# Two Phase Merge Sort Analysis (Mayank Musaddi 20171115)

- ## System Configuration

  Processor : Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
  Graphics : GeForce 940MX/PCIe/SSE2
  Memory : 7.7 GiB
  Disk : 202.4 GB
  System : Ubuntu 18.04.5 LTS 64-bit

- ## Observations

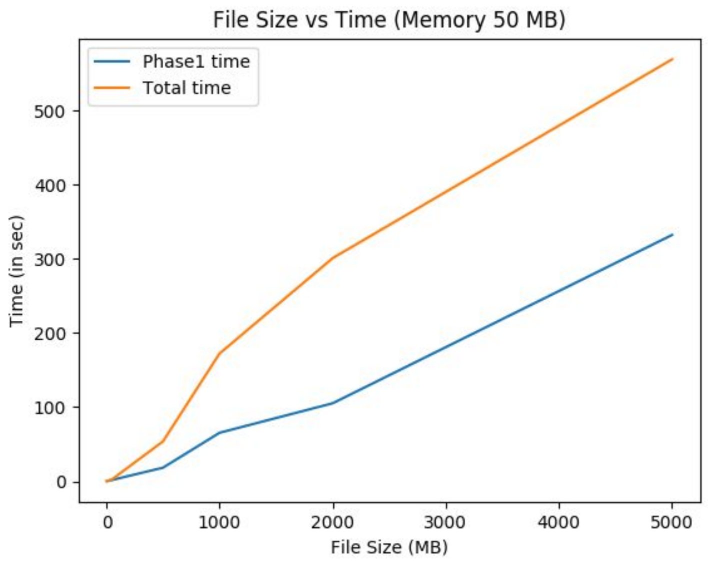
  The Time taken for different configurations were noted by varying the file sizes to be sorted, the memory limits and the number of threads used during the sort.

  The plots and the observations are noted as follows:

  1) Fixed File Size (MB) vs Time (seconds)
     (Memory Allocated is 50MB)

| File Size (MB) | Number of Records | Phase 1 Time | Total Time |
|---|---|---|---|
| 5 | 50,000 | 0.179 | 0.277 |
| 50 | 500,000 | 1.740 | 2.653 |
| 500 | 5,000,000 | 18.317 | 53.806 |
| 1000 | 10,000,000 | 65.428 | 172.314 |
| 2000 | 20,000,000 | 105.211 | 301.023 |
| 3000 | 30,000,000 | 332.301 | 569.203 |

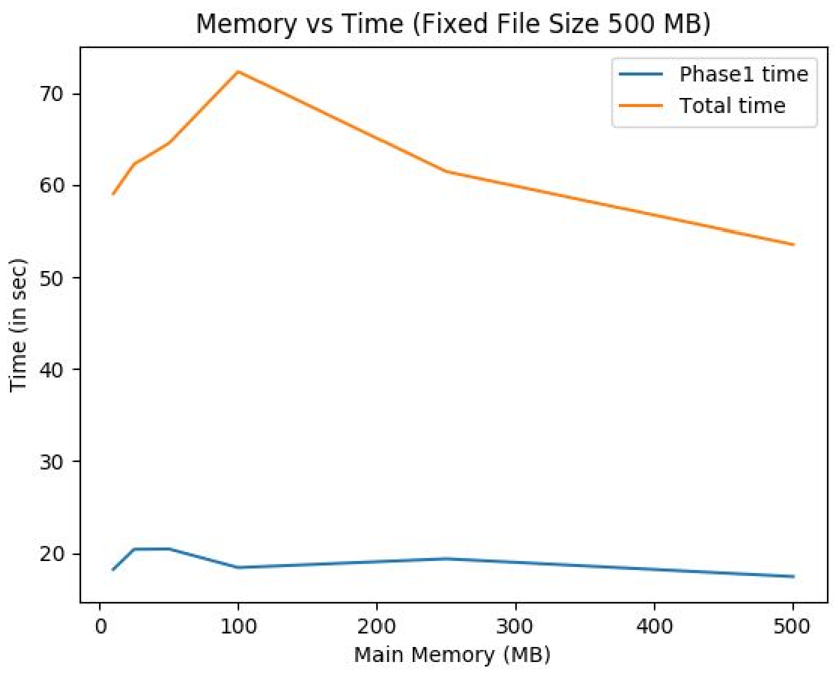
Graph:



