Suresh Gyan Vihar University Journal of Engineering & Technology
(An International Bi-Annual Journal)
Vol . 1, Issue 2, 2015,pp.12-16
ISSN: 2395-0196

Review of Query Processing Techniques of Cloud Databases

Ruchi Nanda
Assistant Professor, IIS University Jaipur.

Abstract: The big challenge at the present time is to manage big distributed data like cloud. Traditional relational database management systems (RDBMS) are a choice but they are not well-suited to scale across large clusters of distributed servers. Hence alternatives to RDBMS have been developed. The development of new database management systems (DBMS) for the cloud computing environment or adaptability of the existing systems to the cloud computing environment is a critical component of cloud computing research.

This paper focuses on the study of the types of existing DBMS in the cloud computing domain. It reviews the various alternatives to RDBMS. It also focuses on the study of the parameters responsible for the performance of cloud database queries and reviews the work carried out so far, for the enhancement of the cloud database queries. Studies conducted in the paper helps in identifying the areas where research is exclusively needed.

Keywords: Databases, Cloud Databases, SQL stores, NoSQL data stores, cloud query performance, indexing.

I. INTRODUCTION

Cloud is a network of servers that pools different resources and cloud computing is a computing where the customers can access those servers using a web browser regardless of their device and location. A cloud provides access to the resources when there is demand and automatically deprivations them when there is no demand.

A. Database Management System

The relational model was first introduced by Edgar Codd in 1970. It uses a collection of tables to represent data items and their relationships. It is well-suited for Online Transaction Processing Systems. These applications are based on the ACID properties i.e. Atomicity, Consistency, Isolation, and Durability (ACID). These traditional DBMS are also known as row-oriented DBMS, as they store the data row-by-row. They keep all the information about an entity together and are preferred when queries access the data regarding an entity. These row-oriented DBMS includes Oracle, DB2, SQLServer, Teradata.

B. Cloud Databases

The data is distributed across several machines in network, so efficient management of data is a big worry for organizations using services of cloud. Here, the most important requirement of database management system is of scalability. Though RDBMS are robust and a vast majority of current database systems are based on the relational model yet they are not well-suited to scaling across large clusters of servers as they are not designed to be distributed. So it is difficult to ensure consistency, referential integrity and query performance in a distributed relational environment. Structured Query Language (SQL) is the dominant language of RDBMS. But SQL databases don’t scale [25].

Hence alternatives to relational database management systems have been developed. Some of them are:

1. Column-oriented DBMS is a database management system that stores the data by column. It is faster to read and allows a more efficient compression of the data. Hence query processing time is decreased. These are well-suited for online analytical processing systems (OLAP) and data warehouses. Some of the database management systems which are already in the market are Vertica (a commercial version of C-store), SADAS as well as open-source DBMS like LucidDB and MonetDB.

2. Key-value Stores (non-relational DBMS or NoSQL datastores) are used for storing large scale data & provide easy access. Its data models are schema-free, join-less and horizontally scaled. Here, domain is just like a table, but there is not a predefined schema. It is like a bucket where we can put the items. Items are identified by keys and a given key can have dynamic set of attributes attached to it. Data is created, updated, deleted and retrieved using API method calls. Some of its projects include Dynamo[11] used by Amazon.com, Google’s Bigtable [7] used in the Google’s application, Cassandra used by

Facebook for inbox search and project Voldemort used by LinkedIn.

3. Mapreduce[7] is a distributed framework that allows us to process large amount of data in parallel approach. Programmers create different map and reduce functions on the basis of user queries. The data files are stored in distributed file system (DFS). This approach is being used in Google's web search service, for the generation of data stored in Bigtable.

II. PARAMETERS DETERMINING PERFORMANCE OF DATABASE MANAGEMENT SYSTEMS

The objective of performance enhancement is to minimize the response time for each query and to maximize the throughput of the database server. The performance of the database management system can be determined by the following factors:

- System level issues: These issues can perform serious performance degradation. It occurs if [26]:
  - Utilization of CPU is high
  - Loads of I/O changes frequently
  - Hardware and software is not configured properly
  - Operating System of virtual machines and hypervisor [25].
  -

  Generally these issues are automatically managed by the cloud DBMS. So it reduces the need for extensive manual testing.

