QUERY PERFORMANCE OPTIMIZATION IN DATABASES FOR BIG DATA

Manoj Muniswamaiah[1], Dr. Tilak Agerwala[2] and Dr. Charles Tappert[3]

[1]Seidenberg School of CSIS, Pace University, White Plains, New York

ABSTRACT

Organizations maintain different databases to store and process big data which is huge in volume and have different data models. Querying and analysing big data for insight is critical for business. The data warehouses built should be able to meet the ever growing demand of data. With new requirements it is important to have near real times response from the big data gathered. All the data cannot be fit in to one particular database "One Size Does Not Fit All" since data originating from sources have different formats. The main focus of our research is to find an adequate solution using optimized data created by data engineers to improve the performance of query execution in a big data ecosystem.

KEYWORDS

Databases, Big data, Optimization, Analytical Query, Data Analysts and Data Scientists.

1. INTRODUCTION

Big data enables organizations to analyze data which is immense in volume, variety and velocity for decision making and take appropriate actions. There are different databases with varied data models to store and query big data: Columnar databases are used for read heavy analytical queries; online transactional processing databases are used for quicker writes and consistency; NoSQL data stores are used to process large volume of data and for horizontal scaling; HTAP databases are hybrid of both OLTP and columnar databases [1].

There are various data stores been designed and developed for specific needs and for optimal performances. Relational databases can store and process structured data efficiently but its performance decreases with read heavy queries. Similarly columnar databases are used for analytical query processing and NoSQL data stores are specialized to handle unstructured data. The processed data is stored in different databases to be used by analysts [1].

Performance optimization and different data models are important for data intensive applications. The main challenge is to build data pipelines that are scalable and efficient. Data engineers optimize and maintain these data pipelines which are crucial for the performance of the applications. Data pipelines process, transform and load the data in to the data warehouse which are used by data analysts and data scientists for research. Big data platforms which use different databases continue to be challenging with regard to improvement of query performance and also added complexity due to different data models used in these databases.

Data engineers primary job is to partition, normalize, index the base tables in order to improve the performance of the queries which are used for dashboards and reports. These optimized copies would be stored in different databases based on their data models for querying and quicker response. In this research we want to automate this process where if data analysts or data

Natarajan Meghanathan et al. (Eds) : CCSEA, CLOUD, SIPRO, AIFU, SEA, DKMP, NCOM - 2019
pp. 85-90, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90908

scientists issues a query against any desired database the query framework should be able to detect the optimized copies created and stored by data engineers and execute the query against optimized copy rather than the base table which would improve the performance of the query response.

2. BACKGROUND

There are several techniques to improve query performance and response time like partitioning the base table, indexing on required columns, materialization and creating OLAP cubes. Indexing helps in faster reads and retrieval of data. It is similar to dictionary data lookup, B-tree indexing keeps the data sorted and allows for sequential access of the data [2]. Consistency and freshness of the optimized copies is maintained by updating them whenever the base table changes.

BigDAWG is a polystore framework implemented to support multiple databases. The main feature of BigDAWG is to provide location independence which routes the queries to the desired databases and semantic completeness which lets the queries to make use of the native database features effectively. Location independence is implemented using island concept where databases with similar data models are grouped together. Shim component is used to translate the incoming query into their respective native database queries to take the advantages of the database features. One of the important features provided by BigDAWG is casting where data from one format is been converted in to another and queried [3].

Apache Kylin is an open source distributed analytical engine that supports SQL interface and multi-dimensional analytics on Hadoop for large datasets originally been developed by eBay inc. OLAP cubes are built offline using map reduce process. Recently Apache Spark is been used to speed up the cube build process which is stored in HBase data store. When users issue a query they are routed to be executed against the prebuilt cubes for quicker response, if the desired cube does not exists then the query would be executed against the Hadoop data [4].

Myria is an academia analytical engine been developed and provided as a cloud service. MyriaX is the analytical engine which efficiently executes the queries. MyriaL is the query language supported by Myria for querying [5].

Apache Hive is used for batch processing of big data. It coverts SQL-like queries in to map reduce jobs and executes the queries. HiveQL is the query language used for processing of the data [6].

Kodiak is an analytical distributed data platform which uses materialized views to process analytical queries. Various levels of materialized views are created and stored over petabytes of data. Kodiak platform can maintain these views efficiently and update them automatically. It shows that query execution was three times faster and also used less resources [7].

Apache Lens is another open source framework which tries to provide a unified analytical layer of Hadoop and databases ecosystem using a REST server and query language CubeQL to store and process data cubes [8].

Analytical queries whose aggregates have been stored as OLAP cubes are used in reports and dashboards. These cubes often need to be updated with latest aggregates. Multiple databases are used within an organization for various tasks to store and retrieve the data. Data engineers implement data pipelines to optimize datasets which would be later used by data analysts and data scientists for research. Creating optimized copies involves partitioning and indexing of the base tables. These optimized copies would later be stored in different databases for querying.

