Performance Analysis of Big Data with Data models using Artificial Intelligence

L.Umarani[1], Research Scholar, Bharathiar University, umarcl@gmail.com
Dr. A. John Sanjeev Kumar[2], Assistant Professor, Thiagarajar College of Engineering, ajscse@tce.edu

Abstract - Proliferation of new information sources such as medical images, financial data, sales data, radio frequency identification and web tracking data, there is a challenge to decipher trends and gain sense of data that is orders of magnitude larger than ever earlier. One of the technologies most often associated with the era of big data is Hadoop. Although in that respect is much expert information about Hadoop, there is not much info around how to effectively structure data in a Hadoop environment. Though the nature of parallel processing and the MapReduce system provide an optimal environment for processing big data quickly, the structure of the big data itself plays a vital role. This paper explores doable used for data modeling in a Hadoop environment. Specifically, the purpose of the experiments described in this paper was to figure out the best structure and physical modeling techniques for storing data in a Hadoop cluster using Hive to enable efficient data access. Although other software interacts with Hadoop, the experiments focused on Hive. The Hive infrastructure is most felicitous for traditional data warehousing-type applications. The experiment does not focus on HBase. This paper explores a data partition strategy and investigates the role indexing, data types, file types, and other data architecture decisions play in designing data structures in Hive. To test the different data structures, it focused on typical queries utilized for analyzing web traffic data. These test included most referring sites, web analyses such as counts of visitors, and other typical business questions used by weblog data. The primary measure for culling the optimal structure of data in the Hive is predicated on the performance of web analysis queries. For comparison purposes, it was quantified the performance in Hive and the performance in an RDBMS. The reason for this comparison is to more preponderant understand how the techniques that we are habituated with utilizing in an RDBMS work in the Hive environment. It explored techniques such as storing data as a compressed sequence file in Hive that are particular to the Hive architecture. Through these experiments, it endeavored to show that how data is structured (in effect, data modeling) is just as consequential in an immensely colossal data environment as it is in the traditional database world.

Index Terms **: Big Data Analysis, Data Modeling, Database, Hadoop, Hive**

I.Introduction

Similar to massively parallel processing (MPP) databases, the puissance of Hadoop is in the parallel access to data that can reside on a single node or on multiple nodes. In general, MapReduce provides the mechanism that enables access to each of the nodes in the cluster. Inside the Hadoop framework, Hive provides the ability to engender and query data on an astronomically immense scale with a familiar SQL-predicated language called HiveQL. It is a vital point to note that in these experiments, Hive within the Hadoop environment is stringently used. For the experiment, a typical data warehouse-type workload is simulated where data is loaded in batch, and the queries are executed to answer non- operational strategic business questions.

In general, all data stored in HDFS are broken into blocks of data. The same blocks of data were replicated across multiple nodes to provide reliability if a node failed, and additionally to increment the performance during MapReduce jobs. By default Hadoop environment replicates each block of data thrice. The NameNode in the Hadoop cluster accommodates as the metadata repository that describes where blocks of data are located in each file stored in HDFS.

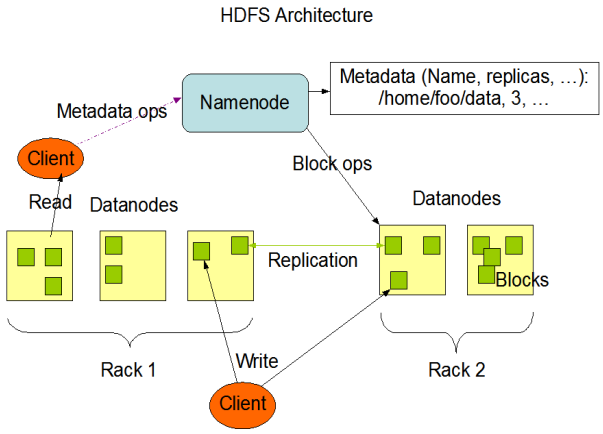


Fig 1: Hadoop Distributed File System Architecture

At a higher caliber, when a table is engendered through Hive, a directory is engendered in HDFS on each node that represents the table. Files that contain the data for the table are engendered on each of the nodes, and the Hive metadata keeps track of where the files that make up each table are located. All these files are located in a directory with the name of the table in HDFS in the /user01 folder by default. For example, in these tests, a table named EXPLORER_DIM is created. We can use an HDFS command to see the new table located in the /user01/explorer_dim directory. By using the command *hadoop fs -ls*, the contents of the browser_dim directory are listed. In this directory, we find a file named

exp_dim.csv. HDFS commands are similar to standard Linux commands.

```
$ hadoop fs –ls /user01/explorer_dim
Found 1 items
-rw-r—r—1 mc1 supergroup 44957179 2014-10-01 12:07 /user01/explorer_dim/exp_dim.csv
```

