Cloud Computing Capstone

A Real-World Challenge Project

Bogdan Gersak - 6 February 2016

http://www.htguk.com/benefits-of-cloud-computing/

Introduction

The goal of the Capstone Project is to provide an opportunity to synthesise the knowledge and skills we have learned from Cloud Computing Specialisation and apply them to solve real world cloud computing challenges. We will work on a transportation dataset in CSV format from the US Bureau of Transportation Statistics (BTS) that is hosted as an Amazon EBS volume snapshot answering a common set of questions using Hadoop.

Video presentation and source code are available on Github: https://github.com/gersakbogdan/coursera/tree/master/Cloud%20Computing%20Capstone/Group%201

System Setup

Our working environment consist into a Hadoop Cluster with 1 name node (master) and 3 data nodes (Figure 1). The full installation process is not trivial but there are a lot of resource about how to do it. The one I followed is this: http://insightdataengineering.com/blog/hadoopdevops/

Name	Instance ID	Instance Type	Availability Zone	Instance State
master	i-7b5704c8	c3.xlarge	us-east-1a	running
slave3	i-7c5704cf	c3.xlarge	us-east-1a	running
slave2	i-7d5704ce	c3.xlarge	us-east-1a	running
slave1	i-7e5704cd	c3.xlarge	us-east-1a	running

Figure 1

For storing our results beside HDFS we choose to use Amazon DynamoDB.
Golang was also used for filtering CSV data.
No other software was installed, all map reduce jobs are written in Java.

Data Ingestion

Because the attached EBS data volume which was available for us contains more data than we need, after mounting and attaching the volume we choose to use only the 'aviation' directory and more specific the airline on time data and only the required fields: FlightDate, UniqueCarrier, FlightNum, Origin, Dest, DepTime, DepDelayMins, ArrTime, ArrDelaysMins.

Cleaning and filtering fields was made with a simple Golang script (cancelled flights were also filtered out) and the import into HDFS with a Hadoop MapReduce job. For saving resources the data saved into hdfs is compressed with bzip2.

Import data bash script:

```
#!/usr/bin/env bash

# key offset, value is a zip file
read offset dirname

# Un-zip files
target=`basename $dirname`

# Move files to slave
mkdir -p $dirname
scp -r master:$dirname `dirname $dirname`

echo "reporter:status:Un-zipping $dirname" >&2
itr=0
# Concat into one file
for filePath in $dirname/*.zip ; do
    filename=`basename $filePath`
    itr=$((itr + 1))

    unzip -p $filePath *.csv | \
    /home/ubuntu/work/bin/gocsv | \
    bzip2 > "$target$itr.bz2"

    echo "reporter:status:Processed $filePath" >&2
done

# Put bzipped version into HDFS
echo "reporter:status:Bzipping $target and putting in HDFS" >&2
cat $target*.bz2 | $HADOOP_HOME/bin/hadoop fs -put -f - data/$target.bz2
```

Hadoop MapReduce job:

```
#!/usr/bin/env bash

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-D mapred.reduce.tasks=0 \
-D mapred.map.tasks.speculative.execution=false \
-D mapred.task.timeout=12000000 \
-input directories.txt \
-inputformat org.apache.hadoop.mapred.lib.NLineInputFormat \
-output output \
-mapper /home/ubuntu/capstone/import/import.sh \
-file /home/ubuntu/capstone/import/import.sh
```

The full source code which includes Golang script for filtering fields can be found here: https://github.com/gersakbogdan/coursera/tree/master/Cloud%20Computing%20Capstone

Questions & Results

Group 1

1. Rank the top 10 most popular airports by numbers of flights to/from the airport.
2. Rank the top 10 airlines by on-time arrival performance.

This group for questions are following the same pattern with the WordCount problem. The important optimisation here is the use of a combiner class for reducing network traffic and also to fast up the calculation.
Source code: https://github.com/gersakbogdan/coursera/tree/master/Cloud%20Computing%20Capstone/Group%201

Group 2

1. For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X.
2. For each airport X, rank the top-10 airports in decreasing order of on-time departure performance from X.
3. For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X.

This group of questions are more complex and this is the place were you can start to ask if a straight MapReduce solution with Java is the best solution or not. When the questions starts to be more complicated you probably need to split them and create a chain of map reduce jobs to find the answer. In our case we first need to find the on-time departure/arrival performance to be able to apply the next filter, top 10 carriers, airports etc.

