Programming Assignment-1
COEN 242, BiG Data

Syeda Gousia Sultana

Due date: 04/26/2020

Problem Statement:

Design and Implement an efficient java/python/scala code (or any language you prefer) to determine Top K most frequent/repeated words in a given dataset (example: K = 10). The objective here is to obtain the result with the least possible execution Time (or with the best performance on your computer).

Execute your code on each of the three input data files separately. It is a good practice to execute your code initially on 1 GB dataset (or any small text file) and then repeat on larger datasets.

Analyze the performance through different metrics such as running time, speedup, CPU utilization, memory usage, etc. Or You can come up with any metrics that make sense for this application/program.

Present a detailed analysis of why you use a particular algorithm or a particular data structure to solve this problem.

Implementation:

This report contains the results of execution of the code which has been implemented separately for each of the three given datasets. The plotted graph shows the performance in terms of Memory_Percent and the number of threads utilized, time vs number of threads utilized, system_time vs number of threads utilized. It compares physical system memory to process resident memory (RSS), calculates process memory utilization in terms of percentage, CPU time taken to run the entire process and separate analysis of the system time.

We have designed and implemented an efficient python code to determine top K most frequent words in a given dataset using the concept of parallelism via threading. We have targeted towards obtaining the result with the least possible execution time and have come up with the best approach for efficient performance on our computers.

By trying to implement various approaches we have come to the conclusion that using multiple threads is the best approach. We have chosen multithreading over multiprocessing as the task is more I/O intensive than CPU intensive. We have implemented parallelism using the concurrent.futures module which launches parallel threads. It is used in this code to provide a high-level interface for

asynchronously executing callables. This asynchronous execution is performed using the ThreadPoolExecutor in which task is distributed between a pool of threads.

During the testing phase we have seen an improvement in performance by increasing the number of threads, however we experienced a performance drop in each of the cases when the number of threads increased beyond an optimum level. Since we have implemented the code in python, a typical python program runs on a single processor and the threads are executed in the same processor so when increasing the thread beyond a limit, the system time increase due to the context switching between the threads within a single processor

We have implemented the task with multiple functions. The 'read_file' method reads the file in chunks by using the itertools.islice module and uses ThreadPoolExecutor which distributes the chunks among threads and the threads process the chunks in parallel thus improving the efficiency. The 'count_words' process each chunk and count the words. We use a global dictionary to store the word and the count. We use a regex for filtering out words with characters in the english alphabet.

We also did some research with multiprocessing, however we cannot implement a global dictionary to store the words as each process uses different memory space unlike threads and hence we have to combine all the temporary dictionaries to a common dictionary in the end. Thus we decided to implement the project with multithreading. We also did some experiments by varying the size of data read, we were able to see some changes with small files when size was varied between 5k-35K(lines). Like the threads there was an optimum value in between but it was not practical to do such testing with 8GB and 32GB files so we have done the testing by making the chunk size to a constant of 35K lines.

Presenting Performance Data :

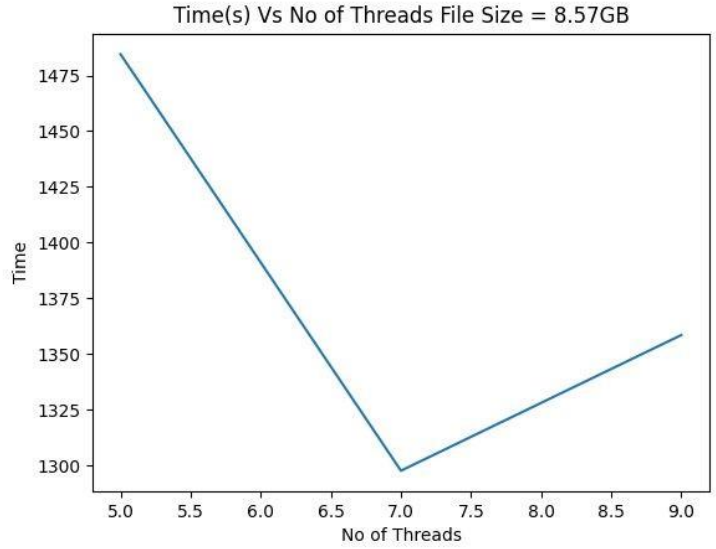
The system used for testing:

