An Analytical Approach to Data Processing in Hive Using User Defined Functions and HQL

Elena Petrova[1], Lukas Schmidt[2], Anna Kowalska[3]*

[1]Department of Materials Science, Lomonosov Moscow State University, Moscow, Russia
[2]Institute of Theoretical Physics, University of Heidelberg, Heidelberg, Germany
[3]Department of Applied Mathematics, University of Warsaw, Warsaw, Poland

ABSTRACT

Apache Hadoop is an open source framework that deals with the distributed computing of large datasets across a clustered of computers using low-cost hardware and simple programming models. Hadoop uses HDFS (Hybrid Distributed File System) for data storage. This allows for distributed processing of large sets across clustered of computers using simple programming languages. Hive is a famous data warehouse package built on top of Hadoop. It provides an SQL tongue called Hive Query Language (HQL) for querying and processing of large sets of data. But the problem with HQL is that it takes more time when queries are applied to convert data in rows to columns i.e horizontal to vertical. This limitation of HQL is improved by using the User Defined Functions (UDF) in Hive. With the help of UDF, many calculations that are outside the scope of built-in HQL operations and functions in Hive like query many columns, combine several column values into one and transformations that are taking more time in HQL, can be solved easily. In the proposed research work five different data sets are taken from web for experimental results. Simple queries using UDF and HQL are applied on these data sets. The results obtained on these data sets show that UDF is better than HQL.

Keywords: *Hadoop, Hive, UDF, HQL, Big data, Data Warehousing.*

I. INTRODUCTION

Hadoop is used for processing data mining, data analytics, decision making and in many research organizations. A framework Map Reduce is used in Hadoop for job scheduling and clustered resource management [4]. This framework is used to process vast amount of data in a distributed and parallel environment on a clustered of commodity hardware. Hive is a relational data warehouse solution for storing and processing large sets of data residing in a distributed storage system Hadoop [5]. Hive uses SQL like language called Hive Query Language (HQL). Hive supports many formats like Text, ORC, RC and Sequence [9]. Hive store metadata in RDBMS which reduces the time to perform semantic checks during query execution. Hive run both high performance native code written in C++ and Java based Hive UDF that might already have written [6]. The code either a scalar function for producing results one row at a time or more complex aggregated functions for doing analysis that used in this paper. HiveServer2 is improved version of HiveServer that support a new thrift API tailored for JDBC and ODBC clients [8]. It is a server interface that enables remote clients to execute queries against Hive and retrieve the result. HiveServer2 supports multi client concurrency and authentication [11]. This is designed to provide better support for open API clients like JDBC and ODBC [7]. New CLI in HiveServer2 named Beeline. This paper presents the processing time of UDF and HQL in Hive by using five data sets and by firing same queries on both.

II. BACKGROUND STUDY

Phaneendra S. V. et al., [1] illustrated that in past days the data was not generating at a fast pace and this type of data is easily handled by RDBMS. But the data generated in current scenario are not handled by RDBMS that is called Big Data. They distinguish big data from other data by five dimensions volume, velocity, variety, value, complexity. To handle with the problem of big data they illustrated the concept of Hadoop.

Bifet A. [2] suggested that streaming data analysis is the most appropriate and fastest way to improve performance. Huge amount of data is created at a very pace that is called big data. There are many tools used for mining Big data are apache hadoop, apache big, apache Hive, cascading, scribe, storm, apache hbase, apache mahout, etc. Thus, they instructed that the ability to handle many exabytes of data mainly dependent on existence of rich variety dataset, technique, software framework.

Sagiroglu S. et al., [3] explained the advntages, disadvantages, scope, and characteristics of big data. From this paper it can be concluded that each and every organization can easily handle Big Data if they carefully analyze the

data. Because the data is growing at a very fast pace, so the challenge is to extract a useful information from this data. It can only be possible by carefully analyze the data and by using an appropriate and optimized technique.

III. PROPOSED METHODOLOGY

There are three aspects that will determine the result: 1) Data set 2) Average query time 3) Query Type used in HQL and UDF.

A). Case Environment

The environment provides the details view of platform used in this research. The installation of Hadoop is done for a single node. So, there is Hadoop master and after that installation of Hive is done on Hadoop [10]. Hive and HiveServer2 runs on the top of Hadoop shown in the following figure. The data resides on HDFS in Hadoop. The replication factor set to 3 and block size defined 64 Megabytes.

