THE UNIVERSITY OF MELBOURNE

COMP90024 Cluster and Cloud Computing Assignment 1 - Project Report

Matthias Bachfischer - Student ID 1133751

April 11, 2020

1 Introduction

This document serves as a project report for assignment 1 of the COMP90024 Cluster and Cloud Computing course at the University of Melbourne. It describes the general system architecture that was implemented, instructions and commands required for invoking the scripts on the Spartan High Performance Computing (HPC) cluster as well as steps that were taken to parallelize the processing and leverage the available resources in an efficient way.

1.1 Dataset

The task for this assignment was to process the *bigTwitter.json* dataset which consists of multilingual Microposts that were extracted from the Twitter social networking platform[1]. The dataset has a total size of 20.7 Gigabyte (GB) and is structured in the JavaScript Object Notation (JSON) document format.

1.2 Instructions

To run the software on Spartan, copy or clone the contents of the folder containing the source code into your home directory. Change into the `slurm` directory and examine its contents. As shown in Table 1.2, this directory consists of three files that allow the user to trigger the execution of the program in three different configurations.
In order to start processing the dataset, run the command `sbatch scriptname` and replace `scriptname` with the filename of the script that you want to run (i.e. for 1 node - 1 core configuration, run `sbatch tweetanalyzer_1node_1core.slurm`). Executing this command will submit a new job to Slurm queue on Spartan HPC cluster and schedule it for execution.

[1]Twitter social networking platform https://twitter.com

Resource configuration	**SLURM script**	**Execution time**
1 node and 1 core	tweetanalyzer_1node_1core.slurm	447.262 seconds
1 node and 8 cores	tweetanalyzer_1node_8cores.slurm	59.062 seconds
2 nodes and 8 cores	tweetanalyzer_2nodes_8cores.slurm	65.907 seconds

Table 1: Resource configuration, corresponding SLURM script and execution time

2 System description

The software that was developed for this assignment is written in the Python programming language and makes use of the MPI for Python programming library (Dalcin et al., 2011) to ensure parallel execution of computing steps in a multi-core / multi-node computing scenario. The software was specifically designed to run on the Spartan HPC system operated by the University of Melbourne (Lafayette et al., 2016),

2.1 General processing steps

As mentioned in section 2, the software that was implemented for this assignment relies on the Message Passing Interface (MPI) specification - in particular, it uses Scatter and Gather functions to scatter instructions from the master process to the worker processes and gather the results from the worker proceses in the master process (*MPI: A Message-Passing Interface Standard Version 2.2*, 2009). The basic algorithm that was implemented to ensure parallel execution across multiple CPU cores is as follows:

1. Create *n* worker processes where #*n* denotes number of available cores
2. Split the dataset into *n* equally sized chunks using method *chunkify*
3. Distribute parameters *chunkStart, chunkSize* to worker processes via *Scatter* function, where *chunkStart, chunkSize* denotes the offset and size of the chunk in bytes (starting from the beginning of the file)
4. Split chunks into equally sized batches of size 1024 bytes using method *batchify*
5. Processes individual batches line by line as described below
6. Collect counter results from worker processes in master process via *Gather* function
7. Aggregate individual counter values and print final results

2.2 Task 1: Counting of tweet hashtags

In task 1, the objective was to count the occurrences of the top-10 most common hashtags within the dataset. Conveniently, the hashtags for the individual tweets have already been extracted and are present in the JSON field *tweet['doc']['entities']['hashtags']* of each tweet. A hashtag string can match if it has upper/lower case exact substrings and needs follow the Twitter rules for hashtags[2] (i.e. no spaces and no punctuation are allowed in a hashtag).

[2]How to use Twitter hashtags https://help.twitter.com/en/using-twitter/how-to-use-hashtags

In accordance with the given rules for hashtag format, the tweet hashtags were processed according to the following procedure:

1. Convert hashtags to lower-case strings
2. Split the individual strings if punctuation marks or spaces (except underscore _) occur
3. Use the first part of string (if a punctuation mark is present)

An example for matching hashtag strings is given in Example 2.2.1:

Example 2.2.1:
Original: #covid19, #Covid19, #COVID19, #covid19.2020
Processed: #covid19

During the processing step (i.e. call to `process_tweet()`), a list of hashtags for a particular tweet is extracted and a counter variable `extracted_hashtags_counter` keeps track of the number of occurrences of a particular hashtag in the dataset.

