A Study on improving Query Performance using Bucketing in Apache Hive

1. M.Suneetha ,
(Research Scholar in Rayalaseema University, Kurnool, A.P) &
Assoc.Professor, ISE Dept.,R R Institute of Technology,Bangalore. suni.mulumudi@gmail.com

2. Dr. V.S.K.Reddy,
Ph.D(IIT- KGP),FIETE,MIEEE,MISTE,MCSI
Professor in CSE Dept.&
Principal,MRCET,Secunderabad, Telangana
mrcet2004@gmail.com

Abstract

Data is the most important in today's lifestyle of the world. Data can be easily manageable as it stocked in database. With the help of Data Management System all the operations like retrieving and maintaining it becomes easy. But huge volume of data cannot be manageable so easily. So partitioning is the best solution for that. Partitioning is used to slice the data horizontally over the entire range or on a smaller range of values using one or more column. In this paper, brief review of methods of partitioning and helps to reduce wait in response time. Paper shows the positive result with partitioning methods. Hive is a good tool for performing queries on large datasets. Hive organizes tables into partitions. The partition statement lets Hive alter the way it manages the underlying structures of the table's data directory. The objective of partitioning is to reduce the time in extracting the required data using Hive.

Keywords: Partitioning, Bucketing, Apache Hive, Hadoop, HDFS, Bigdata.

1. Introduction

Partitioning in Hive is very useful to prune data during query to reduce query times. Partitions are created when data is inserted into table. There are two types of partitioning in Hive : 1. Static Partitioning 2. Dynamic partitioning . Usually when loading files (big files) into Hive tables static partitions are preferred. That saves time in loading data compared to dynamic partition. In Hive's implementation of partitioning, data within a table is split across multiple partitions. Each partition corresponds to a particular value(s) of partition column(s) and is stored as a sub directory within th table's directory on HDFS(Hadoop Distributed File System). This Apache software Foundation project is designed to provide a fault-tolerant file system designed to run on commodity hardware. The HDFS is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. Data in each partition may in turn be divided into Buckets based on the value of a hash function of some column of the Table.

2. Literature Survey

''Divide and conquer‟ rule is used to reduce the complexity which results in increasing the performance of the database. Near uniform range partition (NURP) approach is based on range partitioning. Traditionally uniform range partitioning algorithm were used for partitioning the tables but to speed up the partitioning of table, current partitioning techniques are studied and three range partitioning strategies are added. To manage the data in partitions, more strategies are involved so that these strategies will automate the partition. NURP-I is used to maintain the data equally in each partition. Uniform range partition distributes data unequally and Adjustment stage adjusts the data by splitting and merging. NURP-II uses a single query which helps to increase the execution speed. NRUP-III starts automating the partition when data is increasing gradually. NRUP is efficiently used for partitioning of large databases which leads into a good advantage of this method because this automates the partitioning of large database. Near-uniform range Partitioning Algorithm (NPA) is used as increased partitioning approach for huge amount of data in real-time data warehouse. The primary work includes new test of data warehouse and aimed for range partitioning using NPA. This paper also focused on the efficiency of the data warehouse. NPA also focuses on multilevel partition of table. It checks small tables to find the complexity such that it helps in increasing the performance. As amount of data increases, real time partition is done by new partitioning plan. Shin obi horizontal partitioning helps to improve the query performance. It most probably focuses on clustering of physical data and improves the performance by frequently accessed indexing data. This paper deals with the algorithms which optimally partition the table and manage the partition on different disk. This paper uses index partitioning approach for dynamic query workload from traffic monitoring application.

3.1 BIGDATA

Hive is a good tool for performing queries on large datasets, especially datasets that require full table scans. But quite often there are instances where users need to filter the data on specific column values. Generally, Hive users know about the domain of the data that they deal with. With this knowledge they can identify common columns that are frequently queried in order to identify columns with low cardinality which can be used to organize data using the partitioning feature of Hive. In non-partitioned tables, Hive would have to read all the files in a table's data directory and subsequently apply filters on it. This is slow and expensive especially in cases of large tables.