- Database design: The database design is one of the most important decision which has to be made carefully as the performance of the queries depends on:
  - File organization techniques
  - Constraints on the attributes
  - Normalization and De-normalization of relations.
  - Indexing

- Query Processing and Optimization techniques: Query processing and optimization are the main components of the database management system. The function of query processor [1] is to transform the query written in high-level language into a correct and efficient execution plan expressed in low-level language. As there are many ways to execute the same query, the aim of query optimization is to choose an efficient execution plan for processing a query. It chooses the one that minimizes the resources.. It takes information from the system catalog. The optimizer are required to consider factors such as the order in which to join the tables, the number of rows for each join when calculating an optimal access path, the algorithm to be used for performing the joins.

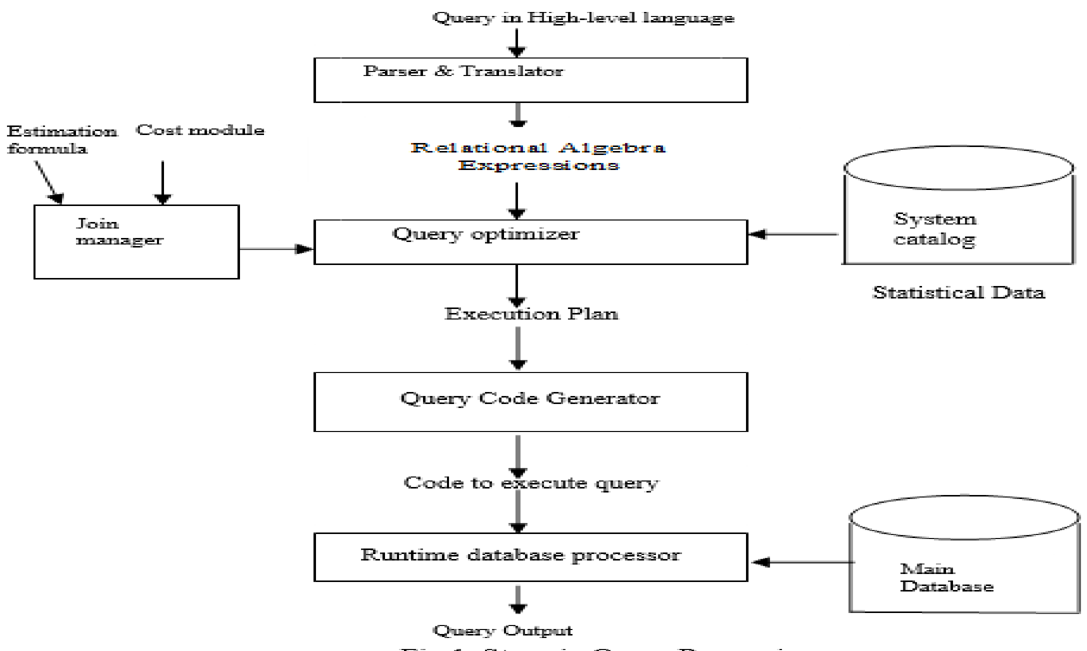

Fig 1. Steps in Query Processing

In a distributed environment like cloud, data is distributed to a number of sites, stored in its entirety on all sites or spilt on many sites. Here the query is processed and optimized in a different way. There are various issues which needs to be considered like, which copy of the data is to be used i.e. site selection, amount of data that needs to be transmitted from its location to the execution site, relative processing speed at each site and transmitting the final result to the site where the query is issued. The cost of the plan depends on these issues.