The principal takeaway is to understand at a high caliber how data is stored in HDFS and managed in the Hive environment. The physical data modeling experiments that are performed ultimately affect how the data is stacked in blocks in HDFS and in the nodes where the data is located and how the data is accessed. This is especially true for the tests in which data is partitioned using the Partition verbal expression to redistribute the data predicated on the buckets or ranges defined in the partitions.

The Hadoop Clusters

The Hadoop cluster consist of two areas. 1. HDFS- stores the data 2. MapReduce - processes all of the computations on the data.

The Job Tracker is responsible for controlling the parallel processing of the MapReduce functionality. The Task Trackers act as instructed by the Job Tracker to process the MapReduce jobs.

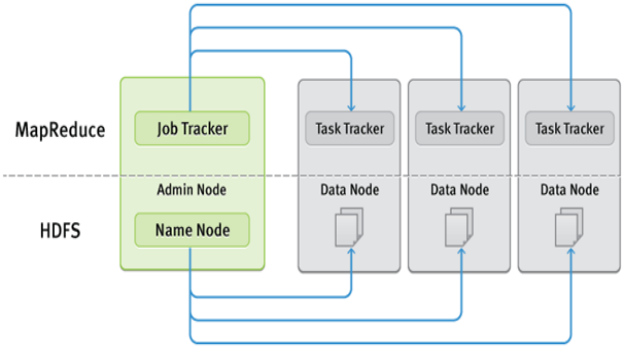


Fig 2 : MapReduce architecture.

Data Input and Load

The data for the experiments was generated to resemble a company's technical support website. The company sells its products worldwide and uses Unicode to support foreign character sets. There were 25 million original weblog sessions featuring 90 million clicks created, and then duplicated it 90 times by adding unique session identifiers to each row. This bulked-up flat file was loaded into the RDBMS and Hadoop. For tests, both a flat file representation of the data and a typical star schema design of the same data were generated.

II.Background of the analysis

To test the various data modeling techniques, following queries were used to simulate the typical types of questions business users might ask of clickstream data.

1. Most visited top-level directories on the customer support site for a given week
2. The most visited pages that are referred from a Google search for any given month
3. The most common search terms used on the customer support site for a given year
4. Total number of visitors per page using the given browser
5. Number of visitors spend more than 15 seconds viewing each page for a given week and year

As part of the criteria, the SQL statements were used to determine the optimal structure for storing the clickstream data in Hadoop and in an RDBMS. Techniques in Hive are investigated to improve the performance of the queries. The purpose of these experiments was to investigate how traditional data modeling techniques apply to the Hadoop and Hive environment. RDBMS is incorporated only to measure the effect of tuning techniques within the Hadoop and Hive environment and to see how comparable techniques work in an RDBMS.

It is important to note that there was no intent to compare the performance of the RDBMS to the Hadoop and Hive environment, and the results were for particular hardware and software environment only. To determine the optimal design for data architecture, following criteria considered:

- There would be no unnecessary duplication of data.
- The data structures would be progressively tuned to get the best overall performance for the average of most of the queries, not just for a single query.

The experiments began devoid of indexes, partitions, or statistics in both schemas and in both environments. The intent of the first experiment was to determine whether a star schema or flat table performed better in Hive or in the RDBMS for the considered queries. During succeeding rounds of testing, parameters like compression, indexing and partitions are used to tune the data

III. Findings

A.Parameter 1: Flat File versus Star Schema

The intent of this first experiment was to determine whether the star schema or flat table structure

performed better in each environment in a series of use cases. The first experiment did not have any tuning applied such as indexing.

As you can see, both the Hive table and the RDBMS table in the star schema structure performed significantly faster compared to the flat file. These results for Hive were surprising, the more efficient practice in HDFS of storing data in a denormalized structure to optimize I/O.

