COMP90024 Assignment 1 Report

Duc Trung Nguyen (1069760), Mayank Sharma (936970)

1 Introduction

The objective of this assignment is to implement a parallelised application that uses the HPC SPARTAN facility. The idea is to identify the top 10 most occurring hashtags and the most used languages for tweeting. Identifying these will give us a better understanding of how our application performs on the HPC SPARTAN facility and obtain more knowledge of benchmarking of applications in high-performance computing.

2 Application

2.1 Outline

Our entire application was run with Python module **3.6.4-intel-2017.u2-GCC-6.2.0-CUDA9**. The main library utilised for our parallel configurations was Message Passing Interface (MPI) standard library. Other extra libraries used that are part of Python module 3.6.4 were **collections** from **Counter**, **re** for regular expression, **os** for system dependent functionalities, **sys** and **getopt** for command line argument parsing and finally **json** and **time** for JSON parsing and time calculation. Options were used in our slurm file such as **--nodes** (to assign the number of nodes), **--ntasks-per-node** (to assign the number of processes to each node), **--job-name** and **--time**. Other options were also used like **--partition** (allow fast communication between nodes) and **--constraint** (used to constraint our job to specific processors to obtain consistent performance and time values).

2.2 Our code

Our code consists of the following files:

- **TwitterAnalyzer.py** - Our main application that can run for a single node mode or multiple node mode.
- **twitter_one_node_one_core.slurm** – Each communicator contains only one processor in allocated node.
- **twitter_one_node_eight_cores.slurm** – All processors are in the same allocated node. One processor is the master processor and the rest are slave processors.
- **twitter_two_node_four_cores.slurm** – All processors can exist across the two allocated nodes.
- **languageConfig.json** – Language configure file used to map languages of the tweets.
- **bigTwitter.json** - Provided by University. Contains a collection of tweets that require processing.

2.3 Steps to invoke

The following is the step to invoke our code in SPARTAN:

- For one node one core -

  ```
  bash$ sbatch twitter_one_node_one_core.slurm
  ```

- For one node eight cores -

bash$ sbatch twitter_one_eight_cores.slurm

- For two nodes eight cores (four cores per node) -
  bash$ sbatch twitter_two_four_cores.slurm

2.4 Approach

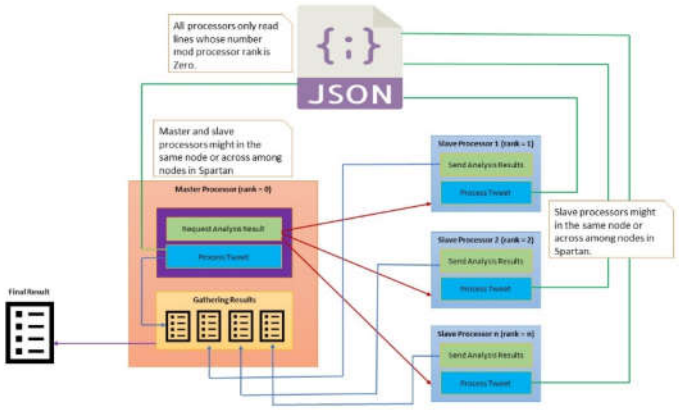

Figure 1. Approach Model

Our application is divided into three parts: 1) processing of twitter data by master processor 2) processing of twitter data by slave processors, and 3) sorting of hashtags and language information. The reason to divide our application into parts is because we are using Point-to-Point[1] communication. Our application contains communicators. Each communicator is an object describing a group of processors. A processor is any object that handles a process. In Point-to-Point communication, two processors in the same communicator communicate with each other. Along with Point-to-Point communication we have also used master-slave model. Each processor is assigned a rank. Processor with **rank = 0** is assigned as the master processor. All the other processors are considered as slave processors. Communication only takes place between master processors and slave processors and therefore no communication takes place among slave processors. All these processors can exist either in the same node or other nodes across SPARTAN. Slurm scripts help us to allocate nodes, processors and other resources for our application. In the background Slurm workflow manager will group all allocated processors to a communicator and these processors can be accessed through **MPI.COMM_WORLD**.

2.4.1 Master Processor

For the processor with **rank = 0**, **process_twitter_data(rank, data_path, processor_size)** is called. This allows the twitter data to be processed by the master processor and all the slave processors assigned by Slurm[2] workflow manager. If there are multiple processors, then **marshall_tweets(com)** function is invoked to request all the processed hashtags (**hashtag_counter**) and language information (**lang_counter**) from the slave processors. Here the master processor sends request to all the slave processors to get analysed data. If data has been received, the master processor sends an exit signal to the slave processors to stop.

2.4.2 Slave Processor

Similarly, for the processors with **rank > 0**, **process_twitter_data(rank, data_path, processor_size)** is called as well. This allows all the twitter data to be processed by all slave processors. Since there exists only

one master processor, all the rest are slave processors. Hence all the processed **hastag_counter** and **lang_counter** are sent to the master processor using **comm.send(analyzed_counters, dest = MASTER_RANK, tag = MASTER_RANK)**.