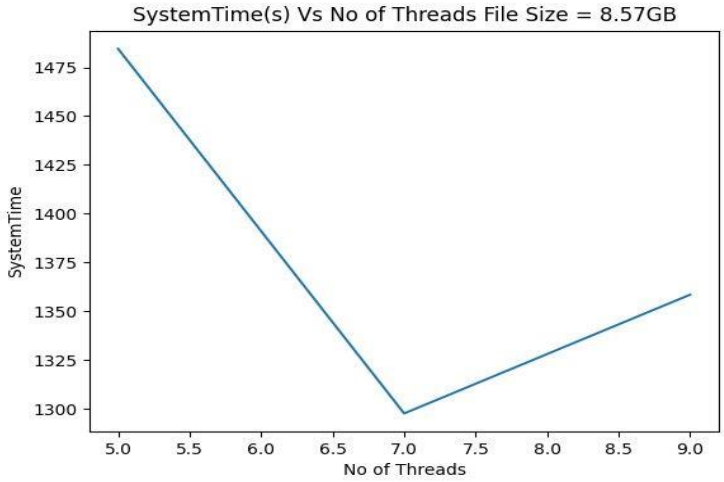
Model Name: MacBook Air Processor Name: Intel Core i5
Processor Speed: 1.8 GHz
Number of Processors: 1
Total Number of Cores: 2
L2 Cache (per Core): 256 KB L3 Cache: 3 MB Hyper-Threading Technology: Enabled
Memory: 8 GB

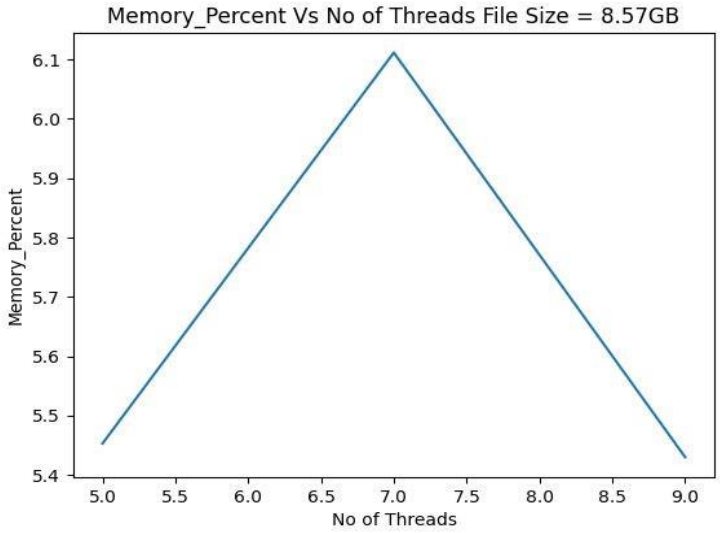
File Size: 8.57GB

The data used for plotting the graph:

No of Threads	CPU time(sec)	Mem Percent
5	User=1467.24749312, System=17.301948416 Total time = 1484.54944154	5.453062057495117 rss=468414464
7	User=1289.709027328, System=7.89744384 Total time =1297.60647117	6.111574172973633 rss=524980224
9	User=1348.36584448, System=10.142390272 Total time= 1358.50823475	5.430078506469727 rss=466440192


Time(s) Vs No of Threads File Size = 8.57GB
Time
No of Threads
1475
1450
1425
1400
1375
1350
1325
1300
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0

