Mini Project 2 – Twitter Mining & MapReduce

Overview

This assignment is worth 30% of your overall coursework mark and is due Friday November 30th 2018 at 21:00 as shown on MMS. Normal lateness penalties apply

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Setting Up

As in many previous practicals, we are using the automated checker `stacscheck`. For this practical, your program must have a `main` method in a class called `ProjectMain` which is in a file `ProjectMain.java` that is located inside your `src` directory in your `MiniProject2` directory.

Background

Twitter is a micro-blogging service that is fairly popular and on a typical day carries some 500 million messages

- https://blog.twitter.com/2013/new-tweets-per-second-record-and-how
- http://www.internetlivestats.com/twitter-statistics/

In this assignment you will write a program to produce statistics on Twitter data taken from a small subset of a 1% sample of Tweets made in January 2018 and conduct performance experiments with your program.

Description

The task is split into two main parts. In part 1, you will develop a program that extracts information from the Twitter data. In part 2, you will run some experiments with your programs and measure performance.

Part 1

In this part, the objective is to build a program that uses Apache Hadoop (http://hadoop.apache.org) to implement a Map-Reduce to extract some possibly interesting information from a small subset of Tweets made in January 2018 in JSON format (by counting the occurrence of hashtags in the Tweets it is given). When developing your application, you should make use of some small test data at

https://studres.cs.st-andrews.ac.uk/CS2101/Practicals/MP2/data

Each file contains a sequence of JSON objects corresponding to Tweets. You should examine the format used to represent Tweets as described at https://dev.twitter.com/overview/api/tweets. You should study examples of using Hadoop to count the occurrence of words as shown in lectures and at

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_MapReduceExample/

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_MapReduce_MultipleRunningMappers_Example/

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_MapReduce_MultiThreaded_Example/

You will have to write code (probably in your *Mapper* class) to interpret the Twitter data and find the hashtags. You should study examples of parsing JSON data as shown in

https://studres.cs.st-andrews.ac.uk/CS2101/Examples/CS2101_JsonReaderExample/

Your program should process Twitter data from an *input path* (to a whole directory or file) specified as the first command-line argument (`args[0]`) and write a file to the *output directory* specified as the second command-line argument (`args[1]`). See examples of compiling and running your program shown below. The output produced by your Hadoop program should be a count of all hashtags used in the input, ordered by hashtag.

Once you have finished developing your program and are satisfied that it works on the smaller data, you may wish to download some more data for your program to examine. For example, you might take 15 minutes, half an hour, or even an hour of data from some date and see whether the hashtags and counts shown by your program are those that you would expect given what was going on at that time. You are supplied with a fairly sizeable amount of compressed 2018 Twitter data which is available on Lab clients and server machines at `/cs/unique/twitter`. As such, you can access this the data in the specified directory when connected to `<username>.host.cs.st-andrews.ac.uk` (replacing `<username>` with your own username) via SSH or SFTP or when logged into one of the machines in the lab. The *Tweet* files for the whole Twitter sample on Student Resources are compressed using bzip2 (http://www.bzip.org/) and stored in a directory hierarchy according to the date and time they were composed, namely as:

year/month/day/hour/minute.json.bz2

Please note, each file (i.e. a minute of data) may take up roughly 15MB when it is uncompressed. A day can take up roughly 20GB, so please don't use that much. only copy some minutes of data to your local system and decompress the file(s) (using e.g. `bunzip2 *.bz2` in a terminal window) onto your local file system. If you want to experiment with a larger amount of data (i.e. an hour or so of data), please make sure you don't store them in your home space which is held on a networked storage server. Storing the files on your home server will not only exhaust your disk quota very quickly but also severely slow down access to the files, stress our networks, and will almost certainly invalidate any results from the experiments mentioned in **part 2** below. If you are working on the Lab Clients you should copy files to `/cs/scratch/<your-username>/` (where you should replace `<your-username>` with your own username).

