Running Map-Reduce in Hadoop on Yelp data

Eleni Ilkou, Jan Swimberghe

Advisors: Nikolaos Deligiannis,
Tien Do, Steven Vanden Broucke

Distributed Computed and Storage Architecture Project

January 2019

1 Introduction

This report investigates the deployment of big data analysis by running map-reduce in Hadoop on Yelp[1] data. Yelp is a local search service powered by crowd-sourced review forum about local businesses. On our assignment we have to analyse data gathered from Yelp about businesses and reviews using Hadoop cluster within Cloudera. The requested tasks are solved by the means of map-reduce, which was introduced in the lectures, and the usage of two JSON files on reviews and businesses. This report consists of a step by step installation guide, the tasks' solutions approaches, the results and the conclusion part.

2 Installation

This section presents a step by step guide for all the installations needed. This is the first part for our project where we needed to get introduced with new technologies. For the purposes of this project, we got introduced to technologies such as the Cloudera cloud, the Hadoop framework and Virtual Box.

2.1 VirtualBox

We need to choose a hyper-visor application to install the Cloudera. For this project we were recommenced to use the Oracle VM VirtualBox as our virtual machine manager. Oracle virtual machine VirtualBox is a free, open source, general-purpose full virtualizer for x86 hardware, targeted at server, desktop and embedded use.

Figure 1: VirtualBox website

To install the VirtualBox:

1. Download the VirtualBox[2]

[1] More information about Yelp can be found at yelp.com
[2] You can download VirtualBox from https://www.virtualbox.org/wiki/Download_Old_Builds_5_2

2. Install the VirtualBox on your desktop

2.2 Cloudera

Cloudera QuickStart VM [3] (single-node cluster) make it easy to quickly get hands-on with CDH (Cloudera's Distribution including Apache Hadoop) for testing, demo, and self-learning purposes, and include Cloudera Manager for managing our cluster.

Once we have the VirtualBox installed and working on our desktop, we can get the Cloudera version for our platform. To get a Cloudera QuickStart VM we needed to:

1. register on their website,
2. download the version for the VirtualBox platform

Now we are ready to open our virtual machine and continue the rest of the installations from the command prompt.

2.3 Python

For this project we use Python 3. It was recommended to use Python 2.7, however after a little research we found out that MRJob and all components are compatible with the latest Python version. Hence for our assignments we used Python 3.6.4

Before we get python, we need to install the SQL lite and development tools:

1. `sudo yum install sqlite-devel`
2. `sudo yum groupinstall -y "Development Tools"`

The steps to install Python and the package manager:

1. `wget https://www.python.org/ftp/python/3.6.4/Python-3.6.4.tar.xz`
2. `tar -xJf Python-3.6.4.tar.xz`
3. `cd Python-3.6.4`
4. `./configure`
5. `make`
6. `sudo make install`

[3] https://www.cloudera.com/downloads/quickstart_vms/5-13.html

From now on, we can install the rest requirements in fewer steps, with the pip3 install command.

2.4 Tokenization

For the task 3 of our project we use the built-in tokenizers from nltk. Tokenization is used to split a raw text into useful meaningful components. The nltk is a tokenization toolkit, that includes different libraries for text processing that simplify the process of workinf with natural language data. To install nltk: **pip3 install -U nltk**

Nltk is now on our desktop, however to enable its capabilities our next step is to download a particular data set in which it will do all the processing. For our purposes we used the popular model.

```
[cloudera@quickstart Python-3.6.4]$ python3
Python 3.6.4 (default, Jan  6 2019, 09:33:14)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-23)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.download('popular')
```

Figure 2: Install nltk

2.5 MRJob

MRJob is the easiest route to writing Python programs that run on Hadoop. It allows to run jobs in the cloud and on a pc. Furthermore, it is a way of interfacing with Hadoop MapReduce in Python. It has built-in support for many options of running Hadoop jobs.

To install MRJob:

```
sudo pip3 install mrjob
```

Figure 3: Install MRJob

To check if MRJob works we can run our script locally by using the command **python3 python_file_name.py json_file_name.json**

2.6 Cloudera Express

Once our VM is running, we start the Cloudera express by the following command.