Due to the advent of new technologies, devices, and communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. If you pile up the data in the form of disks it may fill an entire football field. The same amount was created in every two days in 2011, and in every ten minutes in 2013. This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected.

90% of the world's data was generated in the last few years.

3.2. What is Bigdata?

Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

- **Black Box Data** : It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.

- **Social Media Data**: Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.

- **Stock Exchange Data**: The stock exchange data holds information about the ‘buy’ and ‘sell’ decisions made on a share of different companies made by the customers.
- **Power Grid Data**: The power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data**: Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data**: Search engines retrieve lots of data from different databases.

Figure 3.1: BIG DATA Applications

3.3 Big Data Challenges

The major challenges associated with big data are as follows:

- Capturing data
- Duration
- Storage
- Searching
- Sharing
- Transfer
- Analysis
- Presentation

To fulfill the above challenges, organizations normally take the help of enterprise servers.

3.3.1 Traditional Approach

In this approach, an enterprise will have a computer to store and process big data. Here data will be stored in an RDBMS like Oracle Database, MS SQL Server or DB2 and sophisticated softwares can be written to interact with the database, process the required data and present it to the users for analysis purpose.

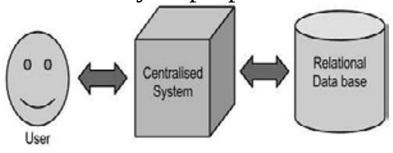


Figure 3.2: Traditional Approach

3.3.2 LIMITATION

This approach works well where we have less volume of data that can be accommodated by standard database servers, or up to the limit of the processor which is processing the data. But when it comes to dealing with huge amounts of data, it is really a tedious task to process such data through a traditional database server.

GOOGLE’S SOLUTION

Google solved this problem using an algorithm called MapReduce. This algorithm divides the task into small parts and assigns those parts to many computers connected over the network, and collects the results to form the final result dataset.

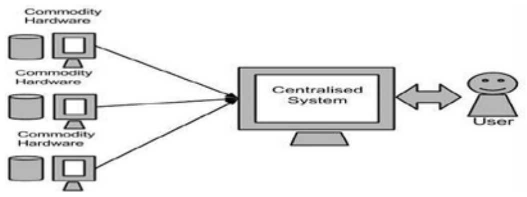


Figure 3.3: Servers with Higher Capacity

Above diagram shows various commodity hardware’s which could be single CPU machines or servers with higher capacity.

4. SYSTEM REQUIREMENT SPECIFICATION

IJSER

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it.

HARDWARE REQUIREMENTS

- 12-24 1-4TB hard disks in a JBOD (Just a Bunch Of Disks) configuration
- 2 quad-/hex-/octo-core CPUs, running at least 2-2.5GHz
- 64-512GB of RAM
- Bonded Gigabit Ethernet or 10Gigabit Ethernet (the more storage density, the higher the network throughput needed)

SOFTWARE REQUIREMENTS

- FRONT END : Jettyserver, Web UI in JSP
- BACK END : Apache Hadoop, Apache PIG
- OPERATINGSYSTEM: Linux-UBUNTU
- IDE : ECLIPSE

.

5. METHODOLOGY

Although many systems have been developed "by the seat of the pants," without using any formal methodology, historically methodologies have proven to be very beneficial. A primary benefit is thoroughness. Without a methodology, it is extremely likely that something will be overlooked, omitted, or improperly linked to the rest of the system.

5.1 Requirements

SDLC Methodology:This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

5.2 Design of HDFS

HDFS is a file system designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware. HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern. A dataset is typically generated or copied from source, and then various analyses are performed on that dataset over time. Each analysis will involve a large proportion, if not all, of the dataset, so the time to read the whole dataset is more important than the latency in reading the first record. Hadoop is written in Java, and all Hadoop filesystem interactions are mediated through the Java API. Hadoop interprocess communication between nodes in the system is implemented using remote procedure calls (RPCs). The RPC protocol uses serialization to render the message into a binary stream to be sent to the remote node, which then deserializes the binary stream into the original message. MapReduce is a programming model for data processing. The model is simple, yet not too simple to express useful programs in. Hadoop can run MapReduce programs written in various languages MapReduce programs are inherently parallel, thus putting very large-scale data analysis into the hands of anyone with enough machines at their disposal. MapReduce comes into its own for large datasets, so let’s start by looking at one.

