IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 10, October 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.625

© 2024 IJIRCCE | Volume 12, Issue 10, October 2024| DOI: 10.15680/IJIRCCE.2024.1210025

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Query Optimization Strategies in Distributed Databases

Abhayanand[1], Dr. M. M. Rahman[2]

Research Scholar, PG Department of CS, Patliputra University, Patna, Bihar, India

Associate Professor, PG Department of Mathematics, A. N. College, Patna, Bihar, India

ABSTRACT: There is no easy solution to the query optimisation issue in large-scale distributed databases; it is NP-hard. As the quantity of relations and joins in a query grows, so does the optimizer's complexity. To discover the best answer, particularly as database sizes grow, researchers are trying to identify the right method. This study reviews several optimisation strategies and shows that integrating the Ant Colony Optimisation Algorithm with other algorithms improves the speed of distributed query optimisation. In light of the advent of high-speed communication networks, a lot of effort is going into studying how to efficiently handle complicated queries within a distributed database setting. In order to increase the logical computer's speed, dependability, availability, and modularity, a distributed database is a set of interconnected databases that are dispersed throughout a network. When compared to centralised environments, distributed ones make query processing a lot more of a pain. The processing of queries necessitates the transmission of data across many locations due to the geographical distribution of the data. Distributed Query Processing Getting Data from Multiple