1. To start cloudera express:

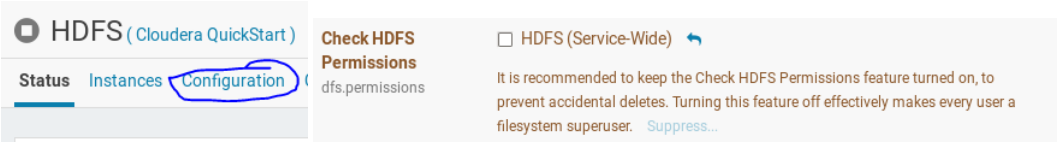
```
[cloudera@quickstart ~]$ sudo /home/cloudera/cloudera-manager --force --express
```

2. To login to Cloudera we use as username and password the word 'cloudera'

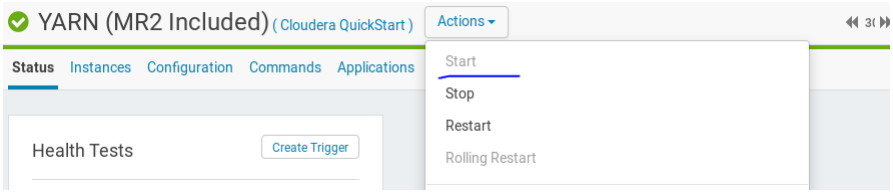
```
http://quickstart.cloudera:7180

Username: cloudera
Password: cloudera
```

3. To check the HDFS permissions



4. To start YARN



After installing everything we are ready to go to the last step.

2.7 Hadoop

The Apache Hadoop[4] software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

To start with Hadoop we need to upload the JSON files.

```
hadoop fs -put reviews.json  /user/cloudera
hadoop fs -put businesses.json  /user/cloudera
```

Figure 4: Hadoop

[4]https://hadoop.apache.org/

3 Computation Tasks

Once we have all the requirements installed we are ready to proceed to the computational part. In this section, we are going to compute four different tasks on data set from Yelp. The data set is consisted of two JSON files, one one reviews and one on businesses, on which we will process in parallel using MapReduce. Each task consists of a python file, which includes the MRJob steps and the MapReduce, which runs in combination with the right JSON file on Hadoop.

To run each of them on Hadoop we use the command

python3 python_file_name.py –python-bin -Python/python_version -r hadoop hdfs:// json_file_name.json

where:

- python3, specifies that we run the Python 3 version
- python_file_name.py, the python script which will run
- –python-bin -Python/python_version, the position where the Python version is placed
- -r hadoop, defines that it does not run locally
- hdfs:// json_file_name.json, the directory of the JSON file on which the python script will run

3.1 Task 1

On the first task we need to compute the top 10 user ids by the amount of reviews given on the reviews JSON file.

Our solution consists of:

1. a mapper, that reads each line in the reviews.json and yields a key pair with the user_id and 1
2. a combiner, that counts the times by calculating the sum of each user_id being present in the review.json
3. the first reducer, that yields all the data to the final step
4. the last reducer, which identifies the top 10.

The results we obtained with the top 10 user IDs are the following:

```
198     "U3Su5FLssAUKNYonpzIgRw"
197     "vLixm38Zd-xgAvN1LQJfKQ"
197     "9BuTUHt6Zlm-UCjaGD6aJQ"
176     "eeBt7Uo5FOXwvaLIeW3fGA"
163     "jEBySjq6tgL-_R3P7LHq0w"
162     "miYadhpWV81SXUEae3LX_A"
155     "460PoNGmuFFKF4cJ2XQj9A"
137     "xovKY9oB8s-NjKz_I1RpHQ"
134     "qt1b6zXExL-uoJGRRouQYw"
131     "yoT5mQ0EeB40AAMi4ZIyTw"
```

Figure 5: Task 1 results

3.2 Task 2

For Task 2, we need to count the number of businesses that have a television, and offer a happy hour. We use the properties HasTV and HappyHour from the businesses.json file.

Our solution consists of a mapper and a reducer. We mapped the 3 different cases we could obtain from our data set, the has both television and happy hour, does not have both, or it is unknown. We specifically added the last category, because we observe the possibility of a business that may have both attributes in reality, however they might not be yet identified on the data set. This could probably assist on a better understanding of big data analysis.

