International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 6 222 - 224

Equity Reports Optimization Using Hive, HDFS.

Mr. Prashant R. Mahajan
Department of Computer Engineering,
DGOI,FOE, Daund
Savitribai Phule Pune University,
Pune. India
prashant.it18@gmail.com

Prof. Amrit Priyadarshi
Department of Computer Engineering
DGOI, FOE, Daund
Savitribai Phule Pune University,
Pune, India
amritpriyadarshi@gmail.com

Abstract: Now a day there is much competition in stock market world. Every brokerage firm Want to lesser down the TURN AROUND TIME for their applications so that business users/analyzers could be able provide solutions to their clients within a time frame. Equity clients prefer to do a business with firms which provide the very fast solutions for their queries. To solve clients query users has to fetch the reports from the software system. As stock market is time critical business. Every corporate has minimum time frame to take their business decisions. For taking business decisions in fast way, we should have usable data available within a time frame. With the help of reporting and research applications, it becomes easy to make accurate business data available in fast way.

*****

Introduction:

Reporting application are those which are used by various business users to fetch data from the system. Data fetched is in a predefined format defined by business analyst as per requirements.

At start of the business day, business users have to take many decisions which are depending upon the last day transaction data. To take a effective decisions transaction data should be available in proper format which should be understandable to users.

Many times higher management/clients also demands answers to their queries to business users. Those answers also have to be finding out by analyzing the transactional data reports.

Day today transaction data is stored in traditional database system. When reporting application tries to fetch data from same database, it becomes overhead on database to run a reporting queries as well as transactions activities.

Here in this paper HDFS is used as a data storage system and Hive is used a technology to manipulate the data stored in HDFS.

Steps to be followed:

Daily transactional data is taken from the traditional system database and transferred to HDFS using hive.

1. Base Table Creation:
   Base table is created in hive db using the base file fetched from the traditional database.
   This Base table contains the dump of all data at start if the system development.

```
CREATE TABLE block (
id int,
symbol string,
accountid int,
quantity int,
allocatedquantity int,
price int,
Commission float,
NetTotal Double,
GsTotal Double,
exchangecode string,
isswap int,
isswift int,
status int,
isdeleted int,
isin string,
cusip string,
sedol string,
createddate string,
modifieddate string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\t";
```

2. Increment Table Creation:
   As day today business runs, there will be new incremental data available daily basis.
   This new data should be transferred to incremental table in hive.

```
CREATE EXTERNAL TABLE block_increment (
id int,
symbol string,
accountid int,
quantity int,
allocatedquantity int,
price int,
Commission float,
NetTotal Double,
GsTotal Double,
exchangecode string,
isswap int,
isswift int,
status int,
isdeleted int,
isin string,
```

```
cusip string,
sedol string,
createddate string,
modifieddate string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\t"
LOCATION
"/user/cloudera/EquityReports/ExternalTables/Block";
```

3. Reconciliation View Creation:
   Now base table data and incremental table data should be reconciled together to accumulate the latest updates with old data.

```
CREATE or Replace VIEW block_reconcile_view
AS
SELECT t1.* FROM
(SELECT * FROM block UNION ALL
SELECT * FROM block_increment) s1
JOIN
(SELECT id, max(modifieddate) max_modified
FROM
(SELECT * FROM block
UNION ALL
SELECT * FROM block_increment) s2
GROUP BY id) s
ON s1.id = s.id AND s1.modifieddate =
s.max_modified;
```

4. Creation of reporting table;
   Now reconciled data in step 4 is stored in reporting table.
   So, Reporting table contains most up to date data and is used to fetch reports from it.

```
CREATE TABLE block_reporting_table AS
SELECT * FROM block_reconcile_view;
```

After fetching all reports from reporting table, most up to date data from reporting table has to be moved to base table before starting next day incremental cycle.

5. Drop Base Table:
   Base table has to be dropped before moving reporting table to base table.

```
Drop Table Block;
```

6. Create base table from reporting table:
   I this step most up to date data from reporting table is moved to base table.

```
CREATE TABLE block AS
SELECT * FROM block_reporting_table;
```

Now base table will contain all data and next update cycle can be started.
In every next update cycle reporting table has to be dropped before executing 4th steps.

Scheduling Jobs using Oozie:
Oozie is a scheduler using which various repetitive jobs can be scheduled.

1. Oozie Workflows :
   All the steps given in above section can accumulated in each separate workflow.

2. Oozie Coordinators :
   All workflows created in next steps can be scheduled at a particulate time. As time comes that jobs will be started automatically.

3. Oozie Bundles :
   A job bundle can be formed from related coordinators which will be triggered depending upon the condition given.

Using same set of steps all the tables can be migrated from traditional database to the hive database which will be used for reporting purpose.

Applications
After migration of all required tables to hive system, developed system can be used for reporting as well as data warehouse system.
Example reports :

1. Reports to show all allocated blocks :
   Select * from block where AllocatedQuantity=Quantity;

2. Report to show all deleted blocks :
   Select * from block where isDeleted=1;

Conclusion:

- Efficient reporting system is developed using given steps in hive/HDFS.
- Traditional database reporting overhead can be decreased using this HDFS base reporting System.

Acknowledgement

I express great many thanks to Prof. Amrit Priyadarshi and Prof. Sachin S. Bere for their great effort of supervising and leading me, to accomplish this fine work. To college and department staff, they were a great source of support and encouragement. To my friends and family, for their warm, kind encourages and loves. To every person who gave me something too light along my pathway. I thanks for believing in me.

References

[1] Apache Hadoop. Available at http://wiki.apache.org/hadoop.
[2] Facebook Lexicon at http://www.facebook.com/lexicon.

[3] Hive wiki at http://www.apache.org/hadoop/hive.
[4] Hadoop Map-Reduce Tutorial at http://hadoop.apache.org/common/docs/current/mapred_tutorial.html.
[5] Hadoop HDFS User Guide at http://hadoop.apache.org/common/docs/current/hdfs_user_guide.html.
[6] Mysql list partitioning at http://dev.mysql.com/doc/refman/5.1/en/partitioning-list.html.
[7] Apache Thrift. Available at http://incubator.apache.org/thrift.
[8] DataNucleus .Available at http://www.datanucleus.org.
[9] A. Pavlo et. al. A Comparison of Approaches to Large-Scale Data Analysis. In Proc. of ACM SIGMOD, 2009.
[10] Hive Performance Benchmark. Available at http://issues.apache.org/jira/browse/HIVE-396
[11] TPC-H Benchmark. Available at http://www.tpc.org/tpch
[12] Running TPC-H queries on Hive. Available at http://issues.apache.org/jira/browse/HIVE-600
[13] K. Fraser, "Practical Lock Freedom", University of Cambridge,2003.
[14] Vinod, P. Suri P, "Maintaining a Binary Search Tree Dynamically. Proceedings of the 10th International Conference on Information Visualization", London, UK, 2006.
[15] J. Derryberry, D. D. Sleator, and C. C. Wang, "A lowerbound framework for binary search trees with rotations", Carnegie Mellon University, 2005.

Authors

Mr. Prashant R. Mahajan received his B.E. degree in I.T from University of Pune in 2011. He has 4.3 years of experience of working with MNCs in Pune. He is currently working toward the M.E. Degree in Computer Engineering from University of Pune. His research interests lies in Hadoop, Software Engineering and Business Process Management.

Mr. Amrit Priyadarshi received his B.E. degree in Electronics Engineering and MTech in Computer Science and Engineering. He has 10 years of experience as Assistant professor and he is currently perusing his Phd Degree.