Analysis of Two Phase sorting
2019121010

System Configuration:-

1. OS -> Linux Ubuntu
2. RAM -> 4GB
3. Hard Disk -> DDR3
4. 5 years old laptop :(

Sorting on col2 (but changing columns won't affect the timings)
Without threading (memory is fixed = 100MB)

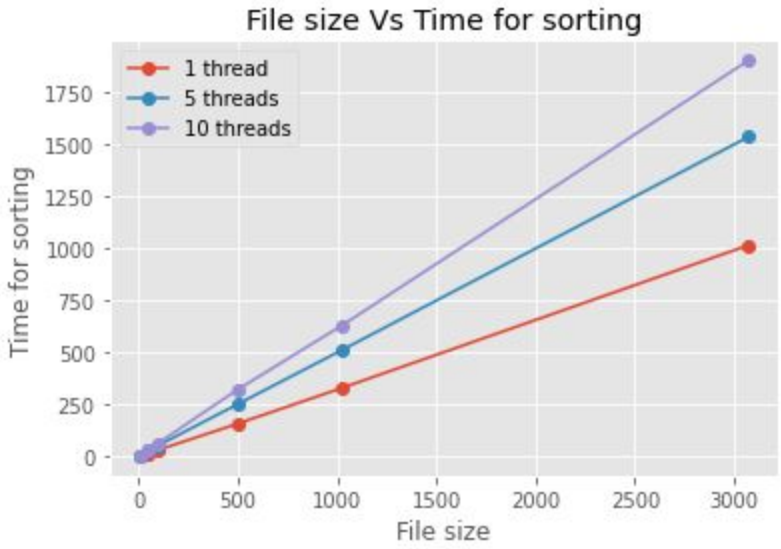
5MB	50MB	100MB	500MB	1GB	3GB
1.5	15	30	156	327	1012

With threading (5 threads, memory is fixed = 100MB)

5MB	50MB	100MB	500MB	1GB	3GB
2.65	25.4	51.6	250	509	1532

With threading (10 threads, memory is fixed = 100MB)

5MB	50MB	100MB	500MB	1GB	3GB
3	31	63	320	627	1899



Without threading (file size is fixed = 500MB)

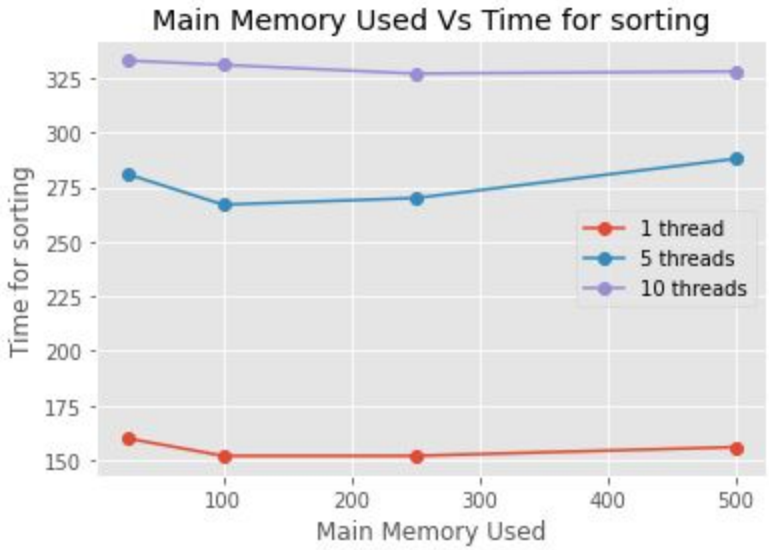
25MB	100MB	250MB	500MB
160	152	152	156

With threading (5 threads, file size is fixed = 500MB)

25MB	100MB	250MB	500MB
281	267	270	288

With threading (10 threads, file size is fixed = 500MB)

25MB	100MB	250MB	500MB
333	331	327	328



Explanations

In two phases sorting the first phase is used for fetching chunks of data into the main memory without exceeding it, sorting it and finally writing it back in temporary files.

The second phase uses the temporary files from the first phase and merges all of their contents according to the sorting criteria given as input(i.e asc or desc and columns list). We also check for feasibility of sorting by using the principle that the maximum number of files from which the main memory can read is M/R (M = main memory and R = record size).

Using the above theory we can analyse the above graphs:-

1. One of the main observations is that using more threads increases the time rather than decreasing it.

This is due to the creation of new input files from the original file, so that data can be divided equally between the threads and threads work in isolation. Since the creation of files and reading from multiple files is time consuming it leads to increase in time.

2. The first graph is almost linear for any number of threads. This is due to the fact that the number of disk accesses increases almost linearly with increase in the input file size (when the main memory is fixed).
3. The second graph is almost a constant line parallel to x-axis. When main memory is increased it reduces the number of temporary files that need to be created in phase one and merged in phase two. But the data to be sorted also reduces when main memory is reduced. These two opposing factors almost balance each other out, hence we see almost constant lines.