A. Hadoop Vs RDBMS Flat File Processing

Table 1 : Flat File execution in Hadoop

Query	Min (H:MM:SS)	Max (H:MM:SS)	Average (H:MM:SS)
1	0:42:00	0:42:13	0:42:07
2	0:39:55	0:40:46	0:40:21
3	0:49:53	0:51:36	0:50:45
4	0:40:37	0:42:37	0:41:37
5	0:39:43	0:40:25	0:40:04

Table 2 :Flat File execution in RDBMS

Query	Min (H:MM:SS)	Max (H:MM:SS)	Average (H:MM:SS)
1	0:54:25	0:57:49	0:56:07
2	0:59:26	0:59:52	0:59:39
3	0:58:14	0:58:53	0:58:34
4	0:56:22	0:57:44	0:57:03
5	0:54:20	0:54:57	0:54:39

B. Hadoop Vs RDBMS Star Schema Processing

Table 3 : Star Schema execution in hadoop

Query	Min (H:MM:SS)	Max (H:MM:SS)	Average (H:MM:SS)
1	0:07:40	0:09:22	0:08:31
2	0:02:08	0:07:57	0:05:02
3	0:45:53	0:50:37	0:48:15
4	0:11:04	0:13:14	0:12:09
5	0:07:57	0:08:32	0:08:14

Table 4 : Star Schema execution in RDBMS

Query	Min (H:MM:SS)	Max (H:MM:SS)	Average (H:MM:SS)
1	0:29:03	0:29:41	0:29:22
2	0:29:19	0:29:35	0:29:27
3	0:29:28	0:30:27	0:29:58
4	0:28:58	0:29:09	0:29:03
5	0:29:00	0:29:56	0:29:28

Table 5 : Performance comparison of flat file and start schema in Hadoop

Query	Flat File Average (H:MM:SS)	Star Schema Average (H:MM:SS)	Performance Difference (Flat to Star)
1	0:42:07	0:08:31	0:33:36
2	0:40:21	0:05:02	0:35:18
3	0:50:45	0:48:15	0:02:30
4	0:41:37	0:12:09	0:29:28
5	0:40:04	0:08:14	0:31:49

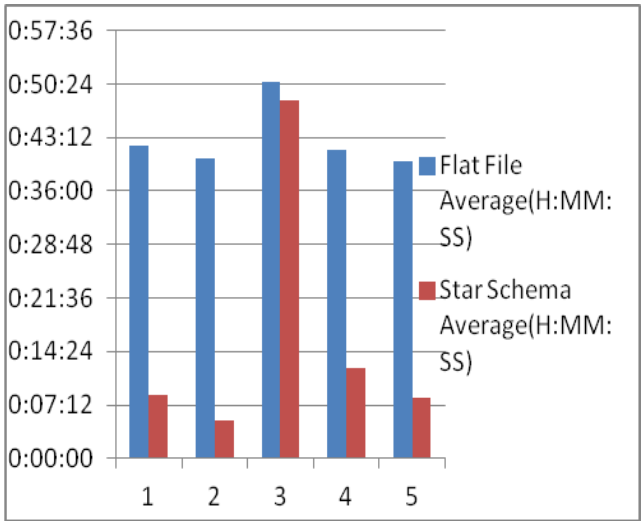


Fig.3 : Performance comparison of flat file and start schema in Hadoop

C.RDBMS Schema Difference

Table 6: Performance comparison of Flat file and Star schema in RDBMS

Query	Flat File Average (H:MM:SS)	Star Schema Average (H:MM:SS)	Difference (Star to Flat)
1	0:56:07	0:29:22	0:26:45
2	0:59:39	0:29:27	0:30:12
3	0:58:34	0:29:58	0:28:36
4	0:57:03	0:29:03	0:27:59

