Two Phase Merge Sort
Analysis

Name : Naman Jain **Roll No : 2020201080**

System Configuration

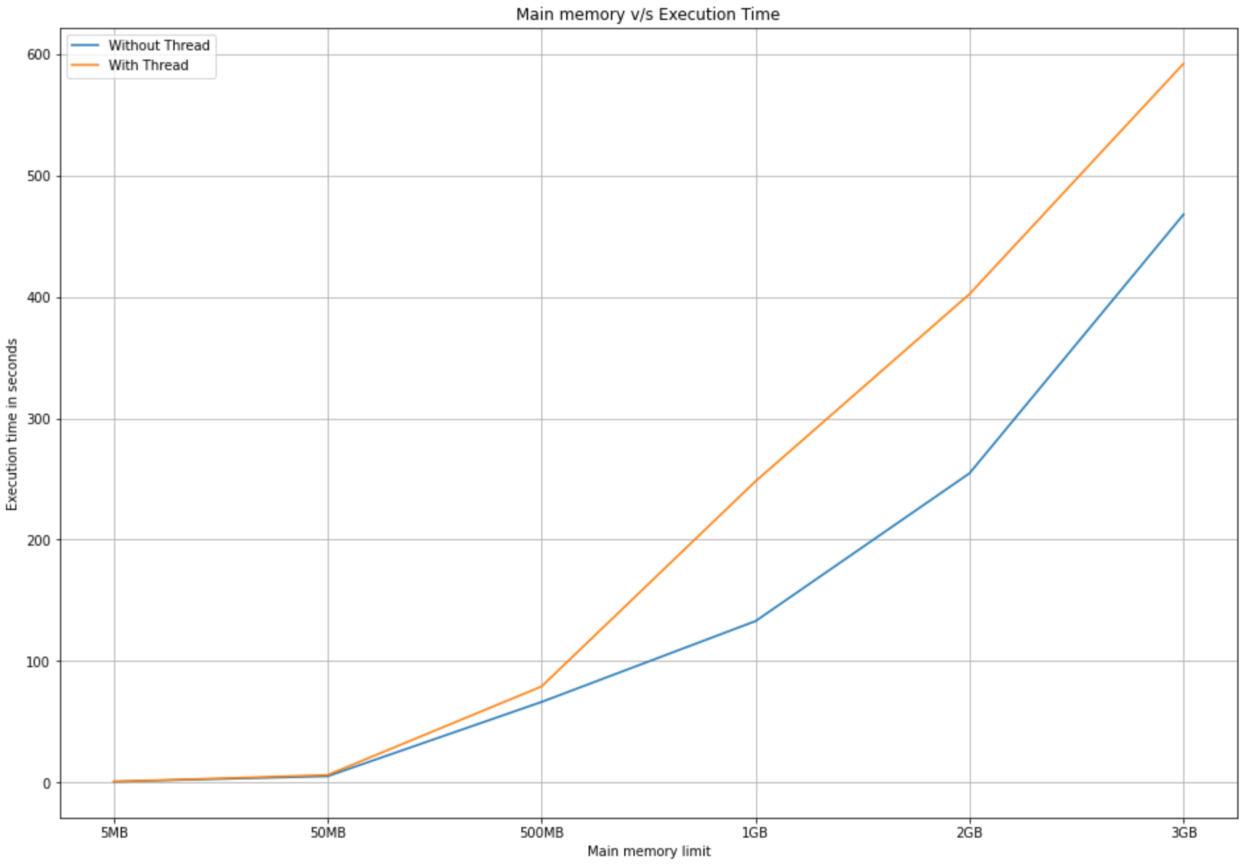
- ★ 8 GB DDR4 Ram
- ★ Intel i5 7th gen Processor
- ★ NVIDIA GeForce 940MX Graphic card
- ★ 100 GB Memory (allocated to Ubuntu)
- ★ Clock speed 2.5 Ghz
- ★ Cache 3 MB

Observations

Observation 1 : Varying FileSize with constant memory

Input File Size	Execution Time without using Threads (in seconds)	Execution Time using Threads (in seconds) (Thread Count = 5)
5 MB	0.62	0.61
50 MB	5.05	6.09
500 MB	66.30	79.05
1 GB	132.95	248.22
2 GB	254.82	402.56
3 GB	468.20	592.48

- ➔ Memory Limit Taken : 100 MB
- ➔ Sorting Order : Ascending
- ➔ Columns (On which sorting applied) : C3 C1



Inference:

Above graph depicts that, Execution time of an algorithm without using threads is faster than that with using threads. Following are some reasons to justify this observation :

- Creation of threads results in increasing the no. of chunks (sublist) created. Due to formation of more chunks, read/write operations on a file increases and hence results in increasing the overall execution time.
- Also, introducing multiple threads in a single processor system increases the no. of context switches which also increases the execution time of the algorithm.