Our research focus and solution is to implement a query framework that routes the data analysts and data scientists queries to the optimized copies created by data engineers which are stored, maintained, updated automatically to achieve better query performance and reduce response time.

3. QUERY OPTIMIZER

Apache calcite framework is an open source database querying framework which uses relational algebra for query processing. It parses the incoming coming query and converts them to logical plans and later various transformations would be applied to convert this logical plan into optimized plan which has low cost in execution. This optimized logical plan would be converted in to physical plan to be executed against the databases by using traits. Query optimizer eliminates logical plans which increase cost of the query execution based on cost model been defined. Apache Calcite Schema contains details about the data formats present in the model which is used by schema factory to create schema [9].

The query optimizer applies the planner rules to the relational node and generates different plans with reduced cost by retaining the original semantics of the query. When a rule matches a pattern query optimizer executes the transformations by substituting the sub tree in to the relation expression and also preserves the semantics of it. These transformations are specific to each databases. The metadata component provides the query optimizer with details of the overall cost of the execution of the relational expression and also the degree of parallelism that can be achieved during execution [9].

The query optimizer uses cost-based dynamic programming algorithm which fires rules in order to reduce the cost of the relational expression. This process continues until the cost of the relational expression is not improved subsequently. The query optimizer takes in to consideration CPU cycles, memory been used and IO resource utilization cost to execute the relational expression [9].

One of the techniques which is used to improve query processing is to use the optimized copies been created by data engineers in big data analytics. The query optimizer needs to have the ability to make use of these optimized copies to rewrite the incoming queries to make use of them. Optimizer does this by substituting part of the relational expression tree with optimized copies which it uses to execute query and return the response.

Algorithm

Optimization of relational expression R:

1. Register the relational expression R.
   a. Check for existence of appropriate optimized copy which can be used to substitute relational expression R.
   b. If optimized copy exists, then register the new relational expression R1 of the optimized copy.
   c. Trigger the transformation rules on relational expression R1 for cost optimization.
   d. Obtain the best relational expression R1 based on cost and execute it against the database.
2. If the relational expression R cannot be substituted with an existing optimized copy
   a. Trigger the transformation rules on relational expression R.
   b. Obtain the best relational expression based on cost and execute it against the database.

4. ARCHITECTURE

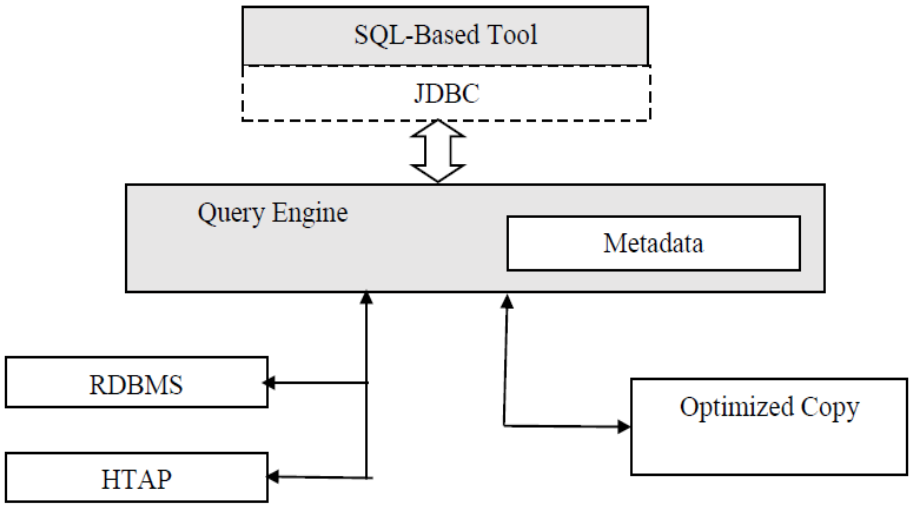

Figure 1. Architecture of the SQL Framework

The analytical queries been executed would be parsed and different logical plans would be generated. The query parser determines if the parsed relational expression can be substituted with the registered optimized copies been created by data engineers automatically. It executes various rules to obtain relational expression with minimal cost. The query would then be rewritten to be executed against the optimized copy and the results would be returned.

SQL-Tool: Includes any tool which can be integrated using JDBC connection and execute SQL analytical queries.

Query Engine: Apache Calcite open source framework that includes query optimizer been extended to include optimized copies.

Metadata: Contains information about the schema and the features of the native databases.
Optimized copy: Optimized tables created by the data engineers.

RDBMS: Includes any relational database to store structured data.
HTAP: Hybrid database which has the scalability of NoSQL data stores.

5. EVALUATION

Analytical queries helps in data driven decision making. In this paper we used NYC Taxi and Limousine Commission (TLC) datasets provided under the authorization of Taxicab & Livery Passenger Enhancement Programs consisting of the green and yellow taxi data [10].