5	0:54:39	0:29:28	0:25:11

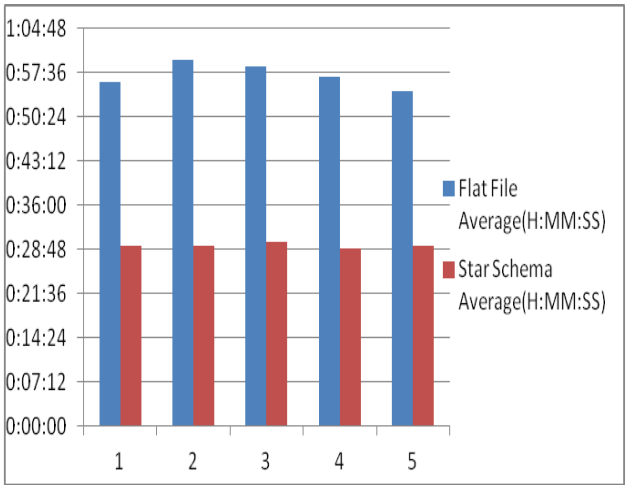


Fig 4 : Performance comparison of Flat file and Star schema in RDBMS

Although the star schema was faster in the Hadoop text file environment, it was decided to complete the remaining experiments for Hadoop using the flat file structure because it is the more efficient data structure for Hadoop and Hive. As we understand through references, the most important reason to avoid normalization is to reduce disk seeks, such as those typically required to steer foreign key relations. Denormalizing data permit it to be scanned from or written to large, adjacent sections of disk drives, which optimize I/O performance. However, we face the consequence of denormalization, data duplication and the greater risk of inconsistent data.

B.Parameter 2: Compressed Sequence Files

The second experiment applied only to the Hive environment. In this experiment, the data in HDFS were converted from uncompressed text files to compressed sequence files to determine whether the type of file for the table in HDFS made a difference in query performance.

The results of this experiment clearly show that the compressed sequence file was a much better file format for the taken queries than the uncompressed text file.

Table 7 : Sequence File execution in hadoop

Query	Min (H:MM:SS)	Max (H:MM:SS)	Average (H:MM:SS)
1	0:05:44	0:05:48	0:05:46
2	0:06:27	0:05:41	0:06:04
3	0:06:51	0:07:04	0:06:57
4	0:06:35	0:06:47	0:06:41
5	0:06:30	0:06:40	0:06:35

Table 8 : Performance comparison of text file and sequence file in Hadoop

Query	Text File Average (H:MM:SS)	Sequence File Average (H:MM:SS)	Difference (Text to Sequence)
1	0:42:07	0:05:46	0:36:21
2	0:40:21	0:06:04	0:34:16
3	0:50:45	0:06:57	0:43:47
4	0:41:37	0:06:41	0:34:56
5	0:40:04	0:06:35	0:33:29

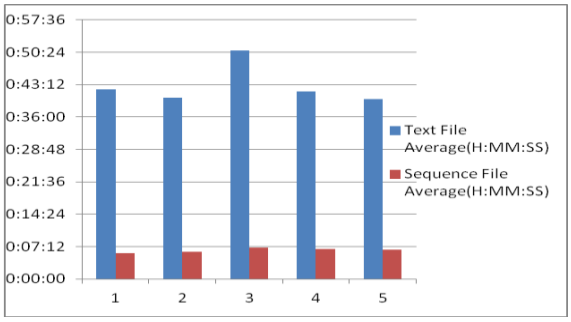


Fig 5 : Performance comparison of text file and sequence file in hadoop

C.Parameter 3: Indexes

In this experiment, indexes were applied to the appropriate columns in the Hive flat table and to the RDBMS fact table. Statistics were gathered for the fourth set of tests. In Hive, a B-tree index was added to each of the columns used in these queries. In the RDBMS, a bitmap index was included to each foreign key in referencing table, and a B-tree index was added to each of the other columns used in the queries that were not already indexed.

With the prominent exception of the third query in the Hadoop environment, adding indexes provided a significant increase in performance across all of the queries.

Table 9 : Flat File execution in hadoop

Query	Min (H:MM:SS)	Max (H:MM:SS)	Average (H:MM:SS)
1	0:01:17	0:01:28	0:01:23

2	0:01:25	0:01:33	0:01:29
3	0:05:55	0:06:03	0:05:59
4	0:01:32	0:01:37	0:01:34
5	0:04:42	0:04:45	0:04:44

Table 10: Star Schema execution in RDBMS

Query	Min (H:MM:SS)	Max (H:MM:SS)	Average (H:MM:SS)
1	0:00:04	0:00:04	0:00:04
2	0:00:25	0:01:01	0:00:43
3	0:00:25	0:00:43	0:00:34
4	0:00:07	0:00:07	0:00:07
5	0:00:25	0:00:31	0:00:28

Table 11: Performance comparison with respect to Indexed and Non-Indexed data in hadoop