III. LITERATURE REVIEW

The section focuses on the techniques used to enhance the performance of query processor in cloud databases. This section highlights the current work on the specific techniques of indexing, processing and optimization of queries.
Query efficiency is achieved by employing a pure key-value data model where, both key and value are arbitrary byte strings (e.g. Dynamo), or its variant (as in Bigtable), where key is an arbitrary byte string and value is a structured record consisting of a number of named columns. But these solutions lack support for secondary indexes, range queries and multidimensional queries [24]. The current solution is Mapreduce [10], which is a programming model and a framework for processing large sets of raw data. A map-reduce program consists of two functions: Map and Reduce. The Map function processes the input data by distributing them to worker nodes for parallel computation and produces a set of intermediate results as key-value pairs, while the reduce function aggregates all the intermediate results with the same key from each node to produce the result. It can be used for structured data analysis of large sets. The limitations of Mapreduce as given by [23] are:

- It produces the necessary secondary indices in an offline batch manner. Hence, secondary indexes are not up-to-date. So newly inserted rows cannot be queried until they are indexed.
- It does not provide data schema support, declarative query language and cost-based query optimizations.

Cloud Global Index (CG-Index) [23], a secondary B^+-tree based indexing scheme for cloud storage systems was proposed. It provides high scalability, high throughput, high availability and high concurrency. It provides range search and dictionary operations. Index distribution technique for desired scalability was used. Experiments on Amazon’s EC2 were carried on and the results demonstrated that it handles a mixed workload of queries and updates efficiently, but the main limitation of CG-index was that it supports one-dimensional queries only.

Performance of the query processing can be increased if operations are directly applied on the compressed data [2]. C-store [22] is a column oriented store which was extended by keeping this view. Algorithms especially suited for column-oriented systems compared with algorithms commonly used by traditional DBMS. The comparisons were performed by changing the parameters like query workload and size of the data set. The results showed that the performance benefits of operating directly on compressed data in column oriented schemes is much greater than the benefits in operating directly on row-oriented schemes. They created decision-tree to aid the database designer to decide how to compress a particular column. The limitation with the optimizer was that it is not aware of the decompression costs of the various compression algorithms.

Various greedy and approximation algorithms have been proposed for optimization of queries. But they do not scale well for realistic workloads [17]. Two greedy algorithms were developed later which emphasizes on finding the most beneficial view in each step instead of finding most promising query.

A number of dynamic programming algorithms have been used but they are not able scale. So a new class of query optimization algorithms was developed known as Iterative dynamic programming. [16].

The performance of the database can be increased if common subexpressions from multiple queries can be evaluated only once and that can be reused in case the same subexpression again comes. Inter-application multi-query optimizer [18] was presented that re-uses previously computed (intermediate) results and eliminates redundant work. It was experimentally proved that the inter-application multi-query optimizer improves the query evaluation performance significantly. The limitation of inter-application optimizer is that the optimizer is limited to identify and re-use equivalent intermediate results, only. It is also beneficial to re-use the smallest superset of a requested intermediate result in case the equivalent result is not available, provided that using the superset is cheaper.

The studies show that there is enough scope in the performance improvement of queries as there are certain limitations with the works reported in the past and the same have a scope for further improvement.

IV. FUTURE WORK & CONCLUSION

With the increasing volume of data across the large number of applications, the challenge is to distribute the computations, responding to the query along with the distribution of data. Relational database management systems have grown overly complex, difficult to manage, and are struggling today to take full advantage of cloud computing technology [6]. So, there is need to reanalyze the design and processing of relational database technologies and refine the existing methods or develop new approaches exclusively for the cloud environment. These can be achieved by focusing on the following issues:

- How to reduce the execution time of various operations using indexing?
- How the desired information can be retrieved immediately from the database?
- How to get the results back from the database in time as well as in cost effective manner?

For this, there is need to analyze thoroughly the operational and architectural characteristics of cloud databases. There is also a need to study the techniques of processing and optimization of queries in the cloud databases, so that existing techniques can be refined.