5.3 **Map and Reduce**

MapReduce works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer. The programmer also specifies two functions: the map function and the reduce function.

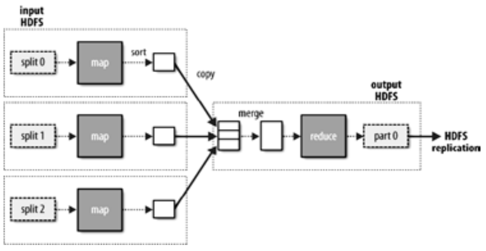


Figure 5.1: Mapping and Reducing

5.4 Commodity hardware

Hadoop doesn't require expensive, highly reliable hardware to run on. It's designed to run on clusters of commodity hardware (commonly available hardware available from multiple vendors3) for which the chance of node failure across the cluster is high, at least for large clusters. HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failure.

5.5 Working of Hive

The following diagram depicts the workflow between Hive and Hadoop.

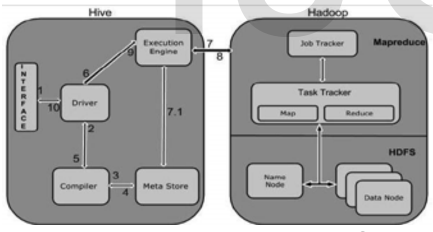


Figure 5.2: Working of Hive

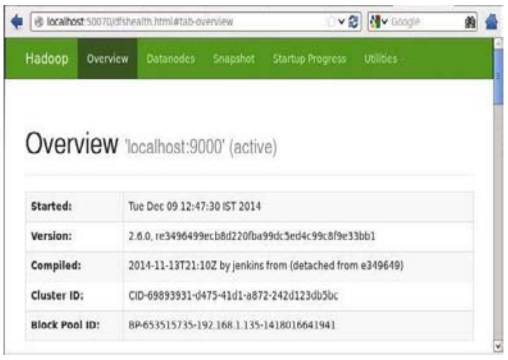
The following table defines how Hive interacts with Hadoop framework:

Step No.	Operation
1	**Execute Query** The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2	**Get Plan** The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3	**Get Metadata** The compiler sends metadata request to Metastore (any database).
4	**Send Metadata** Metastore sends metadata as a response to the compiler.
5	**Send Plan** The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6	**Execute Plan** The driver sends the execute plan to the execution engine.
7	**Execute Job** Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	**Metadata Ops** Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	**Fetch Result** The execution engine receives the results from Data nodes.
9	**Send Results** The execution engine sends those resultant values to the driver.
10	**Send Results** The driver sends the results to Hive Interfaces.

Table 5 .1 Hive interacts with Hadoop framework

6. IMPLEMENTATION

All Hadoop sub-projects such as Hive, Pig, and HBase support Linux operating system. Therefore, need to install any Linux flavored OS.



Overview 'localhost:9000' (active)

Started:	Tue Dec 09 12:47:30 IST 2014
Version:	2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-69893931-d475-41d1-a872-242d123db5bc
Block Pool ID:	BP-653515735-192.168.1.135-1418016641941

Figure 6.1: Accessing Hadoop on Browser

Verify all applications for cluster

The default port number to access all applications of cluster is 8088. Use the following url to visit this service.
http://localhost:8088/

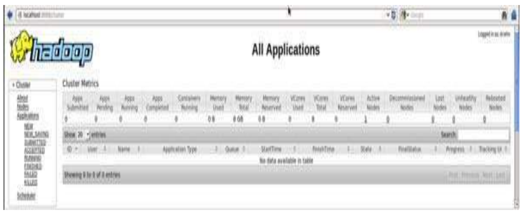


Figure 6.2: Verify all applications for cluster

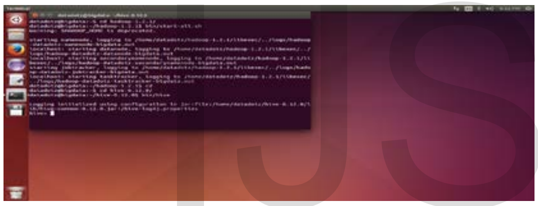
The next step is installing Hive.