Query	No Indexes Average (H:MM:SS)	Indexed Average (H:MM:SS)	Difference (No Indexes to Indexed)
1	0:05:46	0:01:23	0:04:24
2	0:06:04	0:01:29	0:04:35
3	0:06:57	0:05:59	(00:02)
4	0:06:41	0:01:34	0:05:06
5	0:06:35	0:04:44	0:01:52

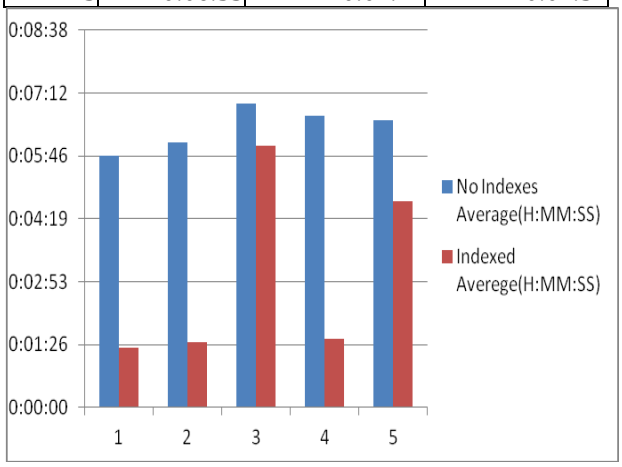


Fig 6 : Performance comparison with respect to Indexed and Non-Indexed data in hadoop

Table 12 : Performance comparison with respect to Indexed and Non-Indexed data in RDBMS Schema

Query	Non Indexed Average (H:MM:SS)	Indexed Average (H:MM:SS)	Difference (Non Indexes to Indexed)
1	0:29:03	0:00:04	0:28:59
2	0:29:19	0:00:43	0:28:36
3	0:29:28	0:00:34	0:28:54
4	0:28:58	0:00:07	0:28:51
5	0:29:00	0:00:28	0:28:32

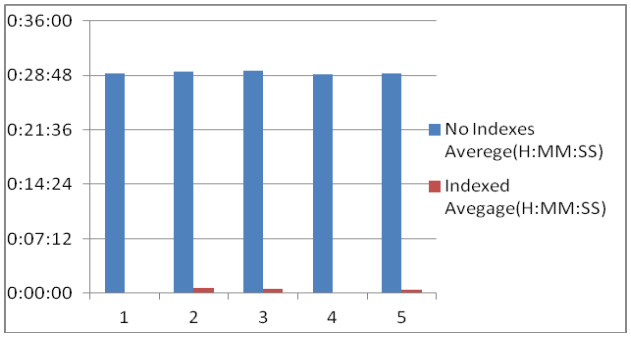


Fig 7 : Performance comparison with respect to Indexed and Non-Indexed data in RDBMS Schema

D. Parameter 4: Partitioning

In experiment 4, every date value is partitioned in both the flat table in Hive and in the fact table in the star schema in the RDBMS.

Partitioning significantly improved all queries except for the third query, which was slightly slower in Hive and significantly slower in the RDBMS.

Table 13 : Flat File execution in details hadoop

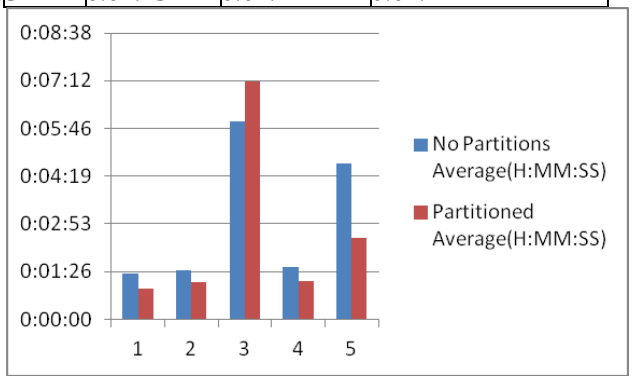
Query	Min.Time (H:MM:SS)	Max.Time (H:MM:SS)	Average (H:MM:SS)
1	0:00:50	0:01:00	0:00:55
2	0:01:04	0:01:10	0:01:07
3	0:06:42	0:07:41	0:07:11
4	0:01:07	0:01:13	0:01:10
5	0:02:25	0:02:28	0:02:27

