Query Processing and Optimization in the Distributed Architecture using NoSql: a Banking System

1Piyush Gupta, 2Chandelkar Kashinath K.
Dept. of Computer Science & Engineering, Birla Institute of Technology, Mesra, Jaipur Campus
Email: 1piyush.bitjpr@gmail.com, 2Kashinath45@gmail.com

Abstract -The banking sector is one of the best examples who create Big Data with increasing transactions per second across the globe. Each bank is administered by the DBA (Database Administrator) as a technical in charge. DBA makes an arrangement in the branch to divert query from authorized user to the appropriate data repository. The SQL core Engine accepts input stream, processes and optimized to increase efficiency across the network. QO (Query Optimiser) and QE (Query Executor) is an important module that helps in the process. Due to the existing limitations of the system, a cloud based distributed architecture comprising of IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service) is used with NoSql. Hadoop and MongoDB are an example that still retains ACID (Atomicity, Consistency, Isolation, Durability) features of cloud environment.

Keywords- DBA, NoSql, QO, QE, ACID, IaaS, PaaS, SaaS, MongoDB, Hadoop.

I. INTRODUCTION

Considering the banking system as shown in above Fig. 1, each branch has its own servers connected to different users with deployed policies managed by system administrators [1] [10]. Each authorized bank employee communicates to the server and data repository as per the need of a transaction. Every time the query is sent to server in need of information or verification, the data stream has to pass through the server core engine that optimizes and executes the query as per the user requirement [11]. Once the information is collected from the repository, it is filtered and cleaned before delivery in specified format. The data repository is a collection of relational database having tables and their relations that result into problem specified is section 4.

II. PROBLEM

The banking system carries financial information across the channel having approximately thousand transactions per minutes per branch. Each session has data stream is diverted to main server. The server denies the request if it has insufficient amount of both physical and logical resources like RAM and Buffer Size. The increasing structured and unstructured data lead to a new concept called “Big Data” [3]. This huge chunk of data in network consumes resources and cost if not utilized for decision making.

Given system is said to be efficient if it scales with demand [4]. In case of banking, query processing and optimization is a challenge with increasing requests from dynamic nodes for information retrieval on a voice based system.

III. PROPOSED SOLUTION

Query processing and optimization shall be made efficient by using 3 layers. At layer first all physical and logical components are identified. The data input stream in traditional system is given as input to the SQL core engine [11] as shown in Fig-2. This is one of the identified areas for query processing and optimization that needs to explore to enhance.

The DBA (Database Administrator) or an end user who provides query as an input is in layer second. Minor changes in the query input shall save large amount resources in the long run.

Instead of using traditional database [5], using instances of NoSql [1] [3] database system in distributed architecture resolves the issues if left any in layer third.
Fig-2: SQL Core Engine Query processing

Layer 1

The organization or banking sector that needs time to migrate to new system needs to focus on the SQL core engine for the time being. The core engine consists of a traditional storage engine sandwiched between SQL operating system and LPE (Language Processing Execution). It is supported by QE (Query Optimiser) and QE (Query Executor) in the process. Query optimisers being the responsible module for efficiency, the following components are identified.

- **Rewriter** – Applies transformation to the given query and produces equivalent query that is more efficient.
- **Planner** – It is a main module of ordering stage. It examines all possible executions for each query produced in the previous stage and selects the overall cheapest one to generate answers of query. The searching strategy is employed. The determined cost is the running time of the optimizer itself, which should be as low as possible.
- **Algebraic space** – It determines the execution order. A series of action produces the same query answer which differs in performance.
- **Method Structure** - It is used to join available methods like loop, hash join, universal join etc.
- **Cost Model** - Used to estimate the cost of the execution plan for every distinct step. Formula to estimate the cost is dependent on buffer management, Disk, CPU overlaps, sequential and random I/O and size of the buffer pool.
- **Size distribution Estimator** - Specifies how the size (frequency distribution and attributes value) of database relations and (sub) query results are estimated. The DBA has to concentrate on minimizing the optimizer running time by increasing the size of the buffer pool.

Layer 2

The query optimization is possible by writing efficient queries as shown in table-1.

<table>
<thead>
<tr>
<th>General Tip</th>
<th>Write query</th>
<th>Instead of</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specific Column Names instead of</strong></td>
<td><strong>SELECT col_1, col_2, col_3, FROM table_name</strong>;</td>
<td><strong>SELECT * FROM table_name</strong>;</td>
</tr>
<tr>
<td>* in SELECT Query</td>
<td><strong>SELECT Col_1, count (Col_1) FROM table_name WHERE col_1 != testvalue1 AND col_1 != testvalue2 GROUP BY</strong></td>
<td><strong>SELECT Col_1, count (Col_1) FROM table_name GROUP BY Col_1 HAVING Col_1 != _testvalue1 AND Col_1 != _testvalue2</strong>;</td>
</tr>
</tbody>
</table>
We should try to carefully use conditions in WHERE clause.

```
SELECT id, col1, col2 FROM table WHERE col2 > 10;
```

We should try to use NON-Column expression on one side of the SQL query as it will be processed before any other clause.

```
SELECT id, col1, col2 FROM table WHERE Col2 < 25000;
SELECT id, Col1, Col2 FROM Table WHERE Col2 + 10000 < 35000;
```

Table-1: Efficient Query Writing

Layer 3

IV. VALIDATION OF PROPOSED SOLUTION

Fig 3 shows an instance of Hadoop running in Bitnami cloud [16], which is one of the examples of NoSql. After migrating Hadoop in the cloud, a set of queries was given as an input. The above figure discloses an output for a single query. Running multiple Hadoop instances is also possible. Running NoSql not only increases efficiency and security in the banking sector, it becomes more economical supporting ACID (Atomicity, Consistency, Isolation, Durability) factors.

- Does not use SQL as a query language.
- Cloud friendly
- Load balancing is by distributing itself over lots of ordinary chips and Intel based servers.
- It is not a relational database.
- Economic
- Supports key value store, document store, big table and graph data.

V. CONCLUSION

Query processing and optimization issues for banking sector are dealt with using NoSql due to the limitations of existing database systems. Query optimiser being the key module in the existing system is elaborated to enhance efficiency. A 3 layered solution having physical and logical resources in first, human query input as second and using instance of NoSql Software in distributed architecture shall definitely overcome existing banking limitations.

REFERENCES


[2] Mr. Kulkarni, N. N., Dr. Pawar V. P., Dr. K.K Deshmukh (2014) Evaluation of Information Retrieval in Cloud computing based services, Asian Journal of Management Sciences 02 (03 (Special Issue))


[8] Michael L. Rupley,(2013) Introduction to Query Processing and Optimization, mrupleyj@iusb.edu


[12] Yannis E. Ioannidis (2012) Query Optimization, yannis@cs.wisc.edu


[14] William Hersh -2008 Future perspectives Ubiquitous but unfinished: grand challenges for information retrieval, Department of Medical Informatics and Clinical Epidemiology, Oregon Health and Science University, Portland, Oregon, USA.