7. SIMULATION RESULTS

7.1 VARIOUS SNAPSHOTS

SNAPSHOT 1

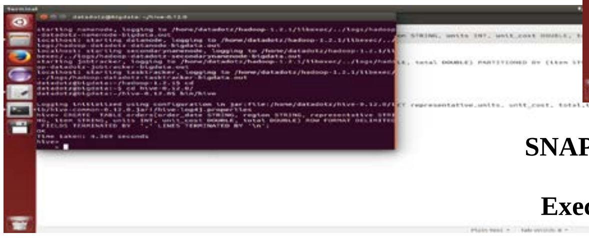
Initializing hive



```
[illegible]
hive>
```

SNAPSHOT 2

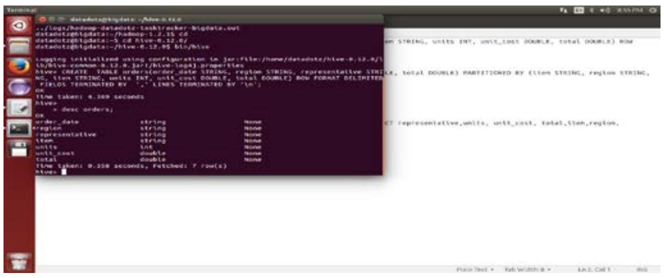
Creating a table in hive



```
[illegible]
hive> CREATE TABLE orders(order_date STRING, region STRING, representative STRI
NG, item STRING, units INT, unit_cost DOUBLE, total DOUBLE) ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';
OK
Time taken: 4.369 seconds
hive>
    >
```

SNAPSHOT 3

Describing the table

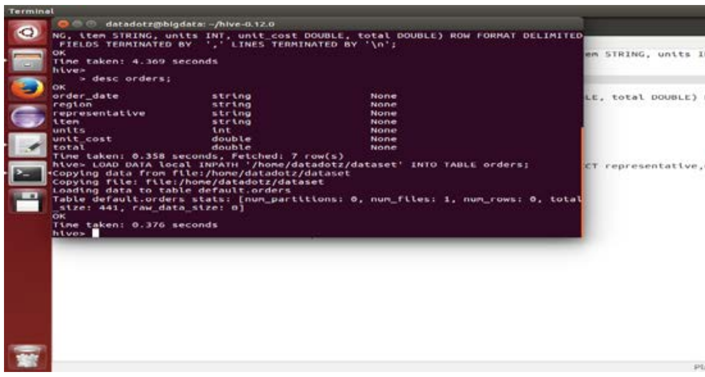


```
../logs/hadoop-datadotz-tasktracker-bigdata.out
datadotz@bigdata:~/hadoop-1.2.1$ cd
datadotz@bigdata:~$ cd hive-0.12.0/
datadotz@bigdata:~/hive-0.12.0$ bin/hive

Logging initialized using configuration in jar:file:/home/datadotz/hive-0.12.0/l
ib/hive-common-0.12.0.jar!/hive-log4j.properties
hive> CREATE TABLE orders(order_date STRING, region STRING, representative STRI
NG, item STRING, units INT, unit_cost DOUBLE, total DOUBLE) ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';
OK
Time taken: 4.369 seconds
hive>
    > desc orders;
OK
order_date              string                  None
region                  string                  None
representative          string                  None
item                    string                  None
units                   int                     None
unit_cost               double                  None
total                   double                  None
Time taken: 0.358 seconds, Fetched: 7 row(s)
hive>
```

