Comparing Performance of Hadoop MapReduce Engine and Spark Engine using HiveQL

COEN: 283 Operating System

Santa Clara University

Harshil Shah

Rasika Telang

Siddhant Sehgal

Utsav Kanpara

ACKNOWLEDGEMENT

The accomplishment of any project relies upon commitment of group and guidance by others. We accept this as an open door to offer our thanks to the people who have been instrumental in the effective fulfillment of this project, particularly Prof. Farokh Eskafi, for his direction all through the term.

Table of content

1. Introduction

Problem Statement:

This is report is compare and analyze the performance between Hadoop Mapreduce engine and Spark using Hive query language(HiveQL) as a structured query language while fetching structured data from hadoop distributed File System (HDFS) in Single-node/Multi-node Hadoop cluster.

Technical Specification:

Operating System: Ubuntu 14.04

RAM: 1GB

Hadoop: Version 2.8.5

Spark: Version 2.3.2

Disk: 8GB

Related Work:

Recently there have been a lot of comparison, where numerous researchers have attempted to look at the performance of Spark and Hadoop MapReduce using different techniques. There was a research conducted in PU(Pacific University) by a group of student, where they tried to compare Hadoop MapReduce, Spark Technology and BDMP (Big Data Management Processing). The students came up with a conclusion that Apache Spark is more faster in analysing data when compared to MapReduce.

Another research, in order to test the performance of Hadoop MapReduce, some researchers made the use of HiBench benchmarking suite. The HiBench benchmark suite is built by Intel, it is a big data benchmark suite that helps evaluate different big data frameworks in terms of speed, throughput and system resource utilizations. The HiBench shell scripts are certified and published under

Apache. With the use of HiBench benchmark suite they came to a conclusion that Apache Spark is faster than Hadoop MapReduce. As, in Hadoop,after the reduce or map action it endures the data back to the disk whereas Apache Spark processes the data in memory.

2. Hadoop Overview

Hadoop:

- It is an open-source software that facilitates using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model

- Hadoop splits files into large blocks and distributes them across nodes in a cluster.

- The center of Apache Hadoop comprises of a storage part, known as Hadoop Distributed File System (HDFS), and a processing unit which is a MapReduce programming model. Hadoop parts documents into substantial squares and disperses them crosswise over nodes in a cluster. It at that point moves bundled code into nodes to process the information in parallel. This methodology exploits information locality, where nodes control the information they have access to. This permits the dataset to be prepared quicker and more effectively than it would be in a more customary supercomputer engineering that depends on a parallel record framework where calculation and information are disseminated by means of rapid networking. Basically the Hadoop bunch is utilized for storing information that must be investigated and to have the capacity to do that we do Mapper Reducer i.n. MapReduce programming to break down the information stored in a Hadoop Cluster.

- Hadoop is based on the following modules:-

    - Hadoop Distributed File Systems(HDFS) - a distributed file-system that stores data
    - MapReduce - processing and programming model

  - YARN - a resource manager responsible for managing and computing resources in clusters and using them for scheduling purposes
  - Hadoop Common - contains libraries and utilities needed by other Hadoop modules.

Hadoop distributed file system

- The HDFS is a conveyed, adaptable, and portable document framework written in Java for the Hadoop system. Some consider it to rather be an information storage system because of its absence of POSIX compliance, however it provides shell directions and Java application programming interface (API) strategies that are like other document systems. A Hadoop is isolated into two HDFS and MapReduce. HDFS is utilized for storing the information and MapReduce is utilized for the Processing the Data.
- HDFS has two administrations as pursues:
    1. Name Node
    2. Secondary Name Node

- Top is a Master Services/Demons/Nodes and second is a Slave Services. Master can speak with one another and similarly Slave administrations can speak with one another. Name Node is a master node and Data node is its corresponding Slave node and can talk with each other.

- Name Node: HDFS comprises of just a single Name Node we call it as Master Node which can follow the records, deal with the document framework and has the meta information and the entire information in it. To be specific Name node contains the subtle elements of the No. of blocks, Locations at what information node the information is put away and where the replications are put away and different subtle elements. As we have just a single Namenode we call it as Single Point Failure. It has Direct associate with the customer.

- Information Node: A Data Node stores information in it as the blocks. This is otherwise called the slave node and it stores the genuine information into HDFS which is in charge of the customer to read and write. These are slave nodes. Each data node sends a Heartbeat message to the Namenode every 3 second like clockwork and passes on that it is alive or not. Along these lines when Namenode does not get a heartbeat from an information node for 2 minutes, it will accept that information node as dead and begins the procedure of block replications on some other Data node.

