International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 4, April - 2014

Efficient Query Processing and Data Integrity in Cloud using Column Oriented Database

Shwetha C H
Department of Computer Science
And engineering
VTU Belgaum, KVGCE Sullia, DK

Balapradeep K N
Asst Professor
Department of Computer Science
And engineering
KVGCE Sullia, DK

Dr. Antony P J
Director, PG Studies
KVGCE Sullia, DK
VTU Belgaum

***Abstract*—Column oriented database system re-examines how and when data is compressed. Column oriented database has become quite popular among relational database system because of its performance benefits. Column oriented database stores its data as a adjacent records thus it has the advantage in the area where combination of many separated units of queries requires large amount of data. They are more efficient for read-only query which requires accessing only the memory.**

This paper provides a detail description of compressing large sized data and storing in cloud using different compression and query execution techniques. In this paper, we compare the performance of row store based on different configurations with column stores and shows that row oriented database performance is quite slower. This paper then analyzes the performance difference and shows that there is a significant difference in storage level. This paper will discuss about efficiently extracting data stored in cloud for improving the performance of query execution. Finally this paper conveys the performance benefits and utilization of column oriented database in accessing and storing large database in cloud.

Keywords— column oriented database, data compression, run-length encoding technique, null suppression.

I. INTRODUCTION

The database systems are simply a collection of records. It often represented as large quantities of information which are organized typically in form of tables. The traditional database system is a two dimensional world as shown in fig. 1. All values of row are stored in single entity in row oriented database where as in column-oriented database system data from each column are store together, which means particular attribute can be accessed without having to read other attribute of the row.

In a relational, row oriented database data values are collected and managed as individual rows and events containing related rows. They are not efficient at performing operations that apply to the entire data set. Large database consist of bulk of data set which may stored in cloud. Query processing requests higher performance in cloud environment, but traditional row oriented database cannot achieve high query speed when processing large volume data. Designers need to balance between optimizing query performance and maximizing query flexibility.

Businesses applications need faster data accessing to support rapid business decisions and better system utilization [1]. Along with that, they need a simple, self managing system that increases performance but helps to reduce administrative complexities and expenses. Large database is so complex that it becomes difficult to process in cloud by using traditional relational data processing applications.

Database usually includes data sets with sizes beyond the ability of commonly used software tools to capture, storage, search, and process the data within a minimum elapsed time. The trend to larger data sets is due to the additional information obtained from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data [2].

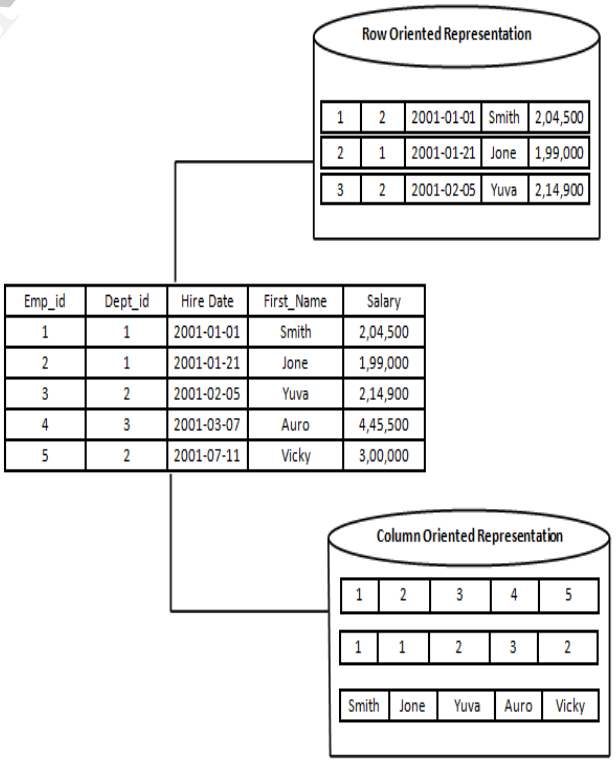

Fig. 1 Column Oriented Database

A Column Oriented DBMS provides unlimited scalability, high availability and self managing administration. A more efficient design is used in column-oriented systems, which eliminates the storage space used in row-based systems. Some column-oriented systems not only store data by column, but store the data values comprising a column within the index itself; efficient methods are selected to optimize storage and query efficiency for each individual column's data type [3].