SNAPSHOT 4

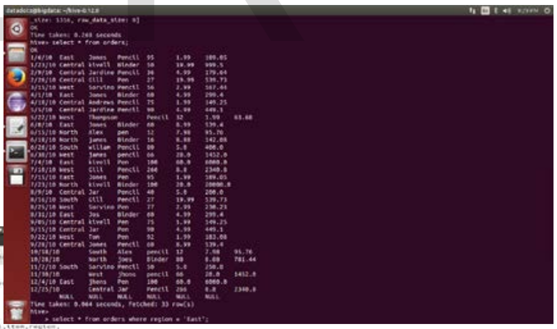
Loading the data into the table



```
NG, item STRING, units INT, unit_cost DOUBLE, total DOUBLE) ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';
OK
Time taken: 4.369 seconds
hive>
    > desc orders;
OK
order_date              string                  None
region                  string                  None
representative          string                  None
item                    string                  None
units                   int                     None
unit_cost               double                  None
total                   double                  None
Time taken: 0.358 seconds, Fetched: 7 row(s)
hive> LOAD DATA local INPATH '/home/datadotz/dataset' INTO TABLE orders;
Copying data from file:/home/datadotz/dataset
Copying file: file:/home/datadotz/dataset
Loading data to table default.orders
Table default.orders stats: [num_partitions: 0, num_files: 1, num_rows: 0, total
_size: 441, raw_data_size: 0]
OK
Time taken: 0.376 seconds
hive>
```

SNAPSHOT 5

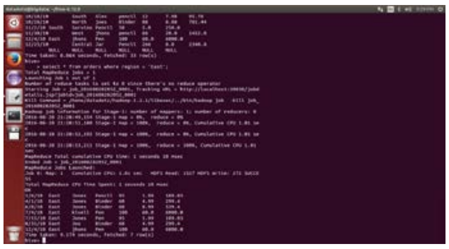
Checking the data in the table



```
_size: 1316, raw_data_size: 0]
OK
Time taken: 0.268 seconds
hive> select * from orders;
OK
1/6/10  East    Jones   Pencil  95      1.99    189.05
1/23/10 Central Kivell  Binder  50      19.99   999.5
2/9/10  Central Jardine Pencil  36      4.99    179.64
2/26/10 Central Gill    Pen     27      19.99   539.73
3/15/10 West    Sorvino Pencil  56      2.99    167.44
4/1/10  East    Jones   Binder  60      4.99    299.4
4/18/10 Central Andrews Pencil  75      1.99    149.25
5/5/10  Central Jardine Pencil  90      4.99    449.1
5/22/10 West    Thompson        Pencil  32      1.99    63.68
6/8/10  East    Jones   Binder  60      8.99    539.4
6/15/10 North   Alex    pen     12      7.98    95.76
6/18/10 North   james   Binder  16      8.88    142.08
6/26/10 South   william Pencil  80      5.0     400.0
6/30/10 West    James   pencil  66      20.0    1452.0
7/4/10  East    kivell  Pen     100     60.0    6000.0
7/10/10 West    Gill    Pencil  260     8.0     2340.0
7/15/10 East    Jones   Pen     95      1.99    189.05
7/23/10 North   kivell  Binder  100     20.0    20000.0
8/9/10  Central Jar     Pencil  40      5.0     200.0
8/16/10 South   Gill    Pencil  27      19.99   539.73
8/25/10 West    Sorvino Pen     77      2.99    230.23
8/31/10 East    Jos     Binder  60      4.99    299.4
9/05/10 Central kivell  Pen     75      1.99    149.25
9/15/10 Central Jar     Pen     90      4.99    449.1
9/22/10 West    Tom     Pen     92      1.99    183.08
9/28/10 Central Jones   Pencil  60      8.99    539.4
10/10/10        South   Alex    pencil  12      7.98    95.76
10/28/10        North   Joes    Binder  88      8.88    781.44
11/7/10 South   Sorvino Pencil  50      5.0     250.0
11/30/10        West    Jhons   pencil  66      20.0    1452.0
12/4/10 East    Jhons   Pen     100     60.0    6000.0
12/25/10        Central Jar     Pencil  260     8.0     2340.0
        NULL    NULL    NULL    NULL    NULL    NULL
Time taken: 0.064 seconds, Fetched: 33 row(s)
hive>
    > select * from orders where region = 'East';
```