2.3 Task 2: Counting of tweet language codes

Objective of task 2 was to count the top-10 most common languages of the tweets that are present within the dataset. The language attribute of a particular tweet can be extracted from the JSON field *tweet['doc']['metadata']['iso_language_code']* of each tweet. The language attribute is stored as a two-letter code as specified in the ISO 639-1 specification (*ISO 639-1:2002, Codes for the representation of names of languages*, 2002) and needs to be mapped to the corresponding ISO language name.
In order to achieve this, the list of languages along with the language code supported by Twitter was obtained by using the Twitter Application Programming Interface (API) endpoint *GET help/languages*[3]. The result of this API call is stored in the file `language_codes.json`. Since setting up API communication with Twitter has not been within the explicit scope of this assignment, the details of how the file `language_codes.json` was obtained is left out of this project report.
Similar to the procedure described in section 2.2, the language code of a particular tweet is extracted during the call to `process_tweet()` and a counter variable `extracted_language_-counter` keeps track of the number of occurrences of a particular language code in the dataset. When printing the final result output, the file `language_codes.json` is then used to map the two-letter language code to its corresponding language name (e.g. language code ”en" corresponds to "English").

3 Performance results

The system architecture presented in this report was tested on the dataset described in section 1.1. The dataset was processed using three different computing configurations: 1 node - 1 core, 1 node - 8 cores and 2 nodes - eight cores (i.e. 4 cores per node).

[3]Get Twitter supported languages https://developer.twitter.com/en/docs/developer-utilities/supported-languages/api-reference/get-help-languages

For each process, the execution time from when the process was started until when the process was finished was measured. To measure the total execution time, the execution time of the master process identified with rank 0 was used.

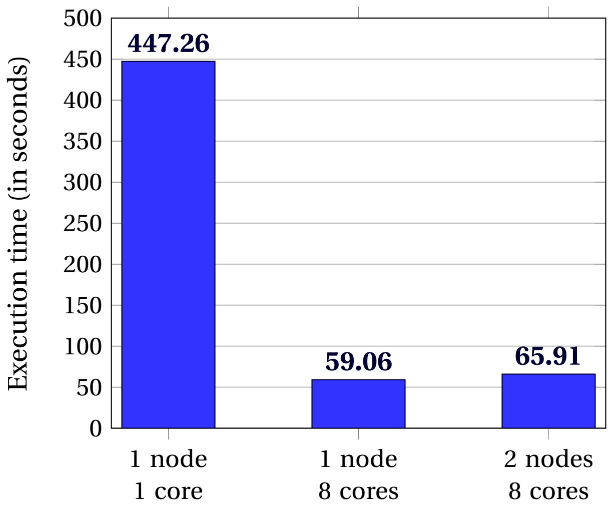

Figure 1: Performance benchmark

As shown in figure 3, the configuration running on 1 node - 8 cores achieves best performance and is able to finish the processing of the dataset in 59.06 seconds. The configuration running on 2 nodes - 8 cores requires approximately 5 seconds more - this is probably caused by the communication overhead involved in scattering instructions to a different node via the network and subsequently gathering the results from this node. The 1 node - 1 core configuration performs worst and takes approximately 447 seconds to complete. The results of this benchmark are not surprising, as they show the great benefit of parallelization when it comes to processing in a big data context.

4 Conclusion

The objective of this assignment was to efficiently process a large dataset using the HPC facilities provided by the University of Melbourne. As part of this, a program was developed that efficiently processes the dataset in small batches in parallel. It has been shown that the approach that was taken to parallelize the execution enabled the processing of a 20 GB Twitter dataset in less than one minute in the optimal configuration (1 node - 8 cores scenario).
Moreover, the implementation used for this assignment does not rely on any external libraries (expect MPI for Python) and can therefore be applied in a multitude of environments, thus making it highly reusable when dealing with large datasets in HPC environments.

References

Dalcin, L. D., Paz, R. R., Kler, P. A., & Cosimo, A. (2011). Parallel distributed computing using python. *Advances in Water Resources*, *34*(9), 1124–1139.

Iso 639-1:2002, codes for the representation of names of languages (Report). (2002). International Organization for Standardization. Geneva, CH.

Lafayette, L., Sauter, G., Vu, L., & Meade, B. (2016). Spartan performance and flexibility: An hpc-cloud chimera, In *Openstack summit*, Barcelona.

Mpi: A message-passing interface standard version 2.2 (Report). (2009). Message Passing Interface Forum.