Column-oriented organizations are more efficient when aggregate needs to be computed over many rows but only for a notably smaller subset of all columns of data, because reading that smaller subset of data can reduce the execution time than reading all data. Column-oriented organizations are more efficient when new values of a column are supplied for all rows at once, because that column data can be written efficiently and replace old column data without touching any other columns for the rows [4].

Column-oriented compression schemes and query processing techniques improve CPU performance by allowing database operators to operate directly on compressed data. It also keeps the memory consumption low.

II. BACKGROUND AND RELATED WORK

This section briefly presents the related works to characterize column oriented database performance relative to row oriented database system.

Today, there is a growing demand for high-quality performance in query processing and data integrity. Almost all of the research community has moved towards column oriented database technique, which has overcome the lacks of the scope of traditional row oriented databases system by introducing different compression and query execution techniques. The main developments in Column oriented database systems are as follows.

Column stores have been implemented from the early days of DBMS development. TAXIR was the first application of a column-oriented database storage system with focus on information-retrieval in biology. For many years, only the Sybase IQ product was commercially available in the column-oriented DBMS class. However, that has changed rapidly in the last few years with many open source and commercial implementations.

In a recent set of articles, Don Haderle and Michael Stonebraker review a bit of database history and point the way to column-oriented databases. Haderle and Stonebraker discuss the constraints of the original relational database implementations, and how changes in the cost of processing can usher in column-oriented databases more suitable to analyze rich data types.

Haderle and Stonebraker describe that Current relational database management systems are largely built on designs from the 1980s. Back then, computers were expensive and slow relative to today's systems. The minimization of expensive CPU cycles was the driving force in early relational DBMS design. The market sweet spot was transaction processing coupled with simple decision support, which was generally satisfied by access on a limited set of attributes (dimensions) [1].

Samuel R. Madeen published paper on Integrating Compression and Execution in Column-Oriented Database Systems. They discussed how it extended C-Store [1] with a compression sub-system and presented some compression schemes not traditionally used in row oriented DBMSs can be applied to column-oriented systems. Then they evaluated a set of compression schemes and showed that the best scheme depends not only on the properties of the data but also on the nature of the query workload.

Daniel J. Abadi presented a paper on Column Stores vs. Row Stores: How Different Are They Really? [5]. He compared the performance of a commercial row-store under a variety of different configurations with a column-store and show that the row-store performance is significantly slower on a recently proposed data warehouse benchmark and then it analyze the performance difference and show that there are some important differences between the two systems at the query executor level.

Naresh Kumar and Dr. Kapil Kr. Bansal presented a paper on Different Compression Techniques and Their Execution in Database Systems to Improve Performance [6]. They described that significant database performance gains can be had by implementing light-weight compression schemes and operators that work directly on compressed data. By classifying compression schemes according to a set of basic properties, it was able to extend C-Store to perform this direct operation without requiring new operator code for each compression scheme.

III. COLUMN ORIENTED DATABASE EXECUTION

Column oriented database system is aimed at the data integrity and fast query processing with different compression techniques and query execution techniques. Business environment uses complex queries of large size data. Processing these complex queries may include null values and repeated values.

A. Techniques

Different compression and query execution techniques are used to improve the performance of query execution process and data integrity of large sized data.

1) *Compression techniques*: Compression is a known technique used by many database management systems to increase performance [7]. Storing data in columns allows all the data to store together which may also be of large size but compressing the data within a column will reduce the size of database and optimizes the query execution time. Compression improves query performance by reducing the size of data on disk, decreasing seek times, increasing the data transfer rate and increasing buffer pool hit rate. Compression schemes not traditionally used in row-oriented DBMSs can be applied to column-oriented systems.

