COMP90024 – Cluster and Cloud Computing Assignment 1

The Happiest City

Arnav Garg 1248298

Piyush Bhandula 1163716

Objective

Read a Twitter Dataset consisting of tweets from a large geographical area, filter the tweets based on the grid/mesh file containing Melbourne's different areas and calculate the sentiment score according to a dictionary provided, to find the happiest/most miserable areas in Melbourne.

Approach

1. Read the file names from the system arguments
2. Using mpi4py, initialize the parallel threads
3. Load the sentiment score and Melbourne grid file in memory
4. Parse the file on the different threads generated by mpi4py, where each thread will read every r^{th} tweet and skip the remaining. (r = rank of thread generated by mpi4py)
5. Check whether tweet lies in range; and capture the grid number, and increment the tweet counter for that grid and add the sentiment score to the counter if matched
6. Loop exits when the last json line is read, identified by *"]}"*
7. The master thread waits for the data from other threads, which share the tweet count and sentiment score using mpi4py *".send()"*
8. After receiving data, the master thread adds the score from each thread and prints it
9. Execute the code for the respective scenarios

Execution Steps

The script containing the python code is main.py. It takes 3 files as inputs in the following format:

srun -n <total_number_of_threads> python3 main.py <twitter_data_file_name> <area_file_name> <sentiment_analysis_keywords_with_score>

e.g. srun -n 8 python3 main.py bigTwitter.json melbGrid.json AFINN.txt

Results

The following run times were observed while processing the bigTwitter.json file:

Total Number of Nodes	Number of Threads on each Node	Execution Time
1	1	991.4 seconds
2	4	193.1 seconds
1	8	189.6 seconds

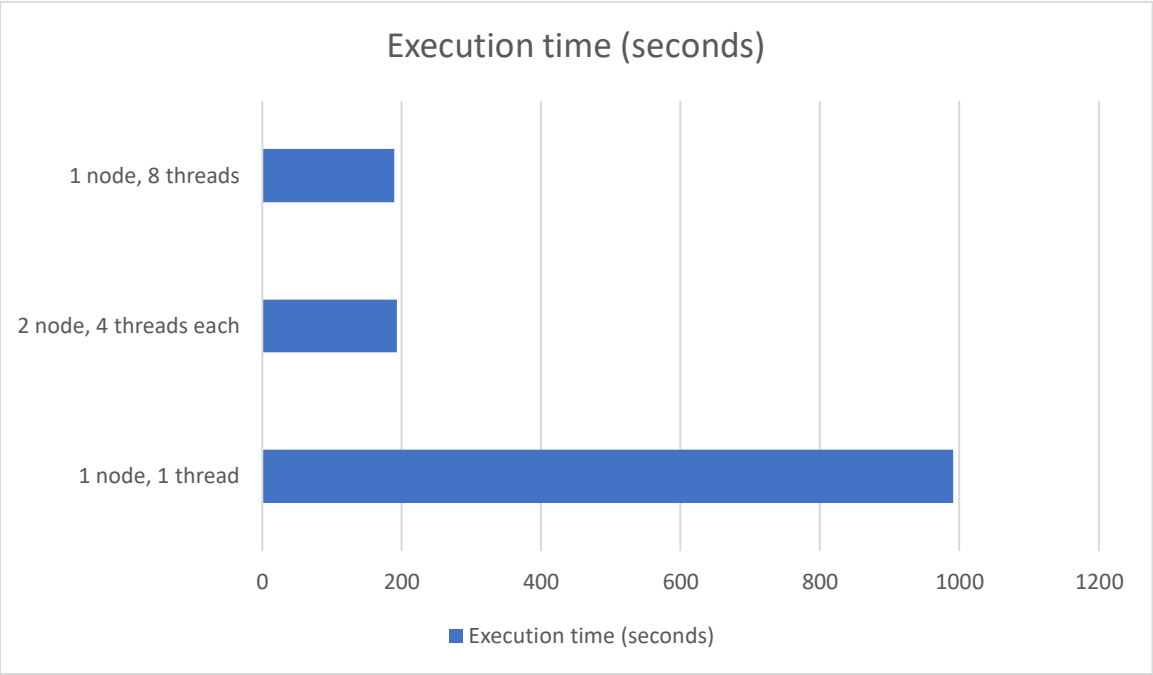

Figure 1: Bar Graph comparing execution times

Conclusion

We observed almost 4x performance improvement when running the core parallelly on 8 threads as compared to running on a single thread.

The execution times for “1 node, 8 threads” and “2 nodes, 4 threads each” is almost similar.

Hence, on increasing the processing power, we can improve the run time.

Appendix

```
[argarg@spartan-login3 ~]$ cat slurm-24796650.out
Area    Sentiment   Tweets
A1    : 763         2752
A2    : 4116        4904
A3    : 2679        5824
A4    : 54          381
B1    : 11614       21232
B2    : 32061       107386
B3    : 20211       34494
B4    : 5733        6643
C1    : 7551        10530
C2    : 191791      246828
C3    : 41434       69901
C4    : 19537       26097
C1    : 7551        5581
D3    : 7777        16220
D4    : 9698        16536
D5    : 3757        4705
Total : 361428      580014
--- 991.4585118293762 seconds ---
```

Figure 2: Output on running code on 1 node, 1 thread

```
[argarg@spartan-login3 ~]$ cat slurm-24796463.out
Area    Sentiment   Tweets
A1    : 763         2752
A2    : 4116        4904
A3    : 2679        5824
A4    : 54          381
B1    : 11614       21232
B2    : 32061       107386
B3    : 20211       34494
B4    : 5733        6643
C1    : 7551        10530
C2    : 191791      246828
C3    : 41434       69901
C4    : 19537       26097
C1    : 7551        5581
D3    : 7777        16220
D4    : 9698        16536
D5    : 3757        4705
Total : 361428      580014
--- 193.14605355262756 seconds ---
[argarg@spartan-login3 ~]$
```

Figure 3: Output on running code on 2 nodes, 4 thread each

```
[argarg@spartan-login3 ~]$ cat slurm-24796475.out
Area    Sentiment   Tweets
A1    : 763         2752
A2    : 4116        4904
A3    : 2679        5824
A4    : 77          381
B1    : 11614       21232
B2    : 32061       107386
B3    : 20211       34494
B4    : 5733        6643
C1    : 7551        10530
C2    : 191791      246828
C3    : 41434       69901
C4    : 19537       26097
C1    : 7551        5581
D3    : 7777        16220
D4    : 9698        16536
D5    : 3757        4705
Total : 361451      580014
--- 189.60413789749146 seconds ---
[argarg@spartan-login3 ~]$
```

Figure 4: Output on running code on 1 node, 8 threads