Cluster and Cloud Computing
COMP90024 Assignment 1

Completed by:

Junhong Liu - 1084997

Soham Swapnil Dighe - 1219439

How to invoke

To run the code on the SPARTAN cluster, follow these steps

1. Make sure all JSON data files such as the twitter data, .py, and slurm scripts are in the same directory.
    a. Note that the bigTwitter file was too large to be stored in a git hub repo and needs to be manually placed in the same directory as the other files
2. Run the desired slurm script using command “sbatch <insert slurm script name here>” where the name of the script follows the format “test_n<number of nodes>c<total number of cores used>”
3. Outputs will then be found in an output folder. Each output will be prefixed by the same format that the slurm names follow

Slurm Scripts

The slurm scripts all contain very similar content. Since the scope of the task is relatively small, the timer set is a modest 10 minutes, just in case any unforeseen situations cause our code to fail to terminate. However, it is expected that, for a reasonable file size, our program should terminate well within this 10 minute margin. We also output any output, including error messages, to a folder in order to display our results and catch and exceptions. Furthermore, we specifically load a module of python that guarantees functionality for the functions and methods used within our program (python 3.7 and above).

Next, the actual command we run invokes the main.py and provides it the dataset we want it to process along with the file containing the supported language codes. As mentioned before, all the files that support and provide data to the program is expected to be contained at the same level in the same directory as main.py.

Approach

To begin, our program sets up the grid map of Sydney along with the language codes provided. Storing the grid as a list of 16 dictionaries, each containing the maximum and minimum latitude and longitude for a given cell. The code then enters the parallelisation process.

Since the data we had to process was all contained within a single file, the obvious way to parallelize the code was to split the file into multiple chunks to be processed in parallel by multiple processes. We make use of the Master Worker/Slave model to achieve this. The file was split into chunks of a certain size (except for the last chunk, which could be smaller than the others) and one chunk was assigned per core for further processing. Each process then performed the same actions on their given chunk, extracting and calculating the statistics for each cell and language of the. These partial results were then gathered back up by the Master processor (RANK = 0) and used to calculate the final metrics needed for the assignment.

Variations in performance

The specifications for this assignment asked us to run our code in three different settings on the SPARTAN cluster.

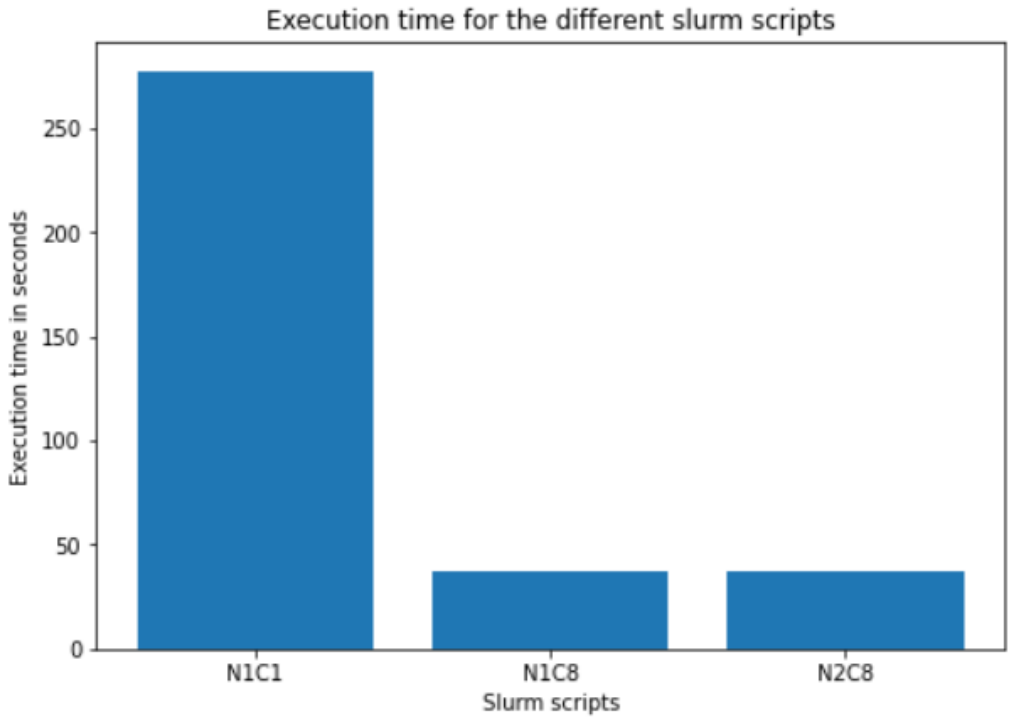
- 1 Node 1 Core
- 1 Node 8 Cores
- 2 Node 8 Cores (4 cores on each node)

The final results for the three different settings were as follows:

Setting	Execution time (seconds)
1 Node 1 Core	277.502371
1 Node 8 Cores	37.546311
2 Nodes 8 Cores	37.092691

These results are also graphed in the figure below.

As can be seen, there is a massive decrease in execution time between the first and second setting. In fact, the second setting is approximately 7.39 times faster than the first. This was expected as a large portion of our code is parallelisable, so introducing more cores should have a large effect on the time saved during execution. The fact that it is not precisely 8 times faster despite having 8 times as many cores is due to the small but significant parts of the code that is still sequential (notably, the final results tabulation). This part of the code acts as a lower bound on the minimum time it takes our program to run, no matter how many cores we introduce.



There is a strange result in the results as we expected the third setting to be slightly slower than the second. This is because the two settings have the same number of cores but the third setting has the extra communication overhead associated with separate nodes. Therefore, it was expected that their performance would be similar but with the second setting being slightly faster due to the lower communication overhead. However, this was not the case empirically. This deviation from our expectations could be attributed to the variance in execution time each time the same code is run on the SPARTAN cluster. We maintain that, on average, the second setting, using one node, should be faster than the third setting which uses the same amount of cores across two nodes.