SNAPSHOT 6

Executing a query and checking out the time taken to execute the given query

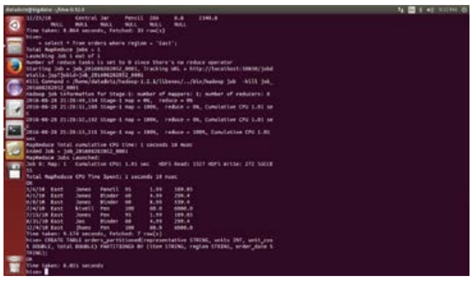


```
[illegible]
hive>
    > select * from orders where region = 'East';
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
[illegible]
MapReduce Total cumulative CPU time: 1 seconds [illegible]
[illegible]
MapReduce Jobs Launched:
[illegible]
Total MapReduce CPU Time Spent: 1 seconds [illegible]
OK
[illegible]
Time taken: [illegible] seconds, Fetched: 7 row(s)
hive>
```

SNAPSHOT 7

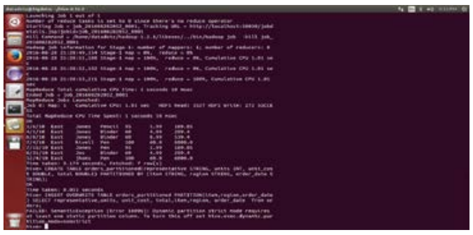
IJSER

Creating a partitioned table in hive



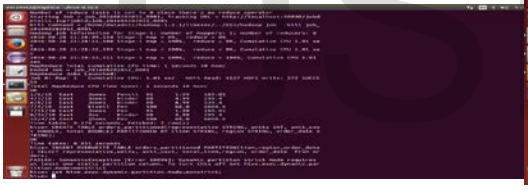
SNAPSHOT 8

Loading data into the partitioned table



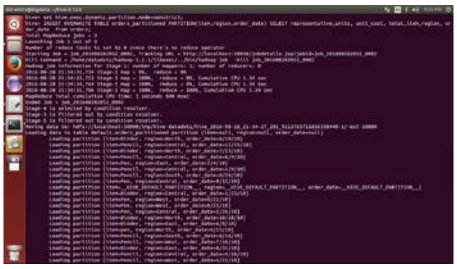
SNAPSHOT 9

Setting hive dynamic partition mode to non-strict

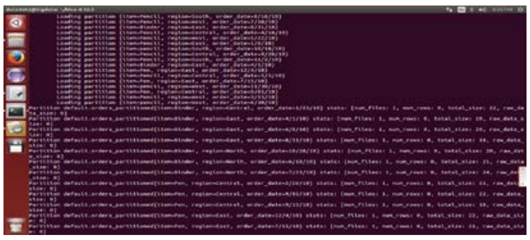


SNAPSHOT 10

Loading the data into the partitionedtable

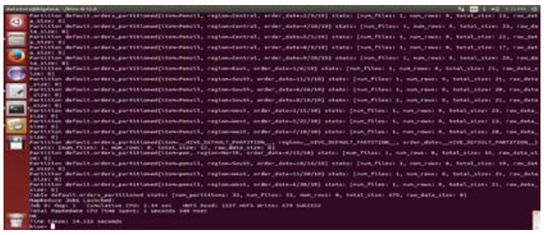


SNAPSHOT 11(Loading the data into the partitioned table)