Table 14 : Star Schema execution in RDBMS

Query	Min.Time (H:MM:SS)	Max.Time (H:MM:SS)	Average (H:MM:SS)
1	0:00:01	0:00:03	0:00:02
2	0:00:02	0:00:06	0:00:04
3	0:39:33	0:45:40	0:42:37
4	0:00:02	0:00:46	0:00:24
5	0:00:01	0:00:03	0:00:02

Table 15 : Performance comparison with respect to partitioned and Non-partitioned data in hadoop

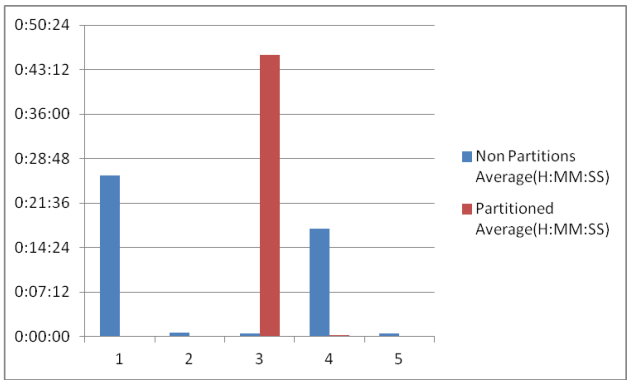
Query	Non Partition Average (H:MM:SS)	Partitioned Average (H:MM:SS)	Difference (Non Partitioned to Partitioned)
1	0:01:22	0:00:55	0:00:27
2	0:01:29	0:01:07	0:00:22
3	0:05:59	0:07:11	(0:01:05)
4	0:01:34	0:01:10	0:00:24
5	0:04:43	0:02:27	0:02:17



Performance comparison with respect to partitioned and Non-partitioned data in hadoop

Table 16 : Performance comparison with respect to partitioned and Non-partitioned data in RDBMS

Query	Non Partition Average (H:MM:SS)	Partition Average (H:MM:SS)	Difference (Non Partition to Partition)
1	0:26:01	0:00:02	0:25:59
2	0:00:39	0:00:04	0:00:35
3	0:00:31	0:45:32	(0:45:01)
4	0:17:28	0:00:17	0:17:11
5	0:00:27	0:00:02	0:00:25



Performance comparison with respect to partitioned and Non-partitioned data in RDBMS

IV.Interpretation of Results

The results of the first parameter were surprising. At the start of tests, it was fully expected that the flat file structure would perform better than the star schema structure in the Hadoop and Hive environment. In the next table, information that helps explain the conflicts in the amounts of time processing the queries is offered. For instance, the measure of storage required is significantly higher in the flat table structure for the query. Moreover, the number of mappers and reducers needed to run the query was significantly higher in the flat table structure. Altering TaskTracker heap sizes, a system setting showed benefits in the denormalized table structure. Nevertheless, the end of the experiment was to run with the default system settings in Cloudera Hadoop and investigate the effects of morphological changes in the data.

Table 17 : Unique Visitors per Page

	Denormalized	Normalized	DIFF
Virtual Memory (GB)	7,317	2,816	4,501
Heap (GB)	3,777	1,401	2,376
Read (GB)	372	218	154
Table Size (GB)	867	217	650
Execution Plan	3483 maps/999 reduce	1123 maps/352 reduce	
Time (minutes)	38	13	25

Our second parameter showed the performance increase that emerged from transitioning from text files to sequence files in Hive. This performance improvement was expected. However, the degree of the improvement was not. Compared to uncompress text file, on compressed sequential file the queries ran about ten times faster. The compressed sequence file optimizes disk space usage and I/O bandwidth performance by using binary encoding and split-table compression. This proved to be the single biggest factor with regard to data structures in Hive. For this experiment, block compression is used.

In third parameter, indexes are added to the fact table in the RDBMS and to the flat table in Hive. As expected, the indexes generally improved the performance of the queries, except third query, where adding the indexes did not show any improvement in Hive. Explain Plan helps explain why this is happening. In Explain Plan it was observed that there were no indexes used in the predicate of the query. Given the distinctiveness of the

data, this makes sense because almost all of the values of DOMAIN_NM were the support site itself. The referring domain was primarily www.google.com.

