**COMP90024 – Cluster and Cloud Computing Assignment 1**

**The Happiest City**

Arnav Garg <Student ID>

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 rth 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 | 8 | 991.4 seconds |
| 2 | 4 | 193.1 seconds |
| 1 | 1 | 189.6 seconds |

**Figure 1: Bar Graph comparing execution times**

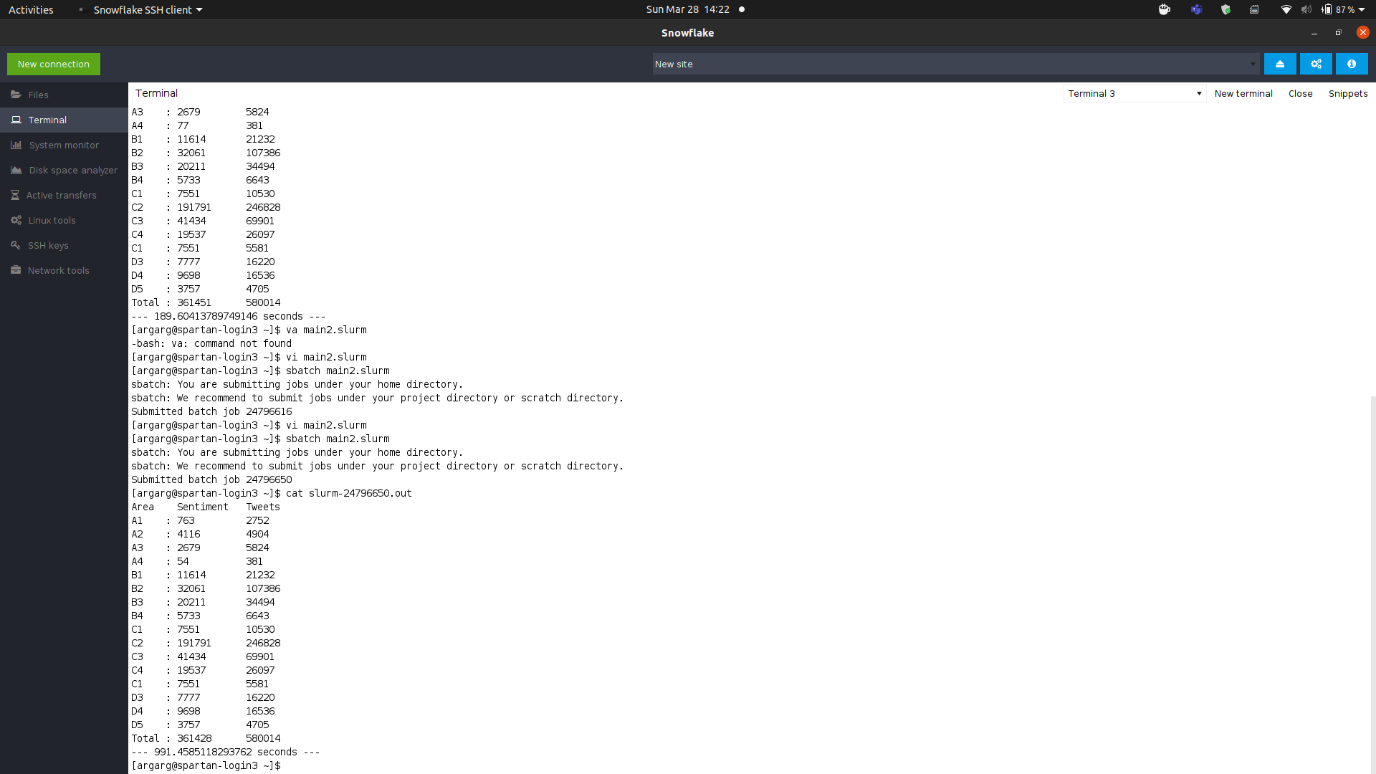
**Conclusion**

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

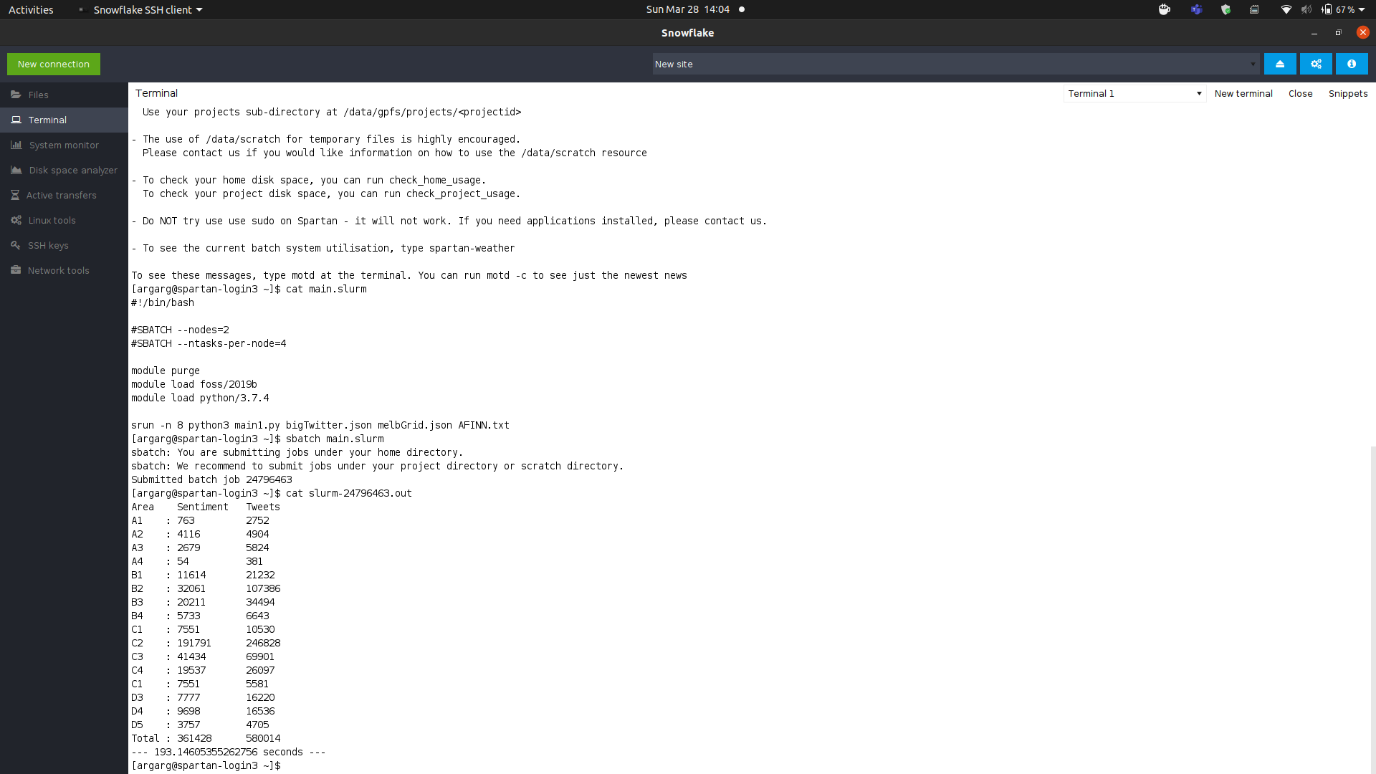
The execution times for “1 node, 8 threads” and “2 nodes, 4 threads each” is almost similar.