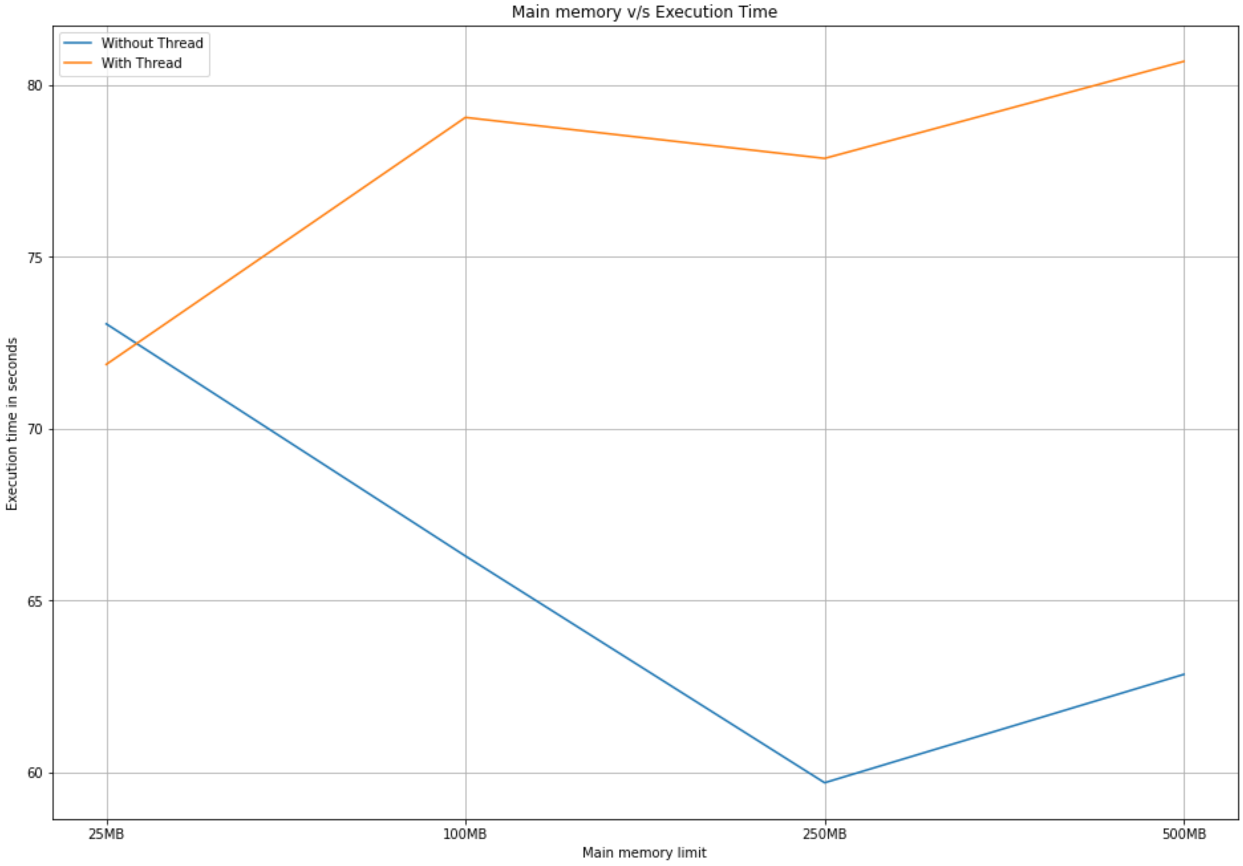
Note:

Since my computer runs on a single processor, I couldn’t utilize the power of parallel computation using multithreading.

Observation 2 : Varying memory with constant FileSize

Main Memory Size	Execution Time without using Threads (in seconds)	Execution Time using Threads (in seconds) (Thread Count = 5)
25 MB	73.05	71.87
100 MB	66.30	79.05
250 MB	59.07	77.86
500 MB	62.86	80.68

- ➔ File Size Taken : 500 MB
- ➔ Sorting Order : Ascending
- ➔ Columns (On which sorting applied) : C3 C1



Inference:

Above graph depicts that,

- Execution time of an algorithm without using threads is still faster than that with using threads. However, with less main memory size, my algorithm takes less time by using multithreading.

- On increasing main memory size, Execution time (without using threads) first decreases to 59 sec (uptill memory size equivalent to 250 MB), but after that, it again starts increasing.

- However, the above graph for a multithreaded algorithm shows a gradual rise in execution time from 73 sec to 82 sec.

.

Note:

Since my computer runs on a single processor, I couldn't utilize the power of parallel computation using multithreading.