The solution we saw over the course was to keep the Top 10 results in memory (at map and reduce steps) then combine and extract the Final Top 10 at reduce step. This is a good algorithm which solves the Top N questions but in our case I decided to apply a different algorithm called Secondary Sort. With Secondary Sort algorithm is no need to sort the values in memory because it uses the shuffle and sort technique of the MapReduce Framework. I preferred this solution because we do not depend on memory.

For this group of questions one of the requirements was to save the data into Cassandra/DynamoDB database. I did this by creating a custom Hadoop Output Format extending the OutputFormat<K, V> class and in this way the results are saved directly into DynamoDB. To enable this we need to set the output format class:

job.setOutputFormatClass(DynamoDBOutputFormat.class);

Conclusion: It was an interesting learning process about Hadoop it self and Java but I think Hive, Pig etc. are a better fit for this type of questions.

More about algorithm: https://www.safaribooksonline.com/library/view/data-algorithms/9781491906170/ch01.html

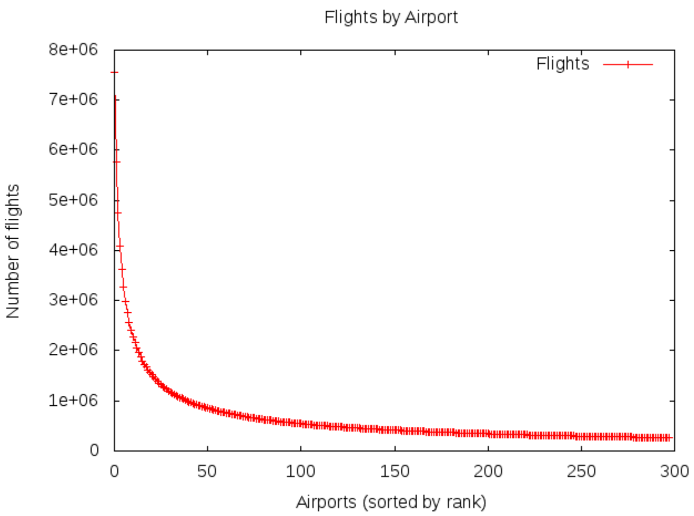
Source code for Group 2 problems: https://github.com/gersakbogdan/coursera/tree/master/Cloud%20Computing%20Capstone/Group%202

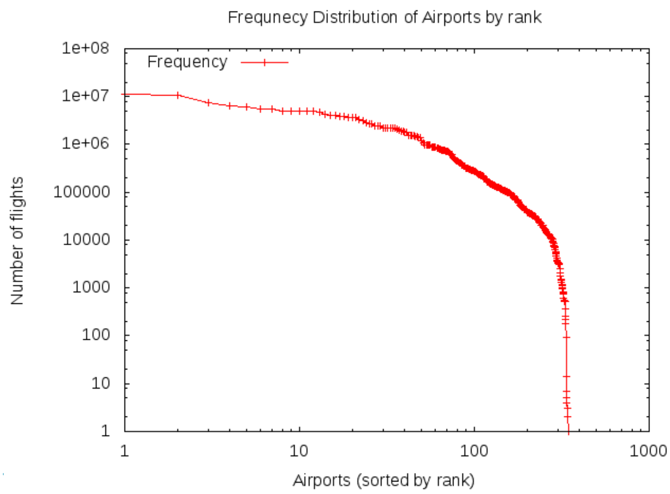
Group 3

1. Does the popularity distribution of airports follow a Zipf distribution? If not, what distribution does it follow?
2. Tom wants to travel from airport X to airport Z. However, Tom also wants to stop at airport Y for some sightseeing on the way. More concretely, Tom has the following requirements (see Task 1 Queries for specific queries):
   A. The second leg of the journey (flight Y-Z) must depart two days after the first leg (flight X-Y). For example, if X-Y departs January 5, 2008, Y-Z must depart January 7, 2008.
   B. Tom wants his flights scheduled to depart airport X before 12:00 PM local time and to depart airport Y after 12:00 PM local time.
   C. Tom wants to arrive at each destination with as little delay as possible.

Question 3.1 is all about analysis of data report. Zipf's law stats that given some corpus or natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. In our case it means that the airport with a higher rank should have a double number of flights compare with the next airport in rank.

Even if from the Flights by Airport figure we can hope for a Zipf distribution, after doing a log-log plot we can see that Airports rank by number of flights is not following this distribution (log log for Zipf should be a straight line and is not). Using special statistical tools (R, Python) it can be prove that this distribution is not Zipf but more a Lognormal one since the bottom half look very different than the top half.