Hadoop Ecosystem

1.Apache Hive

- Apache Hive is an open source data warehouse system for querying and analyzing large data sets that are principally stored in Hadoop files. It is commonly a part of compatible tools deployed as part of the software ecosystem based on the Hadoop framework for handling large data sets in a distributed computing environment.
- Like Hadoop, Hive has roots in batch processing techniques. It was originated in 2007 by developers at Facebook who sought to provide SQL access to Hadoop data for analytics users. Like Hadoop, Hive was developed to address the need to handle petabytes of data accumulating via web activity.

2.Apache Pig

- **Apache Pig** is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.
- At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject).

3.Apache Oozie

- Oozie is a workflow scheduler system to manage Apache Hadoop jobs.Oozie Workflow jobs are Directed Acyclic Graphs (DAGs) of actions.Oozie Coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availability.
- Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system specific jobs (such as Java programs and shell scripts).Oozie is a scalable, reliable and extensible system.

4.Apache Sqoop

- Sqoop is a command-line interface that facilitates moving bulk data from Hadoop into relational databases and other structured data stores. Using Sqoop replaces the need to develop scripts to export and import data. One common use case is to move data from an enterprise data warehouse to a Hadoop cluster for ETL processing. Performing ETL on the commodity Hadoop cluster is resource efficient, while Sqoop provides a practical transfer method.

5.Apache Mahout

- Apache Mahout is a distributed linear algebra framework and mathematically expressive Scala DSL designed to let mathematicians, statisticians, and data scientists quickly implement their own algorithms. Apache Spark is the recommended out-of-the-box distributed back-end, or can be extended to other distributed backends.
- Mathematically Expressive Scala DSL.
- Support for Multiple Distributed Backends (including Apache Spark)
- Modular Native Solvers for CPU/GPU/CUDA Acceleration.

3. **Map Reduce**

Hadoop MapReduce is a product structure for effortlessly composing applications which process huge measures of information (multi-terabyte informational collections) in-parallel on huge bunches (a huge number of nodes) of ware equipment in a solid, blame tolerant way.

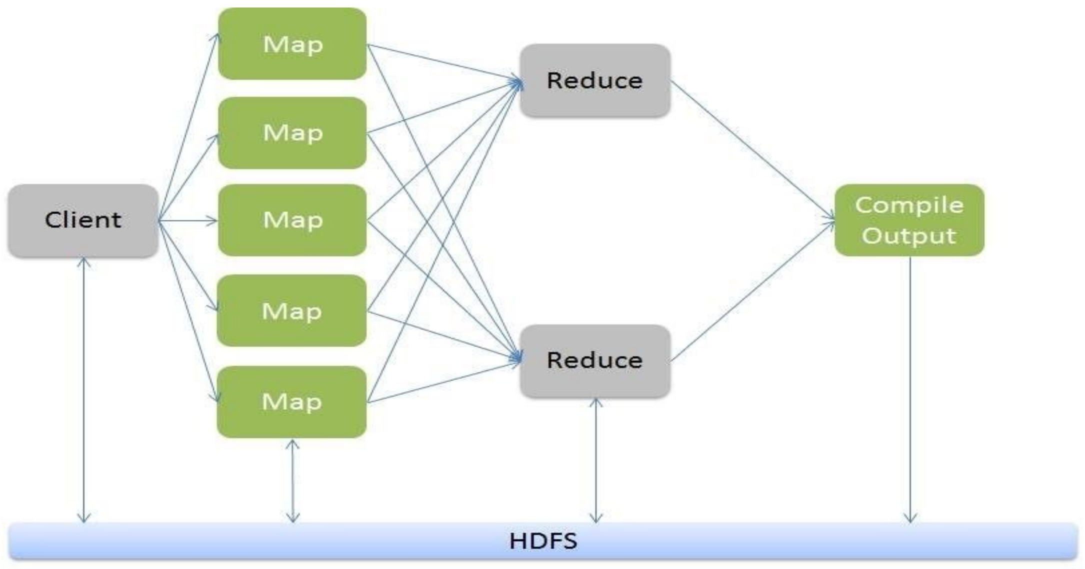
A MapReduce work ordinarily splits the information informational collection into independent pieces which are prepared by the guide assignments in a totally parallel way. The system sorts the yields of the maps, which are then contribution to the decrease errands. Regularly both the information and the yield of the activity are put away in a document framework. The system deals with booking undertakings, observing them and re-executes the fizzled assignments.

Commonly the process nodes and the capacity nodes are the equivalent, that is, the MapReduce structure and the Hadoop Distributed File System (see HDFS Architecture Guide) are running on a similar arrangement of nodes. This setup enables the structure to successfully plan assignments on the nodes where information is now present, bringing about high total data transfer capacity over the bunch.

The MapReduce structure comprises of a solitary ace JobTracker and one slave TaskTracker per group node. The master is in charge of planning the occupations' segment undertakings on the slaves, observing them and re-executing the fizzled jobs. The slaves execute the jobs as coordinated by the master.

For example, consider a MapReduce job that counts the number of times each word is used across a set of documents. Input file is splitted and replicated across various nodes. The map phase counts the words in each document.

- Map task:
    - Map task will convert these tuples into another set of tuples.
    - Mapper will transform file data to smaller, intermediate <key, value> pairs.
- Then the reduce phase aggregates the per-document data into word counts spanning the entire collection.These tuples are passed to mapper to perform map task.
- Reduce task:
    - Intermediate data from mapper will be input to reducer.
    - Two tasks are performed: Shuffling and Sorting
        - Shuffling : Rearranges intermediate pairs according to keys of tuples.
        - Sorting : Shuffled data tuples are sorted to get reduced data output.



MapReduce Advantages

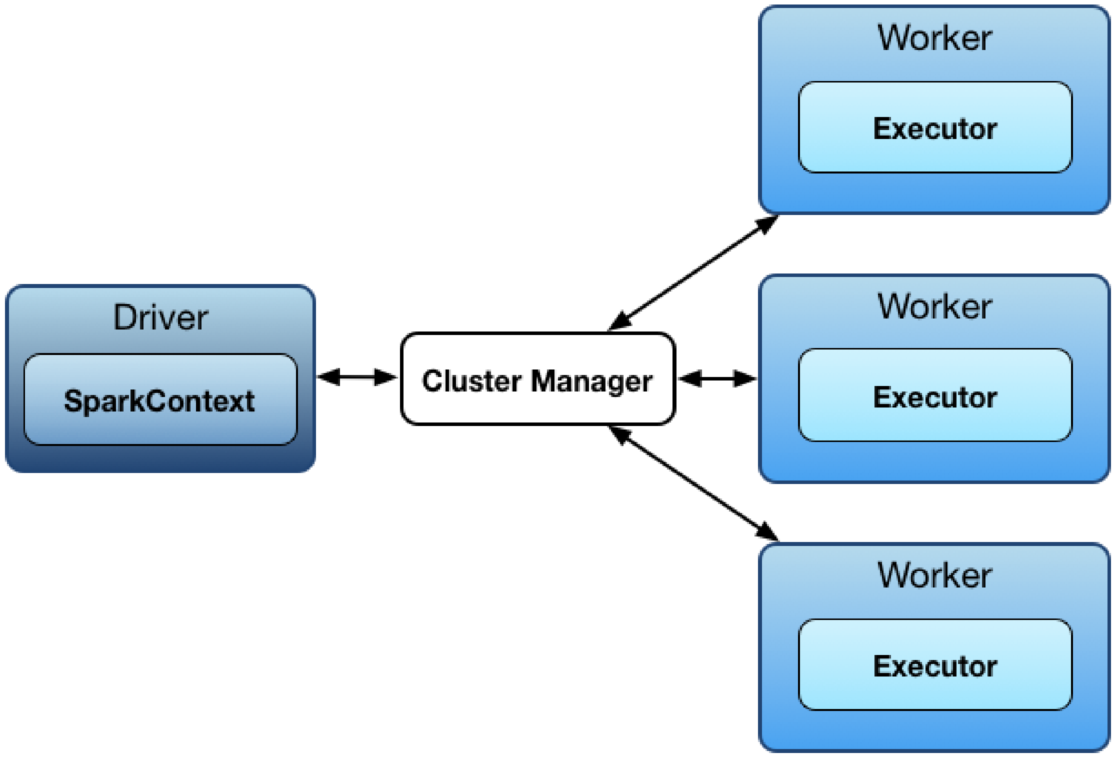
Simplicity	Developers can write applications in their language of choice, such as Java, C++ or Python, and MapReduce jobs are easy to run
Scalability	MapReduce can process petabytes of data, stored in HDFS on one cluster
Speed	Parallel processing means that MapReduce can take problems that used to take days to solve and solve them in hours or minutes
Recovery	MapReduce takes care of failures. If a machine with one copy of the data is unavailable, another machine has a copy of the same key/value pair, which can be used to solve the same sub-task. The JobTracker keeps track of it all.
Minimal data motion	MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and contributes to Hadoop's processing speed.

4. Spark

Apache Spark is a fast, in-memory data processing engine with elegant and expressive development APIs to allow data workers to efficiently execute streaming, machine learning or SQL workloads that require fast iterative access to datasets. It is also an open source cluster computing framework for running data analytics applications.Like MapReduce, it processes data in parallel.Fast in-memory large scale data processing engine.

People are favoring spark over mapreduce more and more day by day mainly because of its performance improvement over mapreduce and also because of the reduced complexity of programing.

Architecture



Spark uses a master/worker architecture. There is a driver that talks to a single coordinator called master that manages workers in which executors run. A Spark driver is a JVM process that hosts SparkContext. It is the master node in a Spark application.

Sparkcontext is the heart of a Spark application.A master is a running Spark instance that connects to a cluster manager for resources.

Workers (aka slaves) are running Spark instances where executors live to execute tasks. They are the compute nodes in Spark.The driver and the executors run in their own Java processes. You can run them all on the same (horizontal cluster) or separate machines (vertical cluster) or in a mixed machine configuration.

The main two parts of Apache Spark are –
Spark Core – A distributed execution engine. Java, Scala, and Python APIs provide the platform which is required for any ETL application development.

Set of Libraries – It helps in streaming, SQL processing and machine learning specific algorithm tasks.
The entire structure is designed for bottom-up performance. Most of the data science specific machines learning algorithms are iterative. When a dataset is cached in memory the data processing speeds up automatically. Also, Apache Spark has this in-memory cache property that makes it faster.

5. Hive

Apache Hive is an information warehouse programming venture based over Apache Hadoop for giving information inquiry and analysis. Hive gives a SQL-like interface to question information put away in different databases and document frameworks that incorporate with Hadoop. Customary SQL inquiries must be actualized in the MapReduce Java API to execute SQL applications and questions over circulated information.

Hive gives the important SQL deliberation to coordinate SQL-like questions (HiveQL) into the hidden Java without the need to execute inquiries in the low-level Java API. Since most information warehousing applications work with SQL-based questioning dialects, Hive helps conveyability of SQL-based applications to Hadoop.

While at first created by Facebook, Apache Hive is utilized and created by different organizations, for example, Netflix and the Financial Industry Regulatory Authority (FINRA). Amazon keeps up a product fork of Apache Hive incorporated into Amazon Elastic MapReduce on Amazon Web Services.

MapReduce, Apache Tez, and Spark employments. Every one of the three execution motors can keep running in Hadoop YARN. To quicken questions, it gives records, including bitmap indexes. Other highlights of Hive include:

- Ordering to give speeding up, list type including compaction and bitmap file as of 0.10, more list types are arranged.
- Distinctive capacity types, for example, plain content, RCFile, HBase, ORC, and others.
- Metadata stockpiling in a social database the board framework, essentially lessening an opportunity to perform semantic checks amid inquiry execution.
- Working on packed information put away into the Hadoop environment utilizing calculations including DEFLATE, BWT, smart, and so forth.
- Worked in client characterized capacities (UDFs) to control dates, strings, and other information mining apparatuses. Hive bolsters stretching out the UDF set to deal with utilizing cases not upheld by implicit capacities.
- SQL-like questions (HiveQL), which are certainly changed over into MapReduce or Tez, or Spark occupations.

6. Testing and Results

Apache Spark is setting the world of Big Data ablaze. With a guarantee of rates up to 100 times faster than Hadoop MapReduce and agreeable APIs, some figure this could be the finish of Hadoop MapReduce.

In what manner can Spark, an open-source information preparing structure, process information so quick? The answer is that it keeps running in-memory on the group, and that it isn't attached to Hadoop's MapReduce two-arrange worldview. This makes rehashed access to similar information a lot quicker. Start can keep running as an independent or over Hadoop YARN, where it can peruse information straightforwardly from HDFS. Organizations like Yahoo, Intel, Baidu, Trend Micro and Groupon are as of now utilizing it.

Advantages of Spark over Hadoop

1. In-memory Computation

Spark is meant to be for 64-bit computers which can handle Terabytes of data in RAM. Spark is designed in a way that it transforms data in-memory and not in disk I/O. Hence, it cut off the processing time of read/write cycle to disk and storing intermediate data in-memory. This reduces processing time and cost of memory at a time. Moreover, Spark supports parallel distributed processing of data, hence almost 100 times faster in memory and 10 times faster on disk.

2. Resilient Distributed Datasets (RDD)

The main abstraction of Apache Spark is Resilient Distributed Datasets (RDD).It is a fundamental data structure of Spark. RDD can be viewed as an immutable distributed collection of objects. These objects can be cached using two methods, either by a cache() or persist().The beauty of storing RDD in memory using cache() method is – while storing the value in-memory if the data doesn't fit it sends the excess data to disk or recalculates it. Basically, it is a logical partitioning of each dataset in RDD which can be computed on different nodes of a cluster. As it is stored in-memory, RDD can be extracted whenever required without using the disks. It makes the processing faster.

3. Ease of Use

Spark follows general programming model. This model does not constrain programmers to design their applications into a bunch of maps and reduce operations. The parallel programs of Spark look very similar to sequential programs, which is easy to develop. Finally, Spark works on a combination of batch, interactive, and streaming jobs in the same application. As a result, a Spark job can be up to 100 times faster and only need 2 to 10 times less code writing.

4. Ability for On-disk Data Sorting

Apache Spark is the largest open source data processing project. It is fast when stores large scale of data on disk. Spark has the world record of on-disk data sorting.

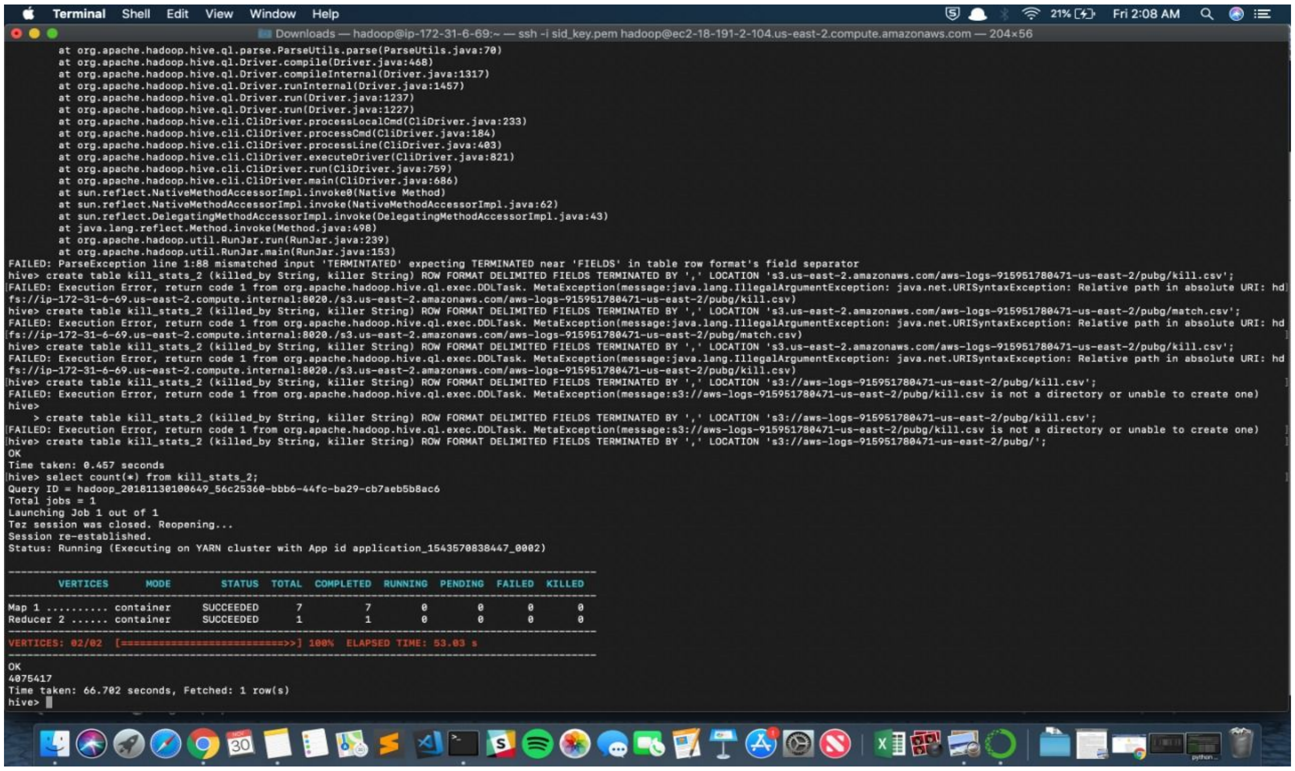
5. DAG Execution Engine

DAG or Direct Acrylic Graph allows the user to explore each stage of data processing by expanding detail of any stage. Through a DAG user can get a stage view which clearly shows the detail view of RDDs. Spark has GraphX, which is a graph computation library and provides inbuilt graph support to improve the performance of machine learning algorithm. Spark uses DAG to perform all required optimization and computation in a single stage rather than going for multiple stages.

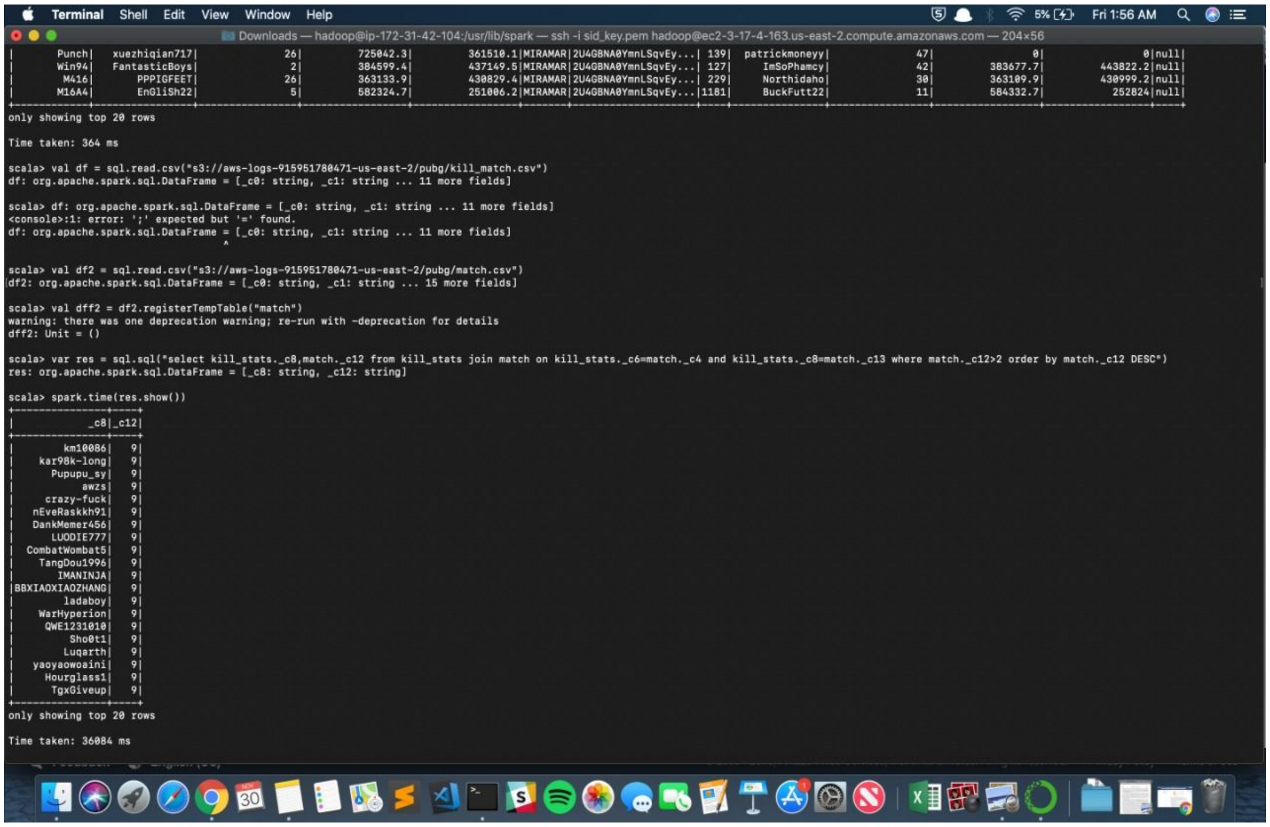
6. Failure Tolerance

MapReduce relies on hard drives; if a process crashes in the middle of execution, it could continue where it left off. Spark will have to start processing from the beginning.

Result Screenshots:

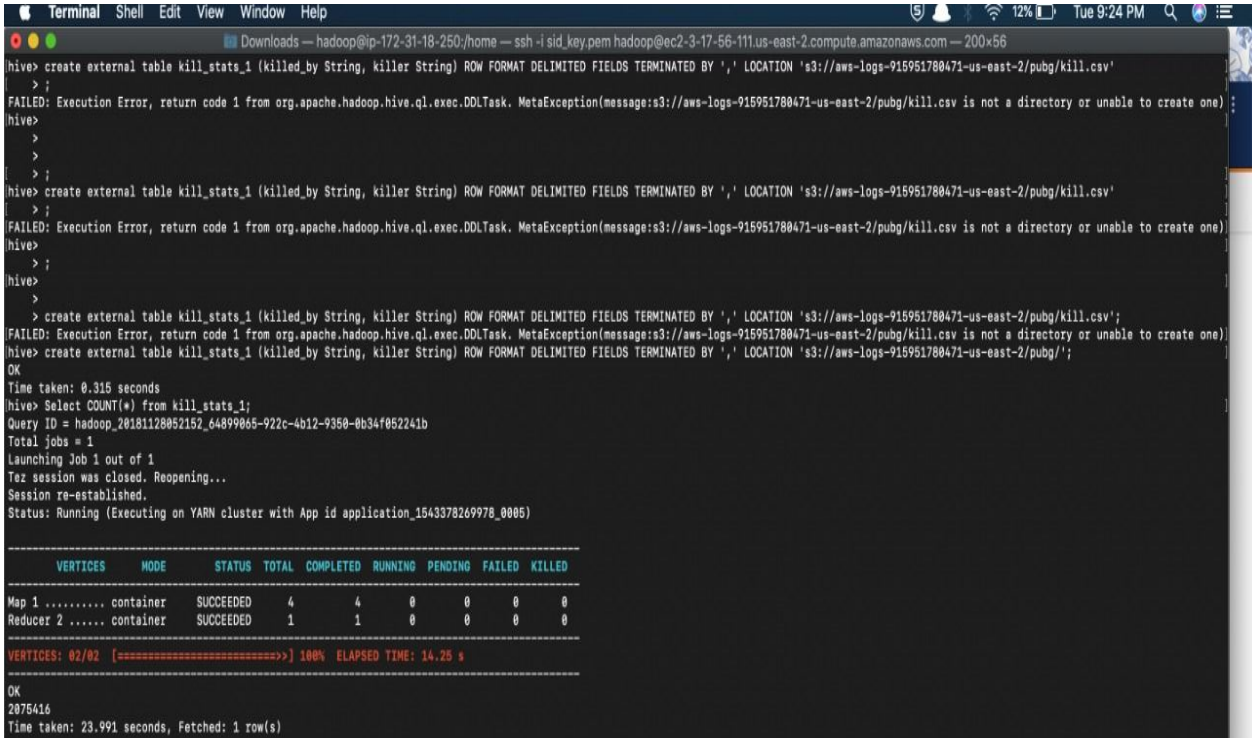

Terminal Shell Edit View Window Help
21% Fri 2:08 AM
Downloads — hadoop@ip-172-31-6-69:~ — ssh -i sid_key.pem hadoop@ec2-18-191-2-104.us-east-2.compute.amazonaws.com — 204×56

```
        at org.apache.hadoop.hive.ql.parse.ParseUtils.parse(ParseUtils.java:70)
        at org.apache.hadoop.hive.ql.Driver.compile(Driver.java:468)
        at org.apache.hadoop.hive.ql.Driver.compileInternal(Driver.java:1317)
        at org.apache.hadoop.hive.ql.Driver.runInternal(Driver.java:1457)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1237)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1227)
        at org.apache.hadoop.hive.cli.CliDriver.processLocalCmd(CliDriver.java:233)
        at org.apache.hadoop.hive.cli.CliDriver.processCmd(CliDriver.java:184)
        at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:403)
        at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java:821)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:759)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:686)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:239)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:153)
FAILED: ParseException line 1:88 mismatched input 'TERMINTATED' expecting TERMINATED near 'FIELDS' in table row format's field separator
hive> create table kill_stats_2 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3.us-east-2.amazonaws.com/aws-logs-915951780471-us-east-2/pubg/kill.csv';
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:java.lang.IllegalArgumentException: java.net.URISyntaxException: Relative path in absolute URI: hd
fs://ip-172-31-6-69.us-east-2.compute.internal:8020./s3.us-east-2.amazonaws.com/aws-logs-915951780471-us-east-2/pubg/kill.csv)
hive> create table kill_stats_2 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3.us-east-2.amazonaws.com/aws-logs-915951780471-us-east-2/pubg/match.csv';
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:java.lang.IllegalArgumentException: java.net.URISyntaxException: Relative path in absolute URI: hd
fs://ip-172-31-6-69.us-east-2.compute.internal:8020./s3.us-east-2.amazonaws.com/aws-logs-915951780471-us-east-2/pubg/match.csv)
hive> create table kill_stats_2 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3.us-east-2.amazonaws.com/aws-logs-915951780471-us-east-2/pubg/kill.csv';
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:java.lang.IllegalArgumentException: java.net.URISyntaxException: Relative path in absolute URI: hd
fs://ip-172-31-6-69.us-east-2.compute.internal:8020./s3.us-east-2.amazonaws.com/aws-logs-915951780471-us-east-2/pubg/kill.csv)
hive> create table kill_stats_2 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://aws-logs-915951780471-us-east-2/pubg/kill.csv';
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:s3://aws-logs-915951780471-us-east-2/pubg/kill.csv is not a directory or unable to create one)
hive>
    > create table kill_stats_2 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://aws-logs-915951780471-us-east-2/pubg/kill.csv';
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:s3://aws-logs-915951780471-us-east-2/pubg/kill.csv is not a directory or unable to create one)
hive> create table kill_stats_2 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://aws-logs-915951780471-us-east-2/pubg/';
OK
Time taken: 0.457 seconds
hive> select count(*) from kill_stats_2;
Query ID = hadoop_20181130100649_56c25360-bbb6-44fc-ba29-cb7aeb5b8ac6
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1543570838447_0002)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 53.03 s
----------------------------------------------------------------------------------------------
OK
4075417
Time taken: 66.702 seconds, Fetched: 1 row(s)
hive> 
```



Terminal Shell Edit View Window Help
Fri 1:56 AM
Downloads — hadoop@ip-172-31-42-104:/usr/lib/spark — ssh -i sid_key.pem hadoop@ec2-3-17-4-163.us-east-2.compute.amazonaws.com — 204×56

```
|       Punch|   xuezhiqian717|                 26|          725042.3|          361510.1|MIRAMAR|2U4GBNA0YmnLSqvEy...| 139|  patrickmoneyy|                 47|                 0|                 0|null|
|       Win94|   FantasticBoys|                  2|          384599.4|          437149.5|MIRAMAR|2U4GBNA0YmnLSqvEy...| 127|     ImSoPhamcy|                 42|          383677.7|          443822.2|null|
|        M416|       PPPIGFEET|                 26|          363133.9|          430829.4|MIRAMAR|2U4GBNA0YmnLSqvEy...| 229|     Northidaho|                 30|          363109.9|          430999.2|null|
|       M16A4|        EnGliSh22|                 5|          582324.7|          251006.2|MIRAMAR|2U4GBNA0YmnLSqvEy...|1181|     BuckFutt22|                 11|          584332.7|            252824|null|
+------------+----------------+-------------------+------------------+------------------+-------+--------------------+----+---------------+-------------------+------------------+------------------+----+
only showing top 20 rows

Time taken: 364 ms

scala> val df = sql.read.csv("s3://aws-logs-915951780471-us-east-2/pubg/kill_match.csv")
df: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 11 more fields]

scala> df: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 11 more fields]
<console>:1: error: ';' expected but '=' found.
df: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 11 more fields]
                                   ^

scala> val df2 = sql.read.csv("s3://aws-logs-915951780471-us-east-2/pubg/match.csv")
df2: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 15 more fields]

scala> val dff2 = df2.registerTempTable("match")
warning: there was one deprecation warning; re-run with -deprecation for details
dff2: Unit = ()

scala> var res = sql.sql("select kill_stats._c8,match._c12 from kill_stats join match on kill_stats._c6=match._c4 and kill_stats._c8=match._c13 where match._c12>2 order by match._c12 DESC")
res: org.apache.spark.sql.DataFrame = [_c8: string, _c12: string]

scala> spark.time(res.show())
+---------------+----+
|            _c8|_c12|
+---------------+----+
|        km10086|   9|
|    kar98k-long|   9|
|      Pupupu_sy|   9|
|           awzs|   9|
|     crazy-fuck|   9|
|    nEveRaskkh91|   9|
|    DankMemer456|   9|
|      LUODIE777|   9|
|   CombatWombat5|   9|
|    TangDou1996|   9|
|       IMANINJA|   9|
|BBXIAOXIAOZHANG|   9|
|        ladaboy|   9|
|    WarHyperion|   9|
|     QWE1231010|   9|
|         Sho0t1|   9|
|        Luqarth|   9|
|    yaoyaowoaini|   9|
|      Hourglass1|   9|
|      TgxGiveup|   9|
+---------------+----+
only showing top 20 rows

Time taken: 36084 ms
```



Terminal Shell Edit View Window Help
Tue 9:24 PM
Downloads — hadoop@ip-172-31-18-250:/home — ssh -i sid_key.pem hadoop@ec2-3-17-56-111.us-east-2.compute.amazonaws.com — 200×56

```
hive> create external table kill_stats_1 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://aws-logs-915951780471-us-east-2/pubg/kill.csv'
    > ;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:s3://aws-logs-915951780471-us-east-2/pubg/kill.csv is not a directory or unable to create one)
hive>
    >
    >
    > ;
hive> create external table kill_stats_1 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://aws-logs-915951780471-us-east-2/pubg/kill.csv'
    > ;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:s3://aws-logs-915951780471-us-east-2/pubg/kill.csv is not a directory or unable to create one)
hive>
    > ;
hive>
    >
    > create external table kill_stats_1 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://aws-logs-915951780471-us-east-2/pubg/kill.csv';
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. MetaException(message:s3://aws-logs-915951780471-us-east-2/pubg/kill.csv is not a directory or unable to create one)
hive> create external table kill_stats_1 (killed_by String, killer String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 's3://aws-logs-915951780471-us-east-2/pubg/';
OK
Time taken: 0.315 seconds
hive> Select COUNT(*) from kill_stats_1;
Query ID = hadoop_20181128052152_64899065-922c-4b12-9350-0b34f052241b
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1543378269978_0005)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      4          4        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 14.25 s
----------------------------------------------------------------------------------------------
OK
2075416
Time taken: 23.991 seconds, Fetched: 1 row(s)
```

7. Conclusion

Hive Query	Single Node MapReduce	Single Node Spark	Multinode MapReduce	Multinode Spark
count(*)	66.72 seconds	36.08 seconds	23.91 seconds	14.13 seconds
Left Outer Join	7 min 50 sec	3 min 27 seconds	1 min 37 seconds	45.72 seconds
Group By	4 min 04 sec	1 min 58 seconds	49.179 seconds	25.650 seconds

8. Bibliography

- https://hadoop.apache.org/
- https://en.wikipedia.org/wiki/Apache_Hadoop
- https://searchdatamanagement.techtarget.com/definition/Apache-Hive
- https://mahout.apache.org/
- https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
- https://www.xplenty.com/blog/apache-spark-vs-hadoop-mapreduce/
- https://www.whizlabs.com/blog/why-is-apache-spark-faster/
- https://en.wikipedia.org/wiki/Apache_Hive