REFERENCES

[1] Abadi, D. J. (2009), Data Management in the Cloud: Limitations and Opportunities**.** *IEEE **Data** Engineering Bulletin*, **Volume 32**, Number 1, March 2009, pp. 3-12

[2] Abadi D. J., S.R. Madden and M.C. Ferreira (2006), Integrating Compression and Execution in Column-Oriented Database Systems. *Proceedings of the 2006 ACM SIGMOD International conference on Management of data*, ISBN:1-59593-434-0 pp. 671 – 682

[3] Abadi D. J., S.R. Madden and N. Hachem (2008), Column-stores vs. row-stores: how different are they really? *Proceedings of ACM SIGMOD International conference on Management of data*, ISBN:978-1-60558-102-6, pp. 967-980

[4] Abadi, D. J. (2008), Query Execution in Column- Oriented Database Systems. Massachusetts Institute of Technology, U.S.A.

[5] Anh (2009), Query Processing and Optimization. http://cnx.org/content/m28213/latest/

[6] Bobrowski, S. (2011), The Future of Databases is in the Clouds. http://wiki.database.com/page/The_Future_of_Databases_is_in_the_Clouds

[7] Chang, F., J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber (2006), Bigtable: A Distributed Storage System for Structured Data. *Proceedings of the 7th symposium on Operating systems design and implementation*, ISBN:1-931971-47-1, pp. 205 – 218.

[8] Cloud Computing vhttp://www.n-axis.in/practices-cloudcomputing.php

[9] Connolly T. and C. Begg (1998), Database Systems A Practical Approach to Design, Implementation and Management, Pearson Education Ltd., New Delhi, India.

[10] Dean, J. and S. Ghemawat (2004) Mapreduce: Simplified data processing on large clusters. *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation ACM* pp. 137–150

[11] DeCandia, G., D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels(2007) Dynamo: Amazon's highly available key-value store, *In SOSP*, pp. 205–220.

[12] Garret (2010), What is a Column Oriented Database? http://www.columnorienteddatabase.com/

[13] Gounaris, A. (2009), A Vision for Next Generation Query Processors and an Associated Research Agenda. *2nd International Conference on Data Management in Grid and Peer-to-Peer Systems*, pp. 1-11.

[14] Jain, V. (1997), Information Technology, BPB Publications, New Delhi.

[15] Hurwitz, J., R. Bloor, M. Kaufman and F. Halpur (2010), Cloud Computing for Dummies, Wiley Publishing Inc., Indianapolis, Indiana.

[16] Kossmann, D. and K. Stocker (2000)**,** Iterative dynamic programming: a new class of query optimization algorithms. *ACM Transactions on Database Systems (TODS)* **Volume 25, Issue 1**.

[17] Kalnis, P. and D. Papadias (2003) Multi-query optimization for on-line analytical processing· *Information Systems*, **Volume 28, Issue 5,** pp. 457-473

[18] Manegold, S., A.J. Pellenkoft and M.L.Kersten, (2000), A Multi-Query Optimizer for Monet 2000. *Springer-Verlag, 2000 (repository id: 11170).*

[19] Marks, E.A. and B. Lozano (2010), Executive Guide to Cloud Computing, John Wiley & Sons, Inc., Hoboken, New Jersey.

[20] Oracle® Enterprise Manager Cloud Control Advanced Installation and Configuration Guide**, 12c Release 2 (12.1.0.2)**

[21] Ramakrishnan, R. and J. Gehrke (2000), Database Management Systems, McGraw-Hill International Editions, Singapore.

[22] Stonebraker, M., D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. R. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik (2005) C-Store: A Column-Oriented DBMS. *In VLDB* Trondheim, Norway, pp. 553–564.

[23] Wu, S., D. Beng and Kun (2010), **Efficient B-tree based indexing for cloud data processing.**, **Volume 3, Issue 1-2**.

[24] Wu, S. and K.L. Wu (2009), An indexing framework for efficient retrieval on the cloud. *IEEE Data Engineering Bulletin*, 32(1):pp. 77–84.

[25] Wiggins, A. (2009), SQL Databases Don't Scale. http://adam.heroku.com/past/2009/7/6/sql_databases_dont_scale/

[26] Fujitsu Technology Solutions GmbH (2010) Performance Report on Hyper-V *globalsp.ts.fujitsu.com/dmsp/Publications/public/wp-PR-Hyper-V-en.pdf*