Compiling and Running your program

In order to be compiled by the stacscheck auto checker, it must be possible to compile your program from your `MiniProject2` assignment directory using the command below

```
mkdir -p bin
javac -cp "lib/*" src/*.java -d bin
```

Below are some examples showing how your program should be run and the expected output when no command-line arguments are supplied. The commands assume you are working on a Linux lab machine, that the current directory is the `MiniProject2` directory (which contains your `src` directory) and that you are running the Hadoop job locally (rather than on a cluster). Please also make sure that the output directory specified as command-line argument does NOT exists prior to running your program, the Hadoop job will terminate if the directory already exists.

```
java -cp "lib/*:bin" ProjectMain
Usage: java -cp "lib/*:bin" ProjectMain <input_path> <output_path>
```

```
   java -cp "lib/*:bin" ProjectMain /cs/studres/CS2101/Practicals/MP2/data/data-very-small/00.json output00
```

The first invocation above does not pass any command-line arguments to the program and results in the required usage message being displayed. The second invocation does pass the expected command-line arguments to the program and as a result would produce some Hadoop debug output to the terminal and more importantly, an output file `output00/part-r-00000` containing the count for 3 hashtags found in the file as shown below.

```
   AllTheStars          1
   Brexit               1
   FindYourRoute        1
   MRpoints             1
   MembersGetIt         1
   RewardsPoints        1
```

Running your program using

```
   java -cp "lib/*:bin" ProjectMain /cs/studres/CS2101/Practicals/MP2/data/data-very-small/01.json output01
```

should produce the output file `output01/part-r-00000` containing the count for 6 hashtags found in the file as shown below.

```
   FindYourRoute        2
   MRpoints             2
   MembersGetIt         2
   Philtippett          1
   RewardsPoints        2
   StarWars             1
   StarshipTrooper      1
```

Similarly, running your program using

```
   java -cp "lib/*:bin" ProjectMain /cs/studres/CS2101/Practicals/MP2/data/data-very-small output
```

should produce the output file `output/part-r-00000` containing the count for all hashtags found in all the files in the `data-very-small` directory (and all of its sub-directories) as shown below.

```
   AllTheStars          1
   Brexit               1
   FindYourRoute        3
   MRpoints             3
   MembersGetIt         3
   Philtippett          1
   RewardsPoints        3
   StarWars             1
   StarshipTrooper      1
```

Hadoop creates the `part-r-00000` output files above and separates hashtags and counts using a TAB. Try running your program on the files in the `data-very-small` directory and try to debug errors before moving on. There are two other data directories at https://studres.cs.st-andrews.ac.uk/CS2101/Practicals/MP2/data which contain 1 minute and 10 minutes of data respectively taken from January 7th 2018 at 16:00. These are also used by the auto checker to test your program.

You should use Hadoop Map-Reduce wherever applicable to compute the above, over conventional methods. You can reuse any code from examples or previous practicals, so long as you clearly identify it.

Running the Automated Checker

As usual, please use the automated checker to help test your attempt. Please make sure you have a `MiniProject2` directory containing

- your `src` directory containing all your source code including a `ProjectMain` class containing your `main` method in `ProjectMain.java`
- a `lib` directory with all the Hadoop jars as shown in the example code on studres

If your `MiniProject2` directory does not contain these the auto checker will fail.

The checker is invoked from the command line on the Linux Lab Machines in your `MiniProject2` folder:

```
stacscheck /cs/studres/CS2101/Practicals/MP2/Tests
```

The tests use the Unix `diff` command to print a message describing how a line (or set of lines) in your output should be changed to match the required output. If some of the tests are failing, examine the `diff` output and try to identify why your program is not compiling, running, or producing the output as expected. By examining the test files at

https://studres.cs.st-andrews.ac.uk/CS2101/Practicals/MP2/Tests/basic

you can also see which data files are used in the tests (by looking in the `test.sh` file for a particular test). The data files are all in the data directory at

https://studres.cs.st-andrews.ac.uk/CS2101/Practicals/MP2/data

You can also see the expected output by examining the `expected-output.txt` file for a particular test and then compare it yourself with the `part-r-00000` file produced by your Hadoop program when you run it on the same input as used in the test. If nothing is working and you don't know why, please don't suffer in silence, ask one of the demonstrators in the lab.