2.4.3 Sorting

Once the analysed data is collected by the master processor, they are sorted using the **most_common(TOP_MOST_COMMON)** function as the counters are dictionaries and can be sorted easily. The sorted information is then printed out.

3 Experiment

3.1 Overview

The experiment was run to understand how our application performs in SPARTAN and to understand the factors that affect the performance. Each slurm file was run once on SPARTAN High-Performance Computing (HPC) Facility. A total of 3 different files were run in total.

3.2 Measures

In our results, we have ignored the line of the Twitter JSON files ending with '**]}**'. As they affect the ability of our application to perform well. Regardless of the omission, our application runs without errors.

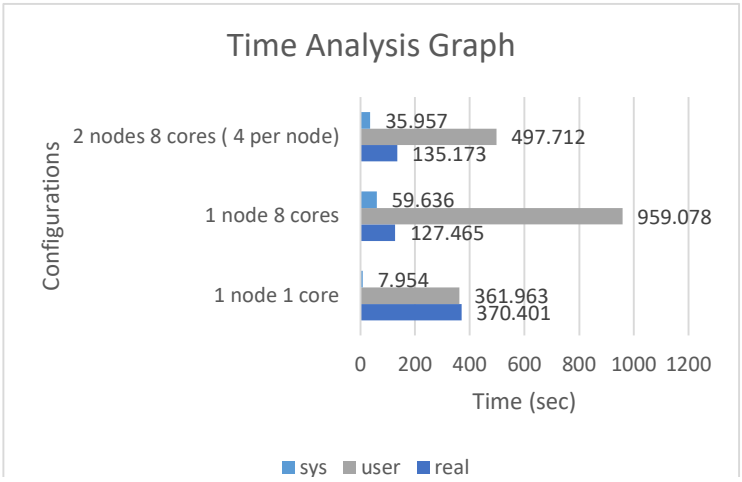

Figure 2. Graph of Configurations vs Time

3.3 Performance Analysis

Configuration	real (sec)	speedup	efficiency
1 node 8 cores	127.465	2.906	0.363
2 nodes 8 cores	135.173	2.740	0.343

Table 1. Values showing time, speedup and efficiency

We have calculated speedup and efficiency as described in [3]. In Figure 2 and Table 1, all values have been converted to seconds. For speedup and efficiency calculations we used real-time (total time of the entire execution) of our different configurations. Our efficiency dropped when the application was run in 1 node 8

core configuration. A further slight decrease in efficiency was observed for 2 nodes 8 core configuration even though the number of cores is the same for both the parallel modes. Therefore, while our MPI application runs well for a 1 node 1 core application, it does not show performance advantage for parallel configurations. Moreover, running application with additional nodes in parallel configurations would further reduce the performance according to our statistics as this is most probably due to the inter-processor communication between master and slave nodes where serialisation and deserialization takes place.

The other observation made was that the real-time is comparatively lower for parallel applications. For a single node, eight cores and two nodes, eight cores configurations the user time and sys time total gives a larger value than the real-time. This is because the total time collected here is across all the nodes.

On the other hand, parallel configurations utilise less time to process data compared to single process configuration. Moreover, the addition of another node reduces the user and sys time by around close to 50%. The extra real-time taken for 2 nodes 8 cores application when compared with 1 node 8 cores application is due to the fact that we cannot parallelise for language configuration loading, waiting time to receive analysed results from slave processors and the gathering time for all results in master processor.

4 Results

Below are the top 10 most common hashtags and most common languages obtained from our application. We evaluated that **#auspol** is the most popular hashtag among the data with a total number of **19,878**. Also, the most popular language is **English (en)** with a total number of **3,107,116**.

```
Top 10 most commonly used hashtags:
1. #auspol, 19,878
2. #coronavirus, 10,110
3. #มาฟ่องเพื่ออะไร, 7,531
4. #firefightaustralia, 6,812
5. #oldme, 6,418
6. #sydney, 6,196
7. #scottyfrommarketing, 5,185
8. #grammys, 5,085
9. #assange, 4,689
10. #sportsrorts, 4,516
```

```
Top 10 most commonly used languages:
1. English (en), 3,107,116
2. Undefined (und), 252,117
3. Thai (th), 134,571
4. Portuguese (pt), 125,858
5. Spanish (es), 74,028
6. Japanese (ja), 49,929
7. Tagalog (tl), 44,560
8. Indonesian (in), 42,296
9. French (fr), 38,098
10. Arabic (ar), 24,501
```

Figure 3. Top 10 most commonly used hashtags and languages

5 Reference

[1]. Introduction to MPI: https://www.codingame.com/playgrounds/349/introduction-to-mpi/mpi_comm_world-size-and-ranks

[2]. Running a Job on HPC using Slurm: https://hpcc.usc.edu/support/documentation/slurm/

[3]. Amdahl's Law: https://www.cise.ufl.edu/~mssz/CompOrg/CDA3101-S16-AmdahlsLaw-TEXTSUMMARY.pdf