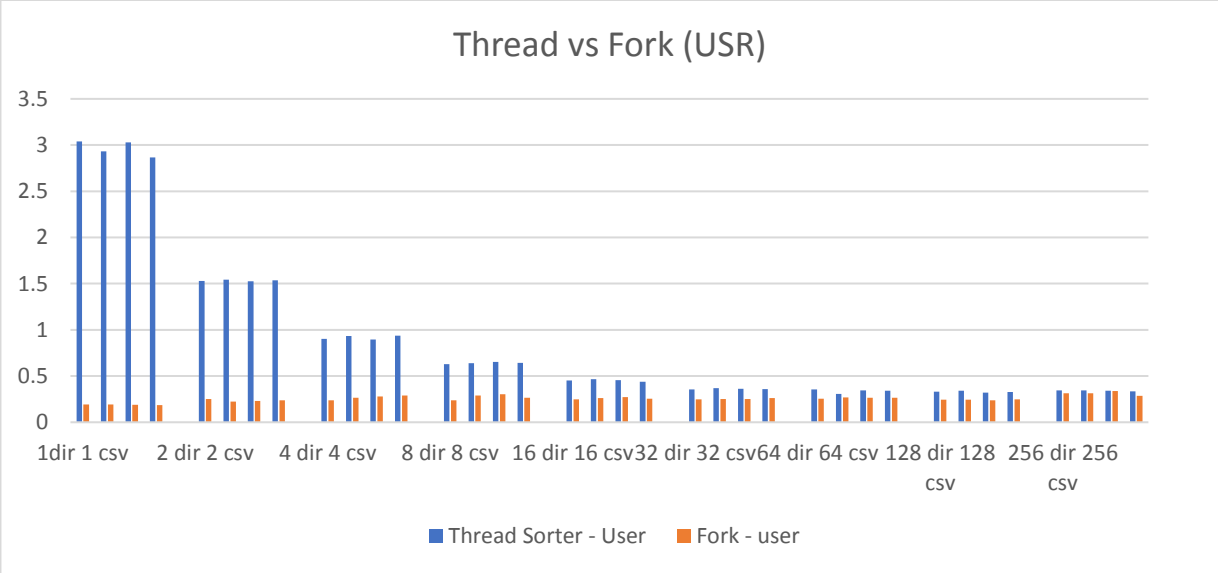

Thread Sys Vs Fork Sys (1 DIR)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1 csv
2 csv
4 csv
8 csv
16 csv
32 csv
64 csv
128 csv
256 csv
Thread Sorter - Sys
Fork - Sys

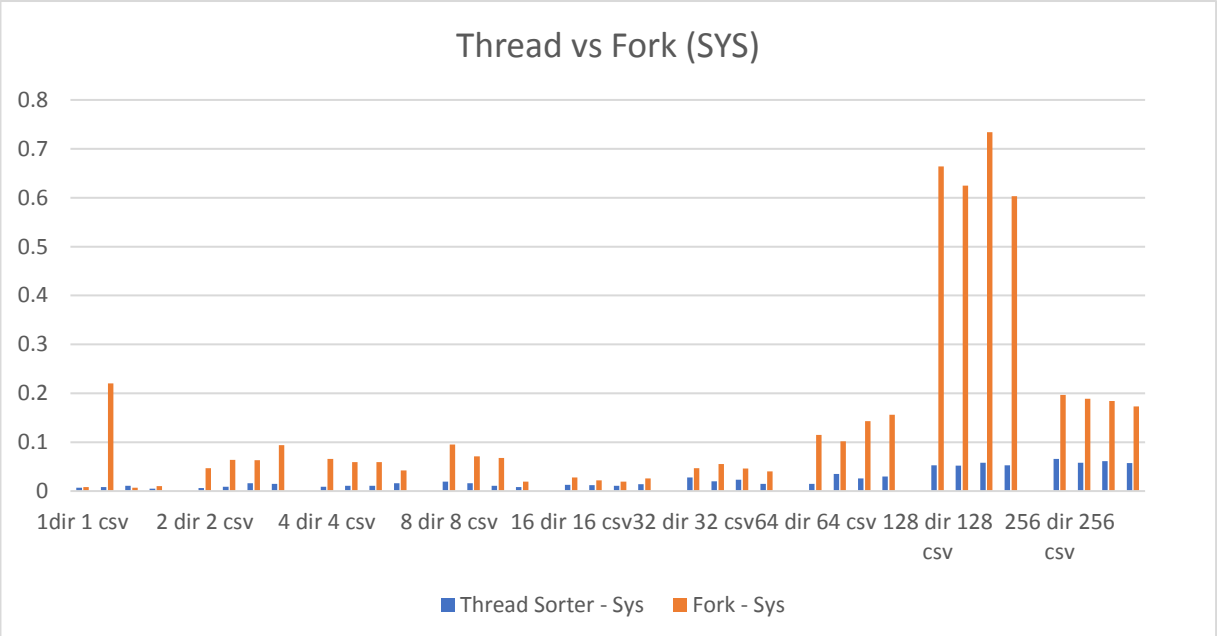

Thread vs Fork Real (2 CSV)
6
5
4
3
2
1
0
1 dir
2 dir
4 dir
8 dir
16 dir
32 dir
64 dir
128 dir
256 dir
Thread Sorter - Real
Fork - Real