In Parameter Both the fact table and the flat table were partitioned by date column. Partitioning tables change how Hive structures the data storage. Added to the directory for each table, Hive creates subdirectories reflecting the partitioning structure. When a query is executed on Hive, it does not need to scan the entire directory. Rather, partition elimination enables the query to go directly to the subdirectory or subdirectories where that data is located to retrieve the results. Because many of the queries used search column in the WHERE clause of the query, execution time enhanced. The same improvement was seen in the RDBMS, which was able to use partition elimination for most of the queries. In the third query, the predicate does not include the column. In this case, having partitions actually hurt query performance because the query needed to examine each partition individually to locate the relevant rows. The decline in performance was noteworthy in the RDBMS.

V.Conclusions

Through these experiments, it was evident that structuring data properly in Hive were as important as in an RDBMS. The decision to store data in a sequence file format alone accounted for a performance improvement of more than 1,000%. The wise use of indexes and partitions resulted in important performance gains by cutting down the quantity of information processed.

For information architects working in the Hive setting, the good information is that many of the same techniques such as indexing that are applied in a traditional RDBMS environment are applicable.

The key takeaway is that it is necessitate interpreting data and the underlying technology in Hadoop to effectively tune the data structures. Merely creating a flat table or star schema does not result in optimized structures. It is important to understand how data is distributed, and accordingly a data structure has to be created that work well for the access patterns of the environment. Being capable to decipher MapReduce job logs as well as run explains plans are key skills to effectively model data in Hive.

And as known that tuning for some queries might have an adverse impact on other queries as observed with partitioning.

Further experimentation should look into the performance enhancements offered with another Hive file format, such as RC File, which organizes information by column rather than by row. Some other data modeling test could examine how well collection data types in Hive work compared to traditional data types for storing information. As big data technology continues to make headway, the features that are available for structuring data will continue to improve, and further options for improving data structures will become usable.

VI.References

[1] Zhen Chen, Fuye Han, Junwei Cao, Xin Jiang, and Shuo Chen, "Cloud Computing-Based Forensic Analysis for Collaborative Network Security Management System," TSINGHUA SCIENCE AND TECHNOLOGY ISSNl l1007-0214l l05/12l l pp40-50 Volume 18, Number 1, February 2013

[2] Yaxiong Zhao , Jie Wu, and Cong Liu, "Dache: A Data Aware Caching for Big-Data Applications Using the MapReduce Framework" TSINGHUA SCIENCE AND TECHNOLOGY ISSN l l1007-0214l l05/10l l pp39-50 Volume 19, Number 1, February 2014

[3] T. Sun, C. Shu, F. Li, H. Yu, L. Ma, and Y. Fang, "An ef_cient hierarchical clustering method for large datasets with map-reduce," in Proc. Int. Conf. Parallel and Distributed Comput., Appl. Technol., 2009, pp. 494_499.

[4] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. "Bigtable: A distributed storage system for structured data," in Proc. of OSDI'2006, Berkeley, CA, USA, 2006.

[5] D. Battr'e, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke, Nephele/pacts: "A programming model and execution framework for web-scale analytical processing," in Proc. of SOCC'2010, New York, NY, USA, 2010

[6] H. Gonzalez, A. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, andW. Shen, Google fusion tables: "Data management, integration and collaboration in the cloud," in Proc. of SOCC'2010, New York, NY, USA, 2010

[7] "Configuring the Hive Metastore." Cloudera, Inc. Available at http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/4.2.0/CDH4-Installation-Guide/cdh4ig_topic_18_4.html. Accessed on June 18, 2013.

[8] Kindy, Diallo Abdoulaye and Pathan, Al-Sakib Khan (2011) *A Survey on SQL injection: vulnerabilities, attacks, and prevention techniques.* In: The 15th IEEE Symposium on Consumer Electronics (IEEE ISCE2011), June 14-17, 2011, Singapore.

[9] http://www.devshed.com/c/a/MySQL/Database-Design-Using-KeyValue-Tables/

[10] http://antirez.com/post/Sorting-in-key-value-data-model.html

[11] RajKumar Buyya, Chee Shin Yeo, Srikumar Vanugopal, James Broberg, Ivona Brandic, "Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility", Future Generation Computer System, 10.1016/j.future.2008.12.001 Pages 599-616 volume 25, Issues 6, June 2009

[12] Manuel P.D, Thamarai Selvi.S, "Trust management system for grid and cloud resources", Advanced Computing, 2009. ICAC 2009. First International Conference, ISBN 978-1-4244-4786-2 Pages 176-181 Dec 2009.