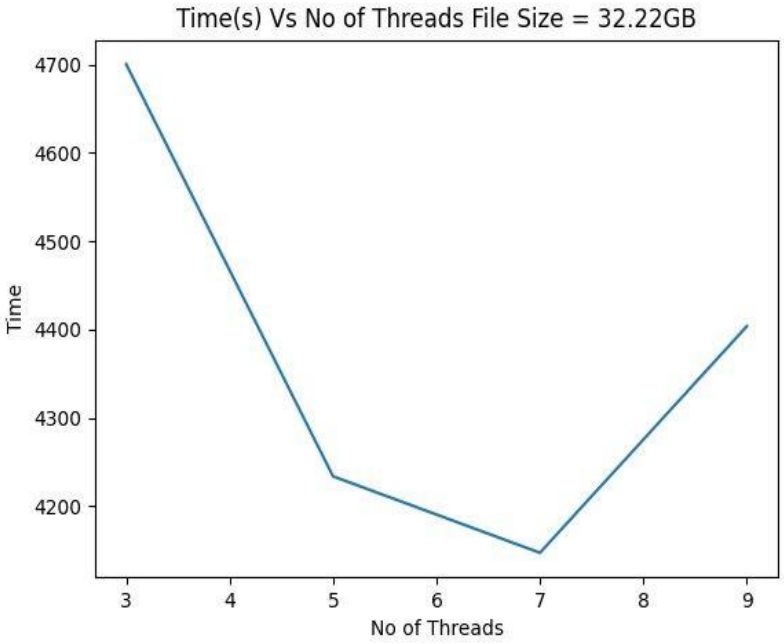

SystemTime(s) Vs No of Threads File Size = 8.57GB
SystemTime
No of Threads
1475
1450
1425
1400
1375
1350
1325
1300
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0