2) Main Memory (MB) vs Time (seconds)
(File Size is 500MB)

| Main Memory (MB) | Phase 1 Time | Total Time |
|---|---|---|
| 10 | 18.213 | 59.032 |
| 25 | 20.395 | 62.245 |
| 50 | 20.429 | 64.536 |
| 100 | 18.402 | 72.325 |
| 250 | 19.362 | 61.432 |
| 500 | 17.422 | 53.534 |

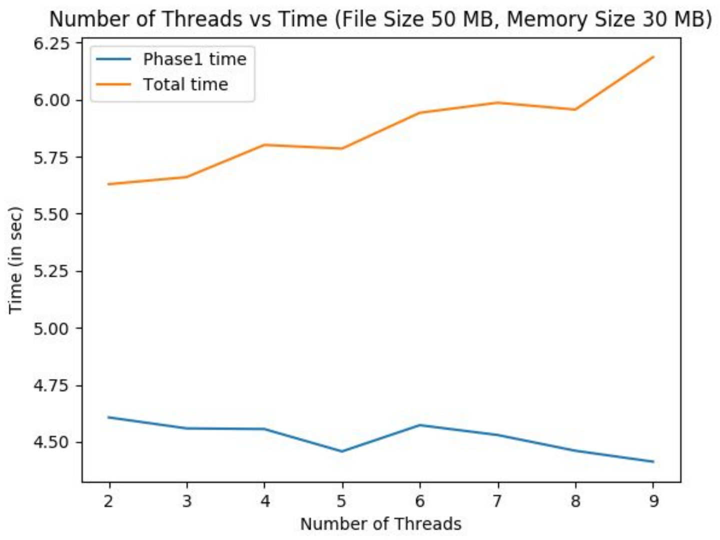
Graph:



3) Number of threads vs Time (seconds)
(Memory Allocated is 50MB and File size is 50MB)

| Number of Threads | Phase 1 Time | Total Time |
|---|---|---|
| 2 | 4.607 | 5.629 |
| 3 | 4.559 | 5.660 |
| 4 | 4.556 | 5.801 |
| 5 | 4.458 | 5.785 |
| 6 | 4.573 | 5.942 |
| 7 | 4.530 | 5.986 |
| 8 | 4.461 | 5.956 |
| 9 | 4.413 | 6.186 |

Graph:



- ## Explanations

Execution Time Data was noted with the use of python time library. Although this time is affected by parametric changes as evident in graphs, there are a lot of noise in the data involved due to other processes running in the OS and system constraints.

**Graph 1:**
We find that as we increase the file size, the time taken for the sort also increases in a somewhat exponential fashion as with the increase in size the number of files created, deleted are also increased apart from the time taken by the usual sort of extra records.

**Graph 2:**
Execution time decreases with the increase in memory size as should be. This is because there are lesser sublists that are created, hence lesser file reads and writes. However initially till the increase of memory till 100 MB the time increases. This increase is only evident in phase 2 and not in phase 1. To investigate it further, time taken for different parts in phase 2 like file reads, heap sort and disk writes were scrutinized. While there was a decrease in file read time, there was an evident increase in sorting which can be attributed to the python heapq module functioning. Hence execution time saw an overall decrease while increasing at the beginning.

**Graph 3:**
As the number of threads are increased we find the total time increasing. However we also find that the part that has been parallelised i.e. the phase 1 has its time decreasing. This is quite justifiable because the increase in time in phase 2 due to extra reads, overpowers the decrease in time in phase 1 which is not so significant due to thread creation time.