These datasets help in answering questions regarding the passenger drop-pick, drop-off, how Uber ride sharing affect taxi services, do passengers use ridesharing or taxi services to reach common public places, the average time taken to reach the destination, the payment mode used by passengers and other such queries.

The following experimentation setup was made to benchmark and evaluate the query optimizer performance to find the optimized copy of the data and substitute it in the query plan during run time and execute it against the optimized copy Data engineers usually store these optimized copies in the columnar database for faster read access. The tables and data used for the query evaluation was obtained from NYC taxi dataset [10]. Data was stored in Mysql [11], Splice Machine [12] and Vertica database [13].

Evaluation was obtained based on the following setup

a.) Base tables had rows ranging from ~10,000,000 to ~20,000,000
b.) Optimized copy tables had rows ranging from ~1,000 to ~4,000,000
c.) Mysql, Splice Machine and Vertica database were running on single node instance with 2X Intel Quad Core Xeon 3.33GHz, 24 GB RAM and 1TB HDD.
d.) Base table data been stored in HDFS [14].
e.) Optimized copies been stored in Vertica database.
f.) SQL query optimizer used cost based dynamic algorithm to substitute optimized in the query plan.

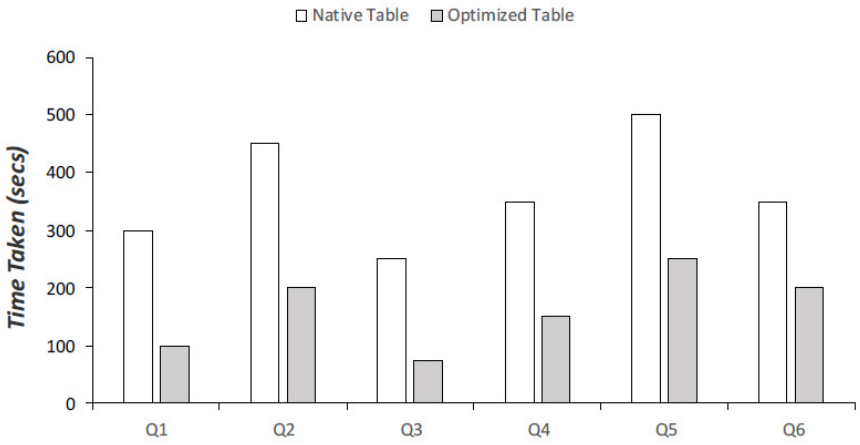

Figure 2. Query Response Time

The two bar graphs show the analytical queries been executed by the query optimizer against the base table and the optimized copy. The query optimizer was successfully able to substitute the optimized copy in the query plan when it existed and improve the performance of the query and its response time.

6. CONCLUSION

In this research we were able to make extensions to cost based query optimizer to make use of the optimized copies to obtain an improved query response time and performance. Various analytical queries were executed to measure the results obtained. The query optimizer was successfully able to substitute the optimized copies when existed during the runtime and modify the incoming query to execute against it. Part of the future work is to extend this query optimizer to the cloud databases.

REFERENCES

[1] Duggan, J., Elmore, A. J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., et al. (2015). The BigDAWG Polystore System. ACM Sigmod Record, 44(3)

[2] V. Srinivasan and M. Carey. Performance of B-Tree Concurrency Control Algorithms. In Proc.ACM SIGMOD Conf., pages 416–425, 1991

[3] A. Elmore, J. Duggan, M. Stonebraker, M. Balazinska, U. Cetintemel,V. Gadepally, J. Heer, B. Howe, J. Kepner, T. Kraskaet al., "A demonstration of the bigdawg polystore system,"Proceedings of theVLDB Endowment, vol. 8, no. 12, pp. 1908–1911, 2015

[4] http://kylin.apache.org

[5] D. Halperin et al. Demonstration of the myria big data management service. In SIGMOD, pages 881–884, 2014.

[6] Fuad, A., Erwin, A. and Ipung, H.P., 2014, September. Processing performance on Apache Pig, Apache Hive and MySQL cluster. In Information, Communication Technology and System (ICTS), 2014 International Conference on (pp. 297-302). IEEE.

[7] Liu, Shaosu, et al. "Kodiak: leveraging materialized views for very low-latency analytics over high-dimensional web-scale data." Proceedings of the VLDB Endowment9.13 (2016): 1269-1280

[8] https://lens.apache.org/

[9] https://calcite.apache.org/

[10] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[11] Luke Welling, Laura Thomson, PHP and MySQL Web Development, Sams, Indianapolis, IN, 2001

[12] https://www.splicemachine.com/

[13] C. Bear, A. Lamb, and N. Tran. The vertica database: Sql rdbms for managing big data. In Proceedings of the 2012 workshop on Management of big data systems, pages 37–38.ACM, 2012

[14] Cong Jin, Shuang Ran, "The research for storage scheme based on Hadoop", Computer and Communications (ICCC) 2015 IEEE International Conference on, pp. 62-66, 2015.