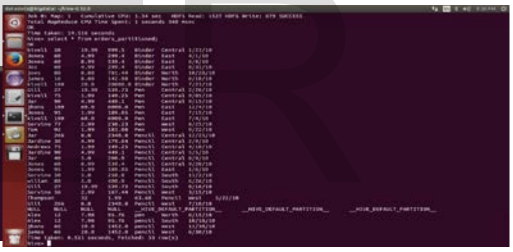


SNAPSHOT 12

Loading the data into the partitioned table

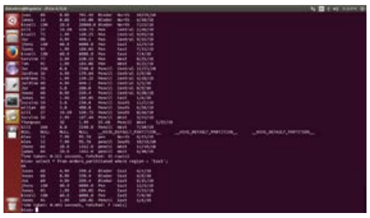


SNAPSHOT 13(Checking the data from the partitioned table)



SNAPSHOT 14

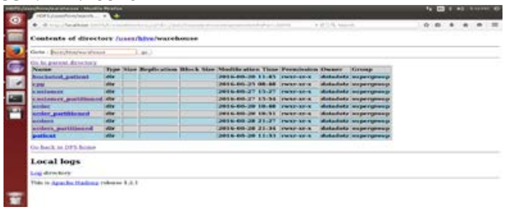
Passing the same query which we gave before partition and checking the time required to execute the same query after partition (See the time difference for the queries)



SNAPSHOT 15

In browser checking the tables

IJSER



Contents of directory /user/hive/warehouse

Goto : /user/hive/warehouse go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
bucketed_patient	dir				2016-08-20 11:45	rwxr-xr-x	datadotz	supergroup
[illegible]	dir				2016-08-25 08:48	rwxr-xr-x	datadotz	supergroup
customer	dir				2016-08-27 15:27	rwxr-xr-x	datadotz	supergroup
customer_partitioned	dir				2016-08-27 15:56	rwxr-xr-x	datadotz	supergroup
order	dir				2016-08-20 10:48	rwxr-xr-x	datadotz	supergroup
order_partitioned	dir				2016-08-20 16:51	rwxr-xr-x	datadotz	supergroup
orders	dir				2016-08-28 21:27	rwxr-xr-x	datadotz	supergroup
orders_partitioned	dir				2016-08-28 21:36	rwxr-xr-x	datadotz	supergroup
patient	dir				2016-08-20 11:33	rwxr-xr-x	datadotz	supergroup

Go back to DFS home

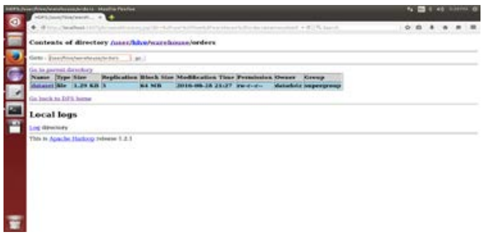
Local logs

Log directory

This is Apache Hadoop release 1.2.1

SNAPSHOT 16

Connecting to our data in the browser



Contents of directory /user/hive/warehouse/orders

Goto : /user/hive/warehouse/orders go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
dataset	file	1.29 KB	1	64 MB	2016-08-28 21:27	rw-r--r--	datadotz	supergroup

Go back to DFS home

Local logs

Log directory

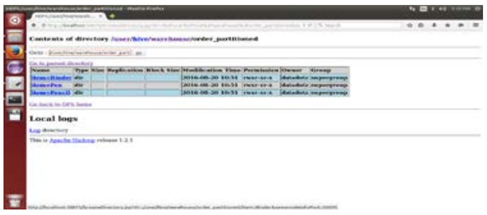
This is Apache Hadoop release 1.2.1

SNAPSHOT 17

Checking the loaded data in the browser

SNAPSHOT 18

Checking the Partitioned Table in the browser



Contents of directory /user/hive/warehouse/order_partitioned

Goto : /user/hive/warehouse/order_parti go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
item=Binder	dir				2016-08-20 16:51	rwxr-xr-x	datadotz	supergroup
item=Pen	dir				2016-08-20 16:51	rwxr-xr-x	datadotz	supergroup
item=Pencil	dir				2016-08-20 16:51	rwxr-xr-x	datadotz	supergroup

Go back to DFS home

Local logs

Log directory

This is Apache Hadoop release 1.2.1

Conclusion

In conclusion, using Hive partitioning in the right context and on appropriate columns will help a data management platform be much more efficient.