Techniques like Null Suppression and RunLength Encoding are used for compression [9].

a) *Null Suppression*: Null suppression is the generic term used for the technique that suppresses either zero or blank. The fundamental idea is that consecutive zeros or blanks in the data are deleted and replaced with a description of how many there were and where they existed. Generally,

this technique performs well on data sets where zeros or blanks appear frequently.

Here sequence of zeros and blanks are represented by a special character and by a number that indicates the length of sequence. Consider an example

Original Data:

ESSFnnnnnYtAtt4600000i77

Compressed Data:

ESSF#5YtAtt46%5i77

The compressed data has less size compared to original data.

b) *RunLength Encoding:* Run length encoding (RLE) is useful for data with large runs of repeated values. Thus, it is well-suited for columns that are sorted or that have reasonable-sized runs of the same value. Consider the following sequence of values:

- {aa,aa,aa,bc,ddd,ddd,ddd,ddd,ddd,ddd.. . . }

By counting the number of repeated values, we can code such a sequence as {value / numRepetitions} pairs. The sequence above could be represented as:

- {{aa / 3}, {bc/ 1}, {ddd/ 6}}

The data structure {value / numRepetitions} is known as an RLEDouble.

By storing the starting position in each RLE Double we can improve performance. The sequence above would now be Coded as:

{{aa /1/ 3}, {bc /4/ 1}, {ddd / 5 / 6}}.

The data structure{value / startPosition/ numRepetitions} is known as an RLE Triple.

2) *Query Execution Technique:* Business application requires maintaining and accessing millions of records at a time which leads to large execution time [8]. The paper focuses on reducing execution time by considers the problem of integrating compression and execution so that the query executer is capable of focusing on compressed data. This leads to gain in performance by improving I/O and CPU.

Next it analyzes the problem of tuple construction. Tuple construction is required when operators need to access multiple attributes from the same tuple [9].

Due to reduction in the file size, execution time of complex queries is also decreased. This has a huge benefit in business and finical sectors.

IV. EXPERIMENTAL WORK

This section compares the row oriented approaches to the performance of column oriented database on the complex database.

In row oriented database uses scan methods to access the records. Scan fetches millions of records which are normally used in analytical business. But when retrieving a particular attribute efficiency of scan will reduce. Since fetching of data need to scan each of the row in the memory or disk which decrease the performance by increasing the execution time.

Consider an Architectural design shown in the Fig. 2.

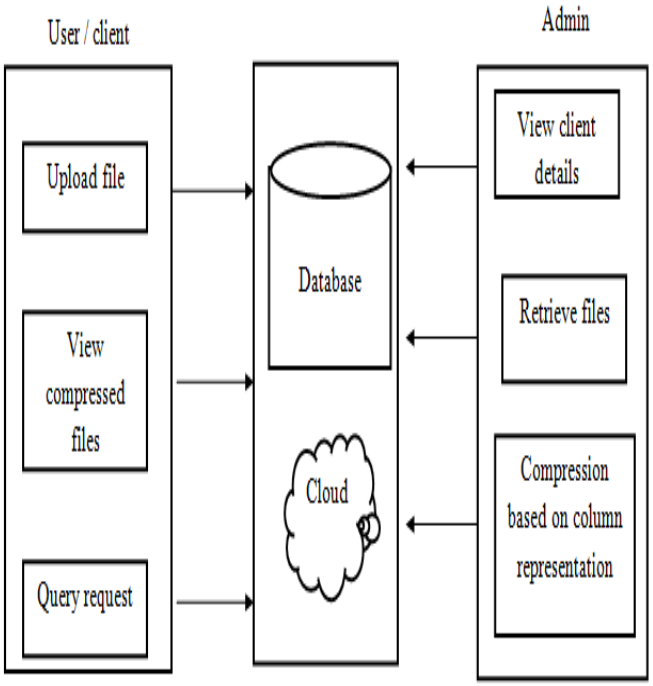

Fig. 2 Architectural design

Files of different size are provided as input from different user. These files may contain old database script or a million of records. They require evaluating the queries much more quickly. This can be done by storing each column separately and compressing the data. Run length encoding, null suppression and complex joins techniques are used to reduce the size of database and stores at cloud. Column oriented representation provides high speed storage and retrieval of large amount of data in a cloud.