Question 3.2 can be solved by creating two MapReduce Jobs. One which filter the flights by arrival performance and the second one which computes the options Tom has and pick up the best one based on arrival performance.
The important step here is the way we choose the keys for map reduce. For the first map reduce I used a key based on Origin, Destination and FlightDate because even if there are more flights from X to Y in that day we are interested only in that one which has the best arrival time. A big help here was also the condition of flight time (before 12 and after 12) because we can be sure from here that a flight can be use only as a first flight or as a second flight but not for both (this add an extra value to our key). In the second MapReduce we will receive a list of best arrival flights grouped by Origin-Destination-Date and from here using the 12h departure condition we can generate for each day all the options Tom has.

Saving the results into DynamoDB was not easy because there were a lot of rows to insert and some tuning was necessary. Setting the Write Capacity Units for DynamoDB was not enough because after some point, in my case after 250units, Hadoop Cluster and the method I used for insert (custom output format) cannot perform better. In the next few days I will investigate to find where exactly the bottleneck is and how it can be fixed.

Tom flights Results:

Airports	Date	Flights
CMI-ORD-LAX	2008-03-04	4401, 557
JAX-DFW-CRP	2008-09-09	845, 3627
SLC-BFL-LAX	2008-04-01	3755, 5429
LAX-SFO-PHX	2008-07-12	889, 2645
DFW-ORD-DFW	2008-06-10	1104, 6119
LAX-ORD-JFK	2008-01-01	944, 918

Opinion

I think the results are really useful and I plan to not stop here and create an Online System from where you can query this type of data. So looking forward for Task 2.
Overall I was a great experience, hard but great.

Results

Group 1 (HDFS)

Top 10 Airlines

HA	3.9723442
AQ	5.0444994
PS	5.672567
ML (1)	8.703598
WN	9.095804
F9	9.933341
PA (1)	10.344809
US	10.465591
NW	10.548275
OO	10.657584

Top 10 Most Popular

SFO	5050872
MSP	5073589
IAH	5400340
DTW	5491596
DEN	6169795
PHX	6494512
LAX	7574328
DFW	10562405
ATL	11301229
ORD	12020931

Group 2.1 (DynamoDB)

Airport	Carrier	Departure Delay (avg)
CMI	DH	9.707415
CMI	EV	9.69266
CMI	MQ	11.731388
CMI	OH	5.367574
CMI	PI	4.4373507
CMI	TW	3.7823129
CMI	US	2.6746335
BWI	AA	7.569195
BWI	CO	7.077083
BWI	DL	8.779586
BWI	EA	8.6006565
BWI	F9	4.9158316
BWI	NW	8.260861
BWI	PA (1)	5.9428573
BWI	TW	9.058592
BWI	US	8.482188
BWI	YV	7.6819596
MIA	9E	0.6666667
MIA	EV	5.6696033
MIA	ML (1)	7.639676
MIA	NW	6.9137883
MIA	PA (1)	4.6715474
MIA	PI	8.402743
MIA	TZ	6.8230352
MIA	UA	8.214287
MIA	US	7.356537
MIA	XE	6.0986886

Airport	Carrier	Departure Delay (avg)
LAX	AA	8.378402
LAX	F9	8.354594
LAX	FL	8.0514965
LAX	ML (1)	6.94721
LAX	MQ	5.0626464
LAX	NW	7.226622
LAX	OO	6.087762
LAX	PS	4.9191217
LAX	TZ	7.4530783
LAX	US	7.787091
IAH	AA	7.183435
IAH	DL	8.24734
IAH	HP	8.0394
IAH	NW	6.1125956
IAH	OO	7.9388795
IAH	PA (1)	5.657978
IAH	PI	4.606278
IAH	TW	7.432934
IAH	US	7.02648
IAH	WN	6.2142253
SFO	CO	8.840944
SFO	DL	7.7030063
SFO	F9	8.906127
SFO	MQ	8.045227
SFO	NW	7.907103
SFO	PA (1)	6.1363635
SFO	PS	6.3918405
SFO	TW	8.647407
SFO	TZ	7.6628203
SFO	US	8.529584

Group 2.2 (DynamoDB)

1. Origin	2. Destination	3. Departure Delay (avg)
CMI	PIT	2.173913
CMI	DAY	3.4394906
CMI	PIA	3.453552
CMI	STL	3.8784971
CMI	CVG	6.3914895
CMI	DFW	9.590446
CMI	ATL	9.69266
CMI	ORD	11.9128065

BWI	MLB	2.390935
BWI	IAD	3.097902
BWI	DAB	3.8508475
BWI	SRQ	4.2281632
BWI	UCA	4.6114526
BWI	CHO	4.826087
BWI	MDT	4.9014306
BWI	BGM	5.055007
BWI	OAJ	5.3214684
BWI	GSP	5.4288726