Thread vs Fork User (2 CSV)
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1 dir
2 dir
4 dir
8 dir
16 dir
32 dir
64 dir
128 dir
256 dir
Thread Sorter - User
Fork - user


Thread vs Fork System (2 CSV)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1 dir
2 dir
4 dir
8 dir
16 dir
32 dir
64 dir
128 dir
256 dir
Thread Sorter - Sys
Fork - Sys


Thread Vs Fork (REAL)
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1dir 1 csv
2 dir 2 csv
4 dir 4 csv
8 dir 8 csv
16 dir 16 csv
32 dir 32 csv
64 dir 64 csv
128 dir 128 csv
256 dir 256 csv
Thread Sorter - Real
Fork - Real


Thread vs Fork (USR)
3.5
3
2.5
2
1.5
1
0.5
0
1dir 1 csv
2 dir 2 csv
4 dir 4 csv
8 dir 8 csv
16 dir 16 csv
32 dir 32 csv
64 dir 64 csv
128 dir 128 csv
256 dir 256 csv
Thread Sorter - User
Fork - user


Thread vs Fork (SYS)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1dir 1 csv
2 dir 2 csv
4 dir 4 csv
8 dir 8 csv
16 dir 16 csv
32 dir 32 csv
64 dir 64 csv
128 dir 128 csv
256 dir 256 csv
Thread Sorter - Sys
Fork - Sys

(1 dir)	Thread Sorter - Real	Thread Sorter - User	Thread Sorter - Sys
1 csv	2.895	2.848	0.011
	3.811	3.018	0.007
	3.176	2.865	0.008
	4.153	2.884	0.008
2 csv	1.1437	1.499	0.009
	1.451	1.513	0.007
	1.468	1.53	0.007
	1.457	1.534	0.008
4 csv	1.505	0.901	0.012
	1.258	0.916	0.012
	1.303	0.907	0.009
	1.029	0.896	0.012
8 csv	0.411	0.582	0.009
	0.251	0.647	0.009
	0.405	0.607	0.018
	0.428	0.631	0.015
16 csv	0.251	0.438	0.013
	0.26	0.448	0.02
	0.269	0.444	0.013
	0.463	0.456	0.014
32 csv	0.486	0.38	0.019
	0.464	0.385	0.023
	1.523	0.397	0.015
	0.463	0.387	0.01
64 csv	0.175	0.319	0.033
	0.163	0.305	0.049
	0.152	0.334	0.025
	0.164	0.315	0.025
128 csv	0.177	0.336	0.032
	0.149	0.313	0.042
	0.149	0.322	0.038
	0.157	0.323	0.045
256 csv	0.922	0.322	0.078
	0.418	0.326	0.067
	0.925	0.331	0.064
	0.824	0.334	0.062

	Thread Sorter - Real	Thread Sorter - User	Thread Sorter - Sys
1dir 1 csv	3.093	3.04	0.007
	3.022	2.932	0.008
	3.078	3.029	0.011
	2.9	2.866	0.005
2 dir 2 csv	1.474	1.531	0.006
	1.496	1.543	0.009
	1.733	1.525	0.016
	2.468	1.536	0.015
4 dir 4 csv	1.014	0.904	0.009
	2.349	0.935	0.011
	1.241	0.896	0.011
	1.022	0.936	0.016
8 dir 8 csv	1.191	0.629	0.019
	1.102	0.641	0.016
	0.693	0.654	0.011
	0.43	0.642	0.008
16 dir 16 cs	0.266	0.453	0.013
	0.279	0.465	0.012
	0.276	0.455	0.011
	0.259	0.438	0.014
32 dir 32 cs	0.26	0.357	0.028
	0.684	0.37	0.02
	0.223	0.363	0.023
	0.195	0.358	0.015
64 dir 64 cs	0.212	0.355	0.015
	0.198	0.306	0.035
	0.48	0.344	0.026
	0.525	0.343	0.03
128 dir 128	0.771	0.332	0.053
	0.477	0.34	0.052
	1.084	0.322	0.058
	0.529	0.329	0.053
256 dir 256	0.476	0.345	0.066
	0.439	0.344	0.058
	1.291	0.342	0.061
	0.171	0.334	0.057