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

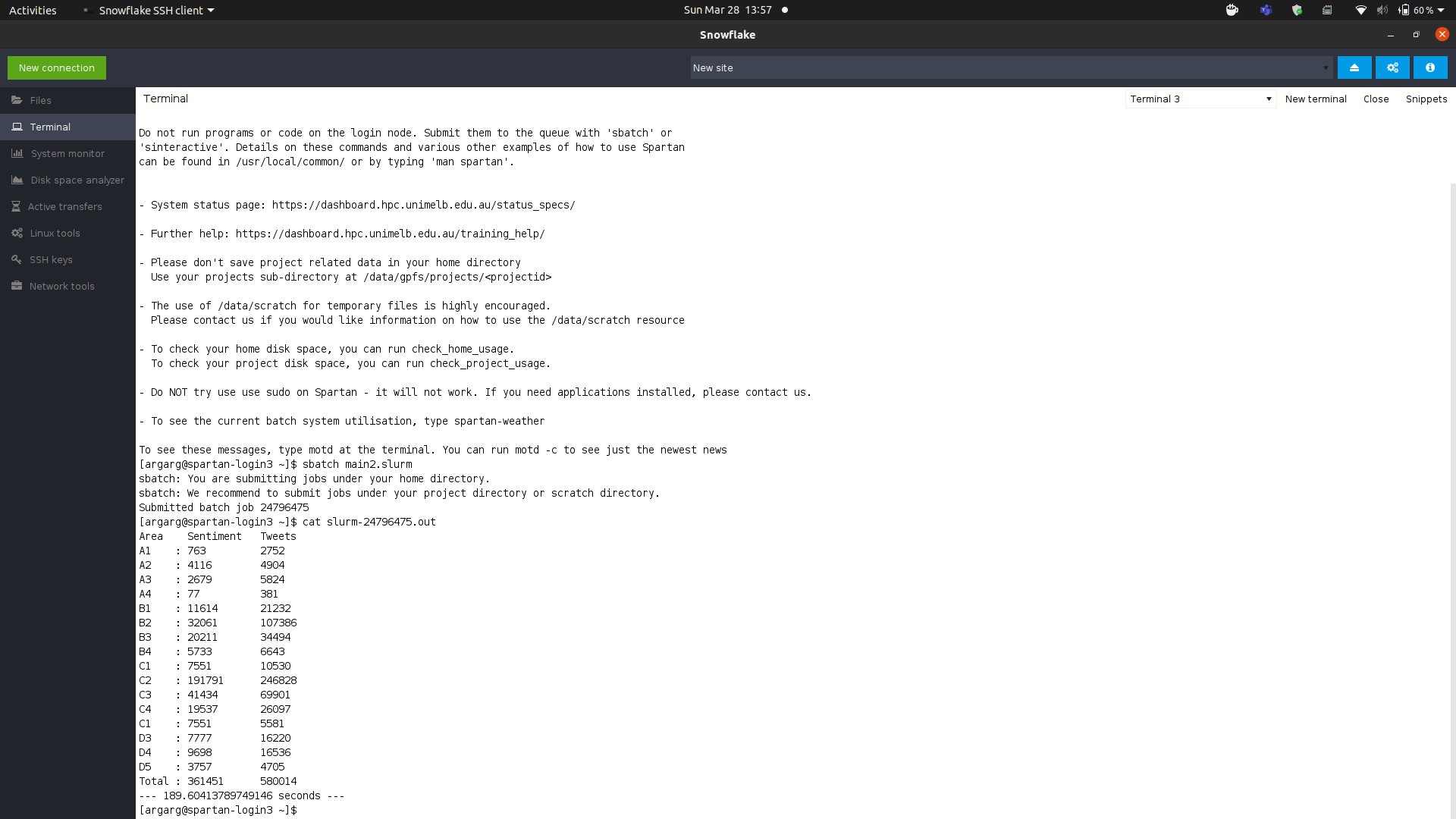
**Appendix**



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

****

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

****

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