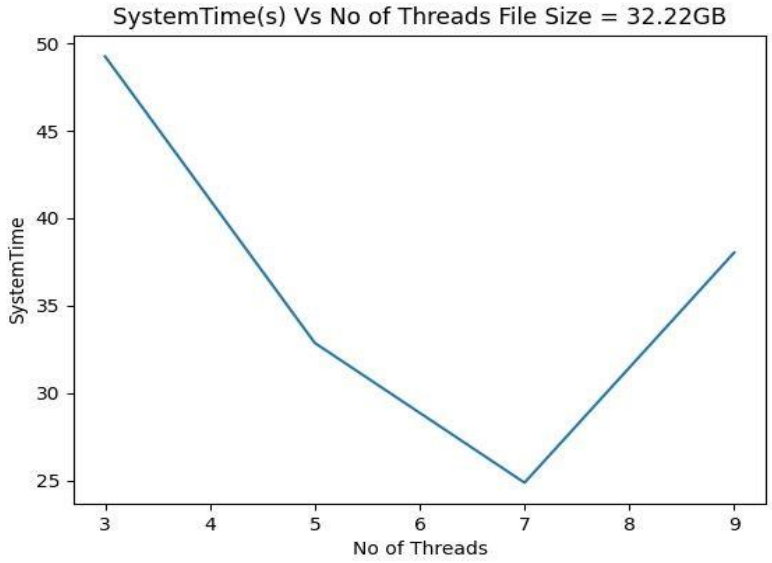


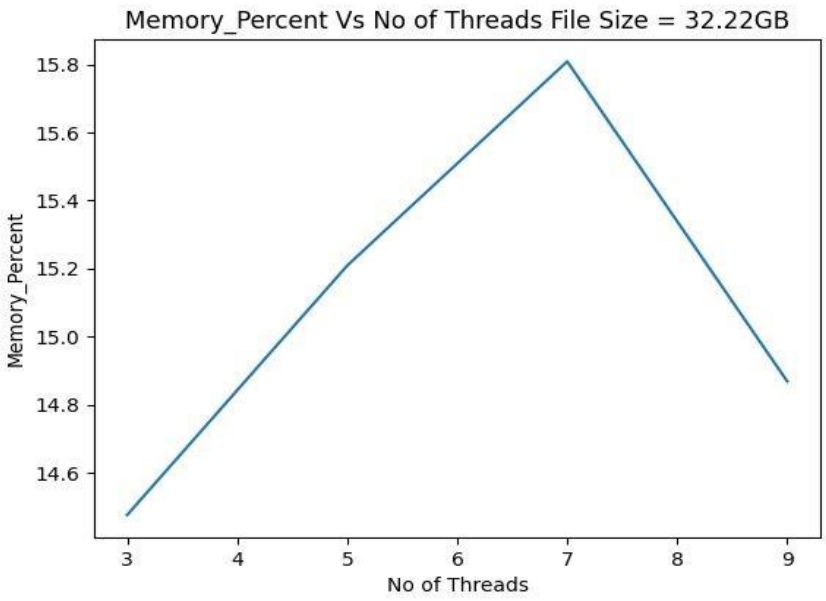
File Size: 32.22GB

The data used for plotting the graph:

No of Threads	CPU time(sec)	Mem Percent
3	User=4651.155456, System=49.26195712 Total time = 4700.41741312	14.476823806762695 rss=1243549696
5	User=4200.804384768, System=32.862222336 Total time= 4233.6666071	15.209102630615234 rss=1306451968
7	User=4122.201554944, System=24.862533632 Total time = 4147.06408858	15.808629989624023 rss=1357950976
9	User=4365.58495744, System=38.0357632 Total time=4403.62072064	14.869117736816406 rss=1277247488


Time(s) Vs No of Threads File Size = 32.22GB
Time
4700
4600
4500
4400
4300
4200
3
4
5
6
7
8
9
No of Threads


SystemTime(s) Vs No of Threads File Size = 32.22GB
SystemTime
50
45
40
35
30
25
3
4
5
6
7
8
9
No of Threads