User can access compressed file in a cloud and can request for query processing. User can compare the size of compressed and uncompressed files by viewing the file details.

V. PERFORMANCE ANALYSIS

The Column oriented databases are best suited for read-only queries where only parts of the data are accessed. This has improved the bandwidth utilization in cloud for complex query execution. By using different compression techniques, performance and data integrity in the queries execution has been increased. Column oriented database reduces the scan time which was inefficient in traditional row oriented database systems. Fig. 3 shows the average query execution time for a huge data. Considering 1TB of data is cloud. These data has been compressed and records are integrated such that its query execution time has been reduced significantly.

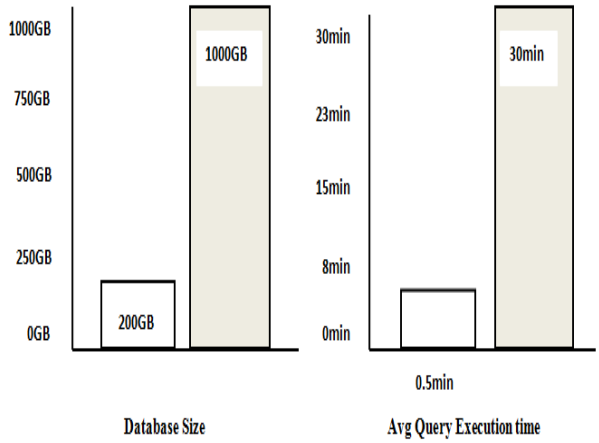

Fig. 3 Average Execution time based on different database size

VI. CONCLUSION

The Columnar Database is evolving software which can overcome the lacks of the scope of row oriented databases. This paper has focused on data integrity and fast query processing by compressing data size in cloud. The average execution time shown in Fig. 3 summarizes the result of data integrity and compression technique used in column oriented database. Business transactions, queries including unrestricted aggregation and time based sequences can be accessed within couple of seconds. In summary, this paper shows that significant database performance gains can be had by implementing compression on large database and operators that work directly on compressed data

REFERENCES

[1] en.wikipedia.org/wiki/Column-oriented_DBMS.
[2] Daniel J. Abadi, Peter A. Boncz, Stavros Harizopoulos, Column-oriented Database Systems, VLDB '09, August 24-28, 2009, Lyon, France.
[3] Miguel C. Ferreira, Samuel R. Madden, Compression and Query Execution within Column Oriented Databases.
[4] Daniel J. Abadi, Query Execution in Column-Oriented Database Systems,
[5] D. J. Abadi, S. R. Madden, N. Hachem, Column-stores vs. row-stores: how different are they really?, in: SIGMOD'08, 2008, pp.967–980.
[6] Tejaswini Apte1, Dr. Maya Ingale2, Dr. A.K. Goyal Sinhgad Institute of Business Administration & Research, Kondhwa(Bk.),Performance Improvement Tecniques in column oriented Database.
[7] Naresh Kumar, Dr. Kapil Kr. Bansal Assistant Professor Department of Information Technology , SRM University, NCR Campus,Different Compression Techniques and Their Execution In Database Systems To Improve Performance.
[8] Punam Bajaj Simranjit Kaur Dhindsa Department of Computer Science & Engineering, Chandigarh Engg. College, Mohali, India, Vision towards Column-Oriented Databases.
[9] Priyanka Raichand and Rinkle Rani Aggarwal Department of Computer Science and Engineering Thapar University, Patiala ,A Short Survey Of Data Compression Techniques For Column Oriented Databases.