The results we obtained with the number of businesses that have a television and offer a happy hour are the following:

```
"HasTVHappy"    5861
"Not"   182732
"Unknown"       25786
```

Figure 6: Task 2 results

3.3 Task 3

Task 3 was about to produce a list of the top 25 most used words from all review texts. An essential part of text analysis is to have clean text. Tokenization is used to split a raw text into useful meaningful components. You can use built-in tokenizers from nltk, namely, the TweetTokenizer. We ran this task twice, once without the tokenizer and one with it to detect the differences that this preprocess tool will make in the mapper.

Also our solution consists of a combiner and two reducers. The second one is used to identify the 25 with the biggest counts.

The results we obtained are the following:

793751	"i"	145453	"food"
258793	"the"	131956	"place"
145453	"food"	129239	"good"
131956	"place"	128414	"great"
129239	"good"	112029	"service"
128414	"great"	98498	"time"
112029	"service"	90486	"like"
98498	"time"	90279	"would"
94541	"we"	88158	"one"
90486	"like"	88135	"get"
90279	"would"	84635	"back"
88158	"one"	75886	"go"
88135	"get"	64778	"really"
84635	"back"	60783	"us"
83691	"..."	56199	"also"
75886	"go"	54991	"even"
67845	"they"	52221	"best"
64778	"really"	49677	"got"
60783	"us"	48928	"staff"
56350	"this"	48069	"nice"
56199	"also"	46239	"never"
54991	"even"	45535	"well"
54650	"it"	45152	"always"
52221	"best"	44839	"went"
51805	"my"	43755	"i've"

Figure 7: Task 3, on the left pre-processed results, on the final right results

3.4 Task 4

For task 4 we needed to compute the top 10 business IDs that have a minimum of 5 review together with their weighted scores, that we calculate based on a the given formula below. Rather than using a standard average we will utilise the useful vote count to create a weighted average. Take note that reviews without a useful vote should not be discarded.

The formula for the weighted score is calculated as follows: for an individual business, where R is the set of reviews for that business, s_i is the score (stars) given for a review and u_i is the number of useful votes for a specific review, we can formulate the weighted score w as:

$$u = \sum_{i=1}^{|R|} (u_i + 1) \tag{1}$$

$$w = \sum_{i=1}^{|R|} \frac{s_i * (u_i + 1)}{v} \tag{2}$$

Our solution consists of:

1. a mapper, that calculates for each review the (u_i +1) from (1) and the s_i *(u_i +1) from (2) and yields a key pair with the business_id and a tupple of the previous calculations and the 1

2. the first reducer, that for each business_id it calculates the (1), and the upper part of the fraction in the (2), and counts the sum of each business_id being present in the review.json

3. the second reducer, that yields the data which have minimum 5 reviews to the final step

4. the last reducer, which identifies the top 25.

By changing the number of top business we want to detect we can see that more than 1000 businesses in our data set manage to get perfect review (5 star review). Therefore, the results we get are only a small part of the group set of excellent businesses.

```
5.0     "zyPGYeXF4XKCqNN1pjFWhg"
5.0     "zvxaG-BIuxo5Hm6UO2yG8Q"
5.0     "zsq3pzBUTf8QZAChHN2urQ"
5.0     "zsNEni7cr4lRnjd4AcO4kA"
5.0     "zniY0Pd9BTUwoFuejSMPBg"
5.0     "zjBJPx2Hm-cx1xfeIjcKHQ"
5.0     "zhj1rDmdmliip6PMYudQug"
5.0     "zfwuWqm8X7IZXieYwiy1fw"
5.0     "ze0vTDa1oFIH6oSRWgxM_A"
5.0     "zdFqaUk76GAWsDZgPSWG-A"
```

Figure 8: Task 4 results final

4 Conclusion and Team contribution

This project taught us a lot about distributed computing the use of map-reduce, the Cloudera server and the Hadoop software. The knowledge gained will be the basis on which we will develop our future skills on big data analysis.

When it comes to the contribution part, we worked well as a team. Jan took responsibility to install all the requirements on his desktop and finished the first task by his own, and did all the debugging. We both worked on the last three tasks, and finally Eleni made the report and commenting on the code.