MIA	BUF	2
MIA	SAN	2.5136611
MIA	HOU	3.618392
MIA	SLC	3.9502075
MIA	ISP	4.4502926
MIA	PSE	4.94686
MIA	GNV	4.97379
MIA	MCI	5.360544
MIA	TLH	5.416809
MIA	MEM	6.1854177

1. Origin	2. Destination	3. Departure Delay (avg)
LAX	SDF	0
LAX	BZN	1
LAX	VIS	2.4805195
LAX	PMD	3
LAX	IYK	3.6435883
LAX	SNA	3.994529
LAX	MEM	4.117761
LAX	CLD	4.594013

1. Origin	2. Destination	3. Departure Delay (avg)
IAH	MSN	0
IAH	HOU	2.3019054
IAH	AGS	2.8345945
IAH	EFD	3.9198737
IAH	VCT	5.319814
IAH	RNO	5.5143776
IAH	MTJ	5.586871
IAH	SNA	5.8282757
IAH	JAC	5.8869047

1. Origin	2. Destination	3. Departure Delay (avg)
SFO	SDF	0
SFO	MSO	0.5833333
SFO	LGA	1.2121212
SFO	OAK	2.5501559
SFO	PIE	2.7283237
SFO	BNA	3.0175695
SFO	SCK	4
SFO	MKE	5.142407
SFO	MEM	5.399692

Group 2.3 (DynamoDB)

```
-----------------------------------
|             Query               |
+-------------+-------------------+
| 1. Origin   | 2. Destination    |
+-------------+-------------------+
| CMI         | ORD               |
+-------------+-------------------+
||         3. Carrier            ||
|+-------------------------------+|
|||             MQ              |||
||+-------+---------------------+||
||| S     | 15.739151           |||
||+-------+---------------------+||
```

```
-----------------------------------
|             Query               |
+-------------+-------------------+
| 1. Origin   | 2. Destination    |
+-------------+-------------------+
| ATL         | PHX               |
+-------------+-------------------+
||         3. Carrier            ||
|+-------------------------------+|
|||             DL              |||
||+-------+---------------------+||
||| S     | 13.867261           |||
||+-------+---------------------+||
|||             EA              |||
||+-------+---------------------+||
||| S     | 14.008674           |||
||+-------+---------------------+||
|||             FL              |||
||+---------+-------------------+||
||| S       | 12.61             |||
||+---------+-------------------+||
|||             HP              |||
||+-------+---------------------+||
||| S     | 13.367141           |||
||+-------+---------------------+||
|||             US              |||
||+-------+---------------------+||
||| S     | 12.687395           |||
||+-------+---------------------+||
```

```
-----------------------------------
|             Query               |
+-------------+-------------------+
| 1. Origin   | 2. Destination    |
+-------------+-------------------+
| IND         | CMH               |
+-------------+-------------------+
||         3. Carrier            ||
|+-------------------------------+|
|||             AA              |||
||+-------+---------------------+||
||| S     | 8.25                |||
||+-------+---------------------+||
|||             CO              |||
||+-------+---------------------+||
||| S     | 4.394164            |||
||+-------+---------------------+||
|||             DL              |||
||+-------+---------------------+||
||| S     | 12.629807           |||
||+-------+---------------------+||
|||             EA              |||
||+-------+---------------------+||
||| S     | 13.06542            |||
||+-------+---------------------+||
|||             HP              |||
||+-------+---------------------+||
||| S     | 7.990588            |||
||+-------+---------------------+||
|||             NW              |||
||+-------+---------------------+||
||| S     | 7.6015387           |||
||+-------+---------------------+||
|||             US              |||
||+-------+---------------------+||
||| S     | 7.8385878           |||
||+-------+---------------------+||
```

Query

1. Origin	2. Destination
DFW	IAH

3. Carrier	S
AA	12.147884
CO	10.000648
DL	10.204433
EV	10.691978
MQ	12.975918
OO	9.736549
PA (1)	9.333333
UA	8.899408
XE	12.892918

1. Origin	2. Destination
LAX	SFO

3. Carrier	S
AA	12.465499
CO	14.0017395
DL	13.4838505
EV	13.398714
F9	6.96531
MQ	10.933456
NW	12.790287
PS	5.830403
TZ	6.2380953
US	10.821993

Query

1. Origin	2. Destination
JFK	LAX

3. Carrier	S
AA	15.044821
DL	16.631231
HP	14.865142
PA (1)	17.093708
TW	18.287762
UA	11.469386