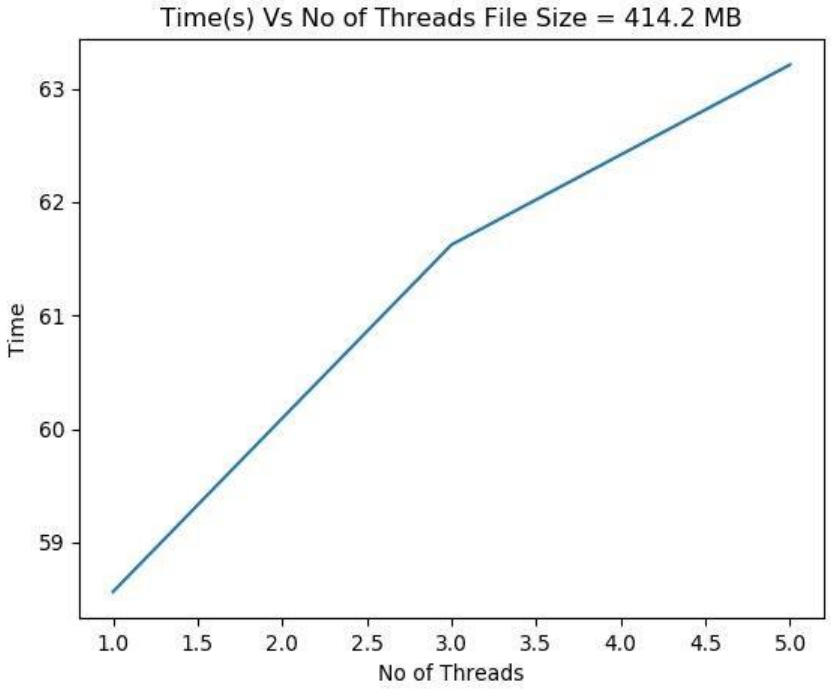


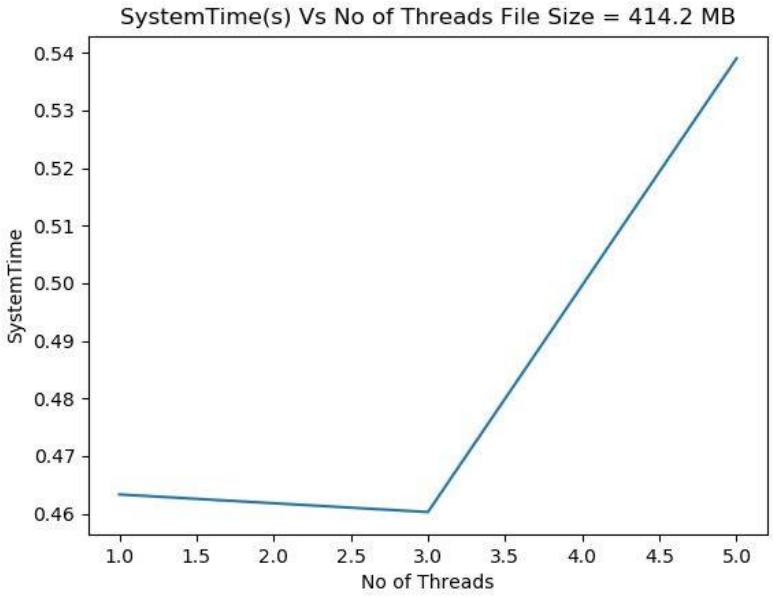
File Size: 400MB

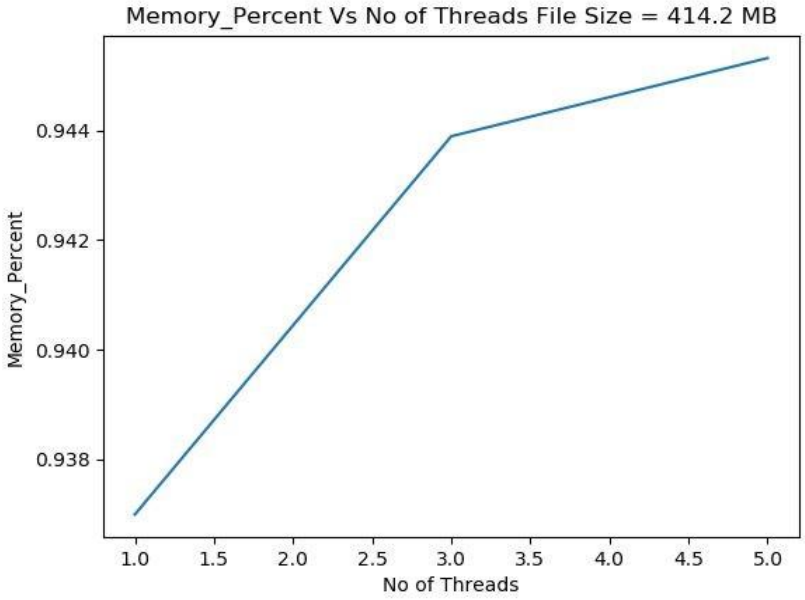
The data used for testing:

No of Threads	CPU time(sec)	Mem Percent
1	User=58.102652928, System=0.463320704 Total time = 58.565973632	0.9369850158691406 rss=80486400
3	User=61.163094016, System=0.460268288 Total time = 61.623362304	0.9438991546630859 rss=81080320

5	User=62.669971456, System=0.539018176 Total time = 63.208989632	0.9453296661376953 rss=81203200




SystemTime(s) Vs No of Threads File Size = 414.2 MB
SystemTime
0.54
0.53
0.52
0.51
0.50
0.49
0.48
0.47
0.46
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
No of Threads


Memory_Percent Vs No of Threads File Size = 414.2 MB
Memory_Percent
0.944
0.942
0.940
0.938
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
No of Threads