References

1. Bittencourt, L.F. and Madeira, E.R.M. "A Performance-Oriented Adaptive Scheduler for Dependent Tasks on Grids," Concurrency and Computation: Practice and Experience.

2. Caron, E. Chis, A. Desprez, F. And Su, A. "Design of Plug-in Schedulers for a GRIDRPC Environment," Future Generation Computer Systems, vol. 24, no. 1, pp. 46-57.

3. Dinda, P.A. And O'Hallaron, D.R. "Host Load Prediction Using Linear Models," Cluster Computing, vol. 3, no. 4, pp. 265-280.

4. Dinda, P.A. "Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems," IEEE Trans. Parallel and Distributed Systems, vol. 17, no. 2, b pp. 160-173.

5. Eddy Caron, Andreea Chis, Frederic Desprez, Alan Su (November 2011) **"**Plug-in Scheduler Design for a Distributed Grid Environment".

6. Liang Hu, Xi-Long Che, (2012)"Online System for Grid Resource Monitoring and Machine Learning-Based Prediction" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 23.
7. Massie, M.L. Chun, B.N. And Culler, D.E. "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," Parallel Computing, vol. 30, no. 7, pp. 817-840.

8. Peter Dinda, A. and David R. O'Halloran (July 2012) "AN Extensible Toolkit for Resource Prediction in Distributed Systems" School of Computer Science Carnegie Mellon University Pittsburgh, PA, 15213.

9. Sam Verboven, Peter Hellinckx, Frans Arickx and Jan Broeckhove (2011) "Runtime Prediction based Grid Scheduling of Parameter

IJSER

Sweep Jobs" University of Antwerp Antwerp, Belgium.

10. Sodan, A.C. Gupta, G. Han, L. Liu, L. and Lafreniere, B. "Time and Space Adaptation for Computational Grids with the ATOPGrid Middleware," Future Generation Computer Systems, vol. 24, no. 6, pp. 561-581.

11. Swany, D.M. and Wolski, R. "Multivariate Resource Performance Forecasting in the Network Weather Service," Proc. ACM/IEEE Conf. Supercomputing, pp. 1-10.

12. Vetter, J.S. and Reed, D.A. "Real-Time Performance Monitoring, Adaptive Control, and Interactive Steering of Computational Grids," Int'l J. High Performance Computing Applications, vol. 14. no. 4, pp. 357-366.

13. Waheed et al., "An Infrastructure for Monitoring and Management in Computational Grids," Proc. Fifth Int'l Workshop Languages, Compilers and Run-Time Systems for Scalable Computers, vol. 1915, pp. 235-245.

14. Wolf, F. and Mohr, B. "Hardware-Counter Based Automatic Performance Analysis of Parallel Programs," Proc. Conf. Parallel Computing (ParCo '03), pp. 753-760.

Authors

1). M. SUNEETHA, working as Assoc.Professor in ISE Department, R R Institute of Technology, Bangalore-90 , having 19 years of teaching experience and pursuing Ph.D(CSE) in Rayalaseema University, Kurnool, Under the guidance of Dr.V.S.K Reddy, Principal, MRCET, Secunderabad.

2). Dr. V.S.K. Reddy, Principal, Malla Reddy College of Engineering & Technology has an experience of more than 22 years in Teaching and Industry put together. He is alumni of IIT Kharagpur, he obtained Ph.D in the area of Multi-media Signal Processing and Communication Protocols. He is versatile in multidisciplinary specializations in Electronics & Communications and Computer Science Engineering. His laurels include more than 45 Publications in the National and International reputed Conferences and Journals. He is fellow of IETE, Life Member of ISTE and Member of IEEE. He was awarded as "Best Teacher" in three consecutive Academic years with citation and cash award. He is the recipient of "India Jewel Award" for outstanding contribution in the research in the field of Engineering and Technology.

IJSER