Part 2

In this part, the aim is to conduct a series of experiments with your program(s) in which you investigate the impact on performance (i.e. execution time) of:

- varying the number of concurrently running map tasks on your system
- varying the number of threads used for individual map tasks

These aspects can be explored by choosing a number of appropriate values for the *LocalJobRunner* class and *job* prior to running the job (as shown in the *MapReduce_MultipleRunningMappers_Example* and *MapReduce_MultiThreaded_Example* on Student Resources.

You are advised to choose a suitable amount of data that will permit the jobs to run for long enough (a minute or two at least), otherwise you are likely to be measuring the time it takes to set up a Hadoop job rather than the time it takes to actually run the job once it is set up.

You may wish to run each configuration multiple times and establish an average so as to minimise noise in your results. You should try to automate the execution of runs where possible rather than merely running the programs manually.

You should produce plots of your results for inclusion and discussion in your report (these do not have to be produced by your program directly, you can plot your result data using MS Excel or some other application). You may discuss the results in relation to the number of cores on the machine you are using.

Basic Requirements

1. Your program should be able to use Hadoop to establish a frequency count of hashtags occurring in suitable subsets of the Twitter sample.
2. Experiment with and report on and discuss the effect of varying the number of concurrently executing map tasks and threads per map task on your system. Provide a means of automating the execution of your experiments.

Extensions

Please first build a basic working solution, fulfilling the Basic Requirements listed above, before trying anything more complex listed below. To gain a mark of 17 or above, you may consider completing one or more of the items below:

- Give the user an option to order the hashtags by occurrence frequency. Hint: You could use the 'Most Popular Words' approach as outlined in lecture slides. Use this to establish a top 40 list of most popular hashtags.
- Establish the most frequently re-tweeted Tweets
- Establish the users most frequently replied to or having their Tweets re-tweeted
- Limit results to Tweets that are in English

Report

You should write a report (with an *advisory* limit of 3000 words) with the information listed below.

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#word-count-penalties

In your report, where appropriate, try to concentrate on *why* you did something rather than merely explaining what you did:

- A short overview summarising the functionality of your application indicating the level of completeness with respect to the requirements listed above
- Design and implementation sections including a short discussion of any interesting features in your implementation such as the addition of any features beyond the original specification
- An accurate summary stating which files or code fragments you have written and any code you have modified from that which was given out in class or which you have sourced from elsewhere
- A short summary of the tests you conducted with your system
- A presentation and discussion of the Tweet statistics you obtained, a short extract from your full Tweet count or preferably your most frequently occurring tweets (top 40 list) if you did this part, specifying precisely the data files you used (e.g. from day *X* hour *I* up to day *Y* hour *J*).
- A discussion of the performance experiments you conducted and the results, supported by suitable graphs and explanations.

Submission

Please submit a single `.zip` archive containing all your code and your report as `.pdf` to the CS2101 "Mini-Project 2" (MP2) slot on MMS by the specified deadline. Please **don't** include any of the Twitter data in your submission. For extension work, please do include instructions for running your application(s) with relevant parameters. Remember that your submission MUST compile and execute on the lab machines, but should ideally work on any machine with Java 1.8.

Marking

The submission will be marked on the University's common reporting scale according to the mark descriptors in the Student Handbook at:

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

The following aspects will also be considered:

- Quality of the report
- Quality of the design and implementation
- Completeness of the implementation and experimentation with respect to the requirements
- Quality and clarity in experimental design and presentation and evaluation of the results
- Extension work

I would remind you to ensure you are following the relevant guidelines on good academic practice as outlined at https://www.st-andrews.ac.uk/students/rules/academicpractice/