Advanced topics in database systems

Szymon Kubiszewski

GITHUB: [HTTPS://GITHUB.COM/KSZMMNN/DATABASES](https://github.com/kszmmnn/databases)

The goal of the project was to create a cluster of machines. All of them were configured to run Hadoop, which was used to provide distributed filesystem (HDFS). All of the machines were set so as to be able to run Apache Spark platform, which was used to process the data.

Machines and software

In my solution, I used three virtual machines (master, slave1, slave2) ran locally using Oracle VM VirtualBox hypervisor. Every virtual machine (VM) was using Ubuntu 18.04 operating system. All instances were using Python 3.6, Java JDK 8, Hadoop 2.7.0 and Apache Spark 3.1.3. Every VM was connected both to virtual local network and Internet network. In order to allow easier programming also Jupyter notebook was installed on the master VM, accessible from hosting computer web browser.

Virtual machines local IP addresses

- master - 192.168.56.107
- slave1 - 192.168.56.108
- slave2 - 192.168.56.109

Configuration

In order to properly set whole cluster several configuration steps were performed. Every VM has two user accounts, but all of the configurations (and runs) described were made using account ***hadoop*** password ***root***. Files */etc/hostname* and */etc/hosts* were modified, setting hostnames according to described higher hosts and corresponding IPs. Hadoop namenode runs on *master* whereas datanodes runs on *slave1* and *slave2*. Apart from global variables, PATH, etc. few modifications in files were made in order to configure the nodes.

To configure Hadoop:

- */usr/local/hadoop/etc/hadoop/core-site.xml*
- */usr/local/hadoop/etc/hadoop/hdfs-site.xml*
- */usr/local/hadoop/etc/hadoop/slaves*
- */usr/local/hadoop/etc/hadoop/yarn-site.xml*
- */usr/local/hadoop/etc/hadoop/mapred-site.xml*

To configure Spark:

- */home/hadoop/spark/conf/spark-defaults.conf*
- */home/hadoop/spark/conf/spark-env.sh*

Which followed with following services:

- Spark Master at http://master:8080/
- Spark History Server at http://master:18080/
- Hadoop Web UI at http://master:50070/
- YARN UI at http://master:8088/
- Jupyter notebook (after call *jupyter-notebook*) at http://master:8888/

Instructions

After starting and logging to every VM (default to machine-specific user), changing accounts to *hadoop* user might be needed (password root). As a *hadoop* user at *master* machine in home directory there is a script *startal1.sh* that starts Spark Master, Hadoop dfs (namenode at master and datanodes at slaves) and Spark History Server. In the same directory there is a script *stopal.sh* stopping all of the services run by *startal1.sh*.

In order to run a Spark worker run of command on one of slave nodes is needed. For instance for slave1:

> *spark-daemon.sh start org.apache.spark.deploy.worker.Worker 1 --webui-port 8080 --port 65509 --cores 8 --memory 8g spark://master:7077*

And for slave2:

> *spark-daemon.sh start org.apache.spark.deploy.worker.Worker 1 --webui-port 8080 --port 65510 --cores 8 --memory 8g spark://master:7077*

To stop a worker, calling the command *~/spark/sbin/stop-worker.sh* is needed.

To run implemented solutions, you need to run script located in home of hadoop user on machine master

> *python task.py*

which will execute the script, and display the results. You can as well run it from jupyter notebook, the file for it is *project.ipynb* located in the same directory.

In both cases the results will be printed, and saved in hdfs directory */user/hadoop/out/**.

Questions and answers

As also written in solution files the questions (and my assumptions below) were:

- Q1 To find the route with the biggest tip in March and arrival point "Battery Park".
- Q2 To find, for each month, the route with the highest amount in tolls. Ignore zero amounts.
    - Here I suppose, that I need to find 6 highest amounts, zeroes (and negatives) excluded.
- Q3 To find, per 15 days, the average distance and cost for all routes with a point of departure different from the point of arrival.
    - Per 15 days, meaning whichever 15 day so I will consider first 15 days (dropoffs) of January 2022.
    -
- Q4 To find the three biggest (top 3) peak hours per day of the week, meaning the hours (eg, 7-8am, 3-4pm, etc) of the day with the highest number of passengers in a taxi race. The calculation concerns all months.

    - I understand the task this way: I am supposed to find top 3 peak hours per day, meaning 3*7 values in total. Each day of week will be considered in group: all Mondays together, all Tuesdays etc. It is a sum of passengers in an hour, but each day is summed separately.

- Q5 Find the top five (top 5) days per month in which the races had the highest tip percentage. For example, if the ride cost $10 (fare_amount) and the tip was $5, the percentage is 50%.

Input data is located in HDFS, inside directory */user/hadoop/intask/* and */user/hadoop/intaskcsv/*. Output data is located also in HDFS, inside directory */user/hadoop/out/**.