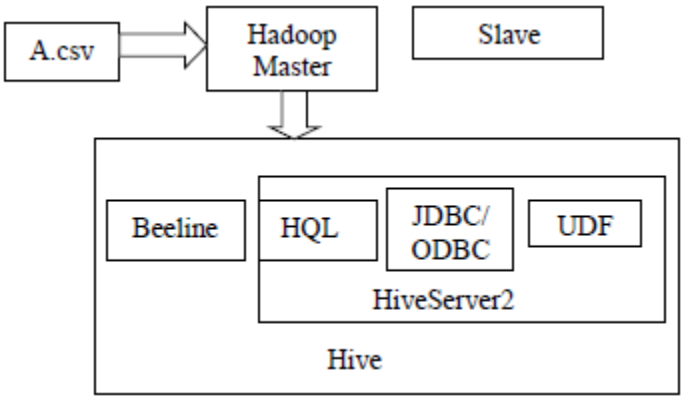

Figure 1.1 Environment used in experimental research

There are some detailed steps that are performed when the processing on data is done.

a) Starting the services of Hadoop and Hive
b) The creation of table is done in Hive which contains 24 columns.
c) The data is load into the table and load this table in the warehouse folder of Hive.
d) UDF is created here and fix the evaluate function by passing as a single argument.
e) Created UDF compiled and make jar file of UDF and upload on the Hive warehouse folder.
f) A new permanent function (grt) is created that linked with UDF by using the properties of Hive.

```
Selectplayerid,yearid,grt(concat(nvl(ing1,0),':',nvl(ing2,0),':',nvl(ing3,0),':',nvl(ing4,0),':',nvl(ing5,0),':',nvl(ing6,0),':',
nvl(ing7,0),':',nvl(ing8,0),':',nvl(ing9,0),':',nvl(ing10,0),':',nvl(ing11,0),':',nvl(ing12,0),':',nvl(ing13,0),':',nvl(ing14,0),'
:',nvl(ing15,0),':',nvl(ing16,0),':',nvl(ing17,0),':',nvl(ing18,0),':',nvl(ing19,0))) from tb_cricket;
```

After this, UDF is called by newly created function and a new table is created which contain 3 columns to store the result of this UDF. Here the outer queries are performed. The Map reduce does not run when the process is executed by UDF. But when same process is executes using HQL the Map Reduce process running. Here the queries are fired on HiveServer2. The queries are:

```
Stringsql1="select playerid,
sum(centcnt) cent,
max(cast(maxrun as BIGINT ))maxrun,
min(cast(minrun as BIGINT)) minrun,";
sql1 += " sum(cast(totalrun as BIGINT )) totrun,";
sql1 += " sum(cast(fiftcnt as BIGINT )) totfifties,";
sql1 += " sum(cast(duccnt as BIGINT )) totducs,";
sql1 += " sum(cast(ingplayed as BIGINT )) totings,";
sql1 += " sum(cast(dnpcnt as BIGINT )) totdnp, ";
sql1 +="sum(cast(totalrun as BIGINT ))/sum
(cast(ingplayed as BIGINT )) totavg ";
sql1 += " from (select playerid,ret[0] maxrun,ret[1] centcnt,
```

```
ret[2] minrun, ret[3] totalrun, ret[4] fiftcnt, ret[5] duccnt,
ret[6] ingplayed, ret[7] dnpcnt from (select playerid,
split(runs,':') ret from tb_data1) bar) bar1 group by
playerid";
System.out.println("Running: " + sql1);
ResultSet res = stmt.executeQuery(sql1);F
```

B). Data sets and file size

The dummy cricket data taken from web and the original data sets can be reconstructed to make it easier to analyze. The five data set called F1 contain a total of 50,000 rows and 24 columns. The second data set called F2 contain 10,48,576 rows and 24 columns. The third data set called F3 contain 31,45,728 Rows and 24 columns. The fourth data set called F4 contain 62,91,456 rows and columns. The last and fifth data set called F5 contain 1,14,06,966 crore rows and 24 columns .

C). Average query time

On Each data set the 9 different queries are executed 5 times and an average time can be calculated in both UDF and HQL. The time takes in seconds.

D). Query Type used in UDF and HQL

The query fired in UDF are written in Java and another query fired in HQL but both are calculating the same results and 9 different fields are calculated of individual player which are sorted by player wise are: Total runs, Maximum runs, Minimum runs, Fifties count, Centuries count, Average, Ducks count, Total innings played, Total innings not played.