(2 csv)	Thread Sorter - Real	Thread Sorter - User	Thread Sorter - Sys
1 dir	1.538	1.532	0.008
	1.476	1.527	0.009
	1.675	1.527	0.004
	1.469	1.538	0.002
2 dir	1.484	1.522	0.009
	1.458	1.516	0.008
	1.475	1.53	0.009
	1.448	1.507	0.009
4 dir	1.471	1.512	0.007
	1.758	1.547	0.015
	2.063	1.54	0.008
	2.909	1.533	0.018
8 dir	2.937	1.607	0.011
	1.81	1.592	0.017
	1.459	1.509	0.008
	1.62	1.505	0.007
16 dir	1.76	1.526	0.009
	1.455	1.514	0.008
	1.479	1.535	0.006
	1.498	1.571	0.007
32 dir	1.501	1.552	0.009
	1.47	1.54	0.009
	1.483	1.535	0.008
	1.46	1.505	0.01
64 dir	1.531	1.524	0.013
	1.481	1.526	0.011
	1.799	1.545	0.016
	1.794	1.562	0.02
128 dir	3.067	1.545	0.024
	2.222	1.52	0.026
	3.247	1.555	0.022
	2.549	1.544	0.024
256 dir	1.938	1.557	0.039
	1.643	1.539	0.026
	2.109	1.555	0.044
	2.214	1.55	0.022

(1 dir)	Fork - Real	Fork - user	Fork - Sys
1 csv	4.807	0.21	0.07
	4.007	0.284	0.051
	5.518	0.229	0.074
	6.252	0.217	0.081
2 csv	0.177	0.188	0.011
	2.597	0.244	0.065
	3.968	0.23	0.069
	3.509	0.246	0.058
4 csv	1.596	0.27	0.069
	1.4	0.257	0.069
	1.051	0.256	0.075
	0.915	0.233	0.07
8 csv	0.092	0.234	0.024
	0.064	0.274	0.018
	0.061	0.272	0.015
	0.059	0.279	0.019
16 csv	0.08	0.228	0.022
	0.061	0.262	0.02
	0.059	0.249	0.031
	0.062	0.273	0.014
32 csv	0.452	0.262	0.064
	0.408	0.266	0.073
	0.411	0.27	0.066
	0.403	0.258	0.064
64 csv	0.157	0.237	0.043
	0.08	0.26	0.034
	0.093	0.273	0.042
	0.082	0.279	0.025
128 csv	0.23	0.3	0.034
	0.121	0.284	0.052
	0.135	0.284	0.059
	0.112	0.292	0.045
256 csv	0.753	0.343	0.107
	0.432	0.322	0.117
	0.423	0.341	0.107
	0.423	0.33	0.116

	Fork - Real	Fork - user	Fork - Sys
1dir 1 csv	0.3	0.192	0.008
	0.223	0.192	0.22
	0.22	0.19	0.007
	0.219	0.184	0.01
2 dir 2 csv	2.793	0.25	0.047
	3.203	0.224	0.064
	2.053	0.232	0.063
	4.457	0.239	0.094
4 dir 4 csv	1.763	0.238	0.066
	1.088	0.265	0.059
	1.593	0.279	0.059
	1.063	0.288	0.042
8 dir 8 csv	1.168	0.236	0.095
	0.655	0.29	0.071
	0.681	0.303	0.068
	0.069	0.264	0.019
16 dir 16 csv	0.108	0.248	0.028
	0.069	0.261	0.022
	0.07	0.272	0.019
	0.064	0.256	0.026
32 dir 32 csv	0.16	0.247	0.047
	0.089	0.252	0.055
	0.093	0.251	0.046
	0.092	0.263	0.04
64 dir 64 csv	0.254	0.255	0.115
	0.135	0.268	0.102
	0.421	0.267	0.143
	0.905	0.267	0.156
128 dir 128 cs	0.67	0.244	0.664
	0.533	0.244	0.625
	0.625	0.236	0.734
	0.532	0.249	0.603
256 dir 256 cs	0.869	0.314	0.197
	0.493	0.314	0.189
	0.63	0.337	0.184
	0.247	0.285	0.173

(2 csv)	Fork - Real	Fork - user	Fork - Sys
1 dir	0.441	0.197	0.013
	0.365	0.191	0.007
	0.137	0.188	0.01
	0.123	0.191	0.01
2 dir	0.142	0.19	0.009
	0.12	0.19	0.009
	0.119	0.187	0.009
	0.147	0.195	0.008
4 dir	0.147	0.196	0.011
	0.131	0.189	0.009
	3.154	0.274	0.053
	2.336	0.225	0.062
8 dir	4.919	0.27	0.077
	3.316	0.257	0.075
	0.123	0.193	0.009
	0.126	0.203	0.004
16 dir	0.524	0.216	0.012
	0.208	0.186	0.012
	0.157	0.19	0.011
	0.126	0.19	0.013
32 dir	0.179	0.195	0.024
	0.137	0.187	0.022
	0.144	0.201	0.016
	0.144	0.201	0.018
64 dir	0.365	0.189	0.044
	0.268	0.187	0.039
	1.937	0.257	0.1
	3.267	0.226	0.083
128 dir	4.253	0.248	0.162
	2.609	0.24	0.154
	3.585	0.241	0.16
	5.676	0.245	0.152
256 dir	1.241	0.195	0.936
	1.362	0.192	0.877
	1.871	0.191	0.871
	2.011	0.205	0.905