Questions Q1-5 were solved using Pyspark DataFrame API, additionaly Q3 was solved using Pyspark RDD API. Everytime queries were ran using every available processor, summed up, and divided by number of runs which was equal to 10 in every case. Computer was not doing any other - apart from necessary - operations. As some of the steps were identical for tasks (loading files etc.) they are not added to exequtions time.

Time results (1w stands for 1 Spark worker, 2w for 2 Spark workers):

Workers were assigned with 8 cores (out of 8) and 8GB of RAM (out of 32 GB) each.

case	average execution time [s]
Q1 1w	3.828761226
Q1 2w	4.736980534
Q2 1w	47.58168952
Q2 2w	49.78992596
Q3 1w RDD	441.9351444
Q3 2w RDD	498.0638633
Q3 1w DF	11.13144054
Q3 2w DF	13.13868236
Q4 1w	8.202548340
Q4 2w	9.824843311
Q5 1w	20.64261550
Q5 2w	39.92332291

From what can be seen in the table, generally speaking 2 workers did not made the time of execution smaller. Increasing the amount of workers increased execution times as well. This situation for sure comes from the fact, that whole cluster was run using virtual machines, and not with real-world equipment. Because of that increasing the amount of machines in reality decreased the usable percentage of hardware towards solving given tasks, as some of the processor time was used for context switching instead of running the tasks. This situation would be for sure different, if with running more workers the sum of accessible hardware was increased.

What can be also seen RDD API is much slower than that of DataFrame. What is more, DF was much easier to both use and verify my solutions.

Answers for questions

Q1

Answers are in file q1.csv attached to the report.

Q2

Answers are in file q2.csv attached to the report.

Q3

For first 15 days of 2022 year the results are:

- avg(trip_distance) = 5.576670281995658
- avg(total_amount) = 19.89843362459359

Q4

day	hour	max_sum	rank
1	17	10973.0	1
1	18	10837.0	2
1	1	10623.0	3
6	21	14296.0	1
6	22	13744.0	2
6	20	13264.0	3
3	21	13320.0	1
3	20	12232.0	2
3	22	12111.0	3
5	21	13507.0	1
5	20	13161.0	2
5	22	12626.0	3
4	20	13246.0	1
4	21	13244.0	2
4	22	13007.0	3
7	21	12792.0	1
7	22	12332.0	2
7	20	12325.0	3
2	21	12176.0	1
2	20	11591.0	2
2	22	10817.0	3

Q5

tpep_dropoff_datetime	tip_amount	fare_amount	tip_percentage	rank
2022-04-12 14:47:13	225.0	0.01	2250000.0	1
2022-04-02 20:31:43	125.0	0.01	1250000.0	2
2022-04-21 01:49:12	120.0	0.01	1200000.0	3
2022-04-12 10:54:15	95.0	0.01	950000.0	4
2022-04-03 09:36:29	40.0	0.01	400000.0	5
2022-06-25 13:01:53	155.0	0.01	1550000.0	1
2022-06-13 22:21:23	150.0	0.01	1500000.0	2
2022-06-10 01:32:27	65.0	0.01	650000.0	3
2022-06-16 04:41:11	57.0	0.01	570000.0	4
2022-06-13 00:49:36	52.0	0.01	520000.0	5
2022-02-21 23:18:36	45.0	0.01	450000.0	1
2022-02-13 07:33:56	29.69	0.01	296900.0	2
2022-02-09 20:01:39	25.0	0.01	250000.0	3
2022-02-27 17:54:33	25.0	0.01	250000.0	3
2022-02-24 11:57:17	15.0	0.01	150000.0	5
2022-05-12 03:28:16	75.0	0.01	750000.0	1
2022-05-12 11:43:29	74.0	0.01	740000.0	2
2022-05-20 01:21:10	65.0	0.01	650000.0	3
2022-05-15 04:25:00	20.0	0.01	200000.0	4
2022-05-16 05:10:46	20.0	0.01	200000.0	4
2022-07-01 00:00:00	2.0	14.0	14.285714285714285	1
2022-03-18 01:08:56	100.0	0.01	1000000.0	1
2022-03-21 18:57:48	70.0	0.01	700000.0	2
2022-03-05 03:15:12	20.0	0.01	200000.0	3
2022-03-26 05:22:57	20.0	0.01	200000.0	3
2022-03-18 19:45:30	12.5	0.01	125000.0	5
2022-01-09 03:07:18	168.88	0.01	1688800.0	1
2022-01-31 12:23:44	110.0	0.01	1100000.0	2
2022-01-31 15:43:58	63.0	0.01	630000.0	3
2022-01-01 17:18:50	50.0	0.01	500000.0	4
2022-01-03 18:02:50	25.0	0.01	250000.0	5

Results

The results descirbed in performance section were predictable, as using more layered software solution (in this case, because of VMs) was not correlated with increasing the amount of hardware.

Overall the - from my point of view - the project was really interesting and I learnt a lot both about large scale data processing and configuration of machines to run the demanded software.