IV. RESULTS AND DISCUSSIONS

All experiments are performed using virtual machine. System runs on linux with 4GB Ram and i3 processor. The query fired using UDF and HQL on five different data sets are give following results shown in Table 1-5.

Table 1. Result for data set F1

Attempts	UDF Time	HQL Time
1	2.124 sec	5.670 sec
2	2.179 sec	4.690 sec
3	2.131 sec	4.460 sec
4	2.512 sec	4.355 sec
5	2.676 sec	4.900 sec
Average	2.324 sec	4.815 sec

Table 2. Result for data set F2

Attempts	UDF Time	HQL Time
1	9.013 sec	16.970 sec
2	10.171 sec	16.740 sec
3	10.208 sec	16.540 sec
4	10.163 sec	16.750 sec
5	10.105 sec	16.850 sec
Average	9.932 sec	16.77 sec

Table 3. Result for data set F3

Attempts	UDF Time	HQL Time
1	31.093 sec	42.770 sec
2	31.104 sec	42.230 sec
3	31.144 sec	42.510 sec
4	31.992 sec	42.330 sec
5	32.011 sec	42.600 sec
Average	31.468 sec	42.488 sec

Table 4. Result for data set F4

Attempts	UDF Time	HQL Time
1	67.119 sec	81.360 sec
2	62.590 sec	81.240 sec
3	61.013 sec	83.690 sec
4	69.972 sec	98.960 sec
5	69.012 sec	78.890 sec
Average	65.941 sec	84.828 sec

Table 5. Result for data set F5

Attempts	UDF Time	HQL Time
1	114.973 sec	149.150 sec
2	109.367 sec	169.30 sec
3	117.071 sec	122.260 sec
4	114.775 sec	125.410 sec
5	113.784 sec	132.710 sec
Average	113.994 sec	139.766 sec

The table's 1-5 shows the results obtained from five different data sets i.e F1, F2, F3, F4, F5. Five attempts are applied on these five data sets. The same query is fired o the same data sets. UDF & HQL is used to run the query. The average time can be calculated among five attempts to take the appropriate results. The average time calculated in all tables showing the resultant performance of UDF and HQL on five different data sets. So, in each data set the result of UDF is better than HQL as UDF takes less time to executing the query in each case.

V. CONCLUSION

The advantage offering Hadoop is scaling up from single server to thousands of machines with local storage and using inexpensive hardware. From the experimental results it is concluded that in Hive, UDF can overcome HQL on low-cost hardware with basic query. However, this is tested on a low-cost hardware. Performance may change when better hardware is used for certain software.

REFERENCE

1. *Phaneendra S. V. and Reddy E. M., "Big Data- Solutions For RDBMS Problems-A Survey," In International Journal of Advanced Research in Computer and Communication Engineering, 2013, pp. 3686-3691.*
2. *Bifet A., "Mining Big Data In Real Time," Informatica 37, 2012, pp. 15–20.*
3. *Sagiroglu S. and Sinanc D., "Big Data:A Review," In International Conference on Collaboration Technologies and System, 2013, pp. 42-47.*
4. *Kala K. A. and Chitharanjan K., "A Review on Hadoop-HDFS Infrastructure Extensions," In IEEE Information & Communication Technologies (ICT), 2013, pp. 132-137.*
5. *Bakshi K., "Considerations for Big Data: Architecture and Approach," In IEEE Aerospace Conference, 2012, pp. 1-7.*
6. *Bhosale H. S. and Gadekar D. P. , "A Review Paper On Big Data & Hadoop," In International Journal of Scientific and Research Publications, 2013, pp. 1-7.*
7. *Khajuria A. et al.; "Analysis of Bidgata using Apache Hadoop and Map Reduce," In International Journal of Advanced Research in Computer Science and Software Engineering, 2014, pp. 555-560.*
8. *IBM Software Information Management, "Hadoop Core-HDFS, MapReduce, Pig, Hive, and Jaql," 2012, Ch.1-4, pp. 3-20.*
9. *Srakar D., "Microsoft SQL Server 2012 with Hadoop," 2013, Ch.1-4, pp. 3-73.*
10. *P.Subhasri, "Cloud Computing: Security Challenges & Encryption Practices", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, March 2013 ISSN: 2277 128X, pp. 255-259.*
11. *Bamford J. (2012). [Online]. http://www.wired.com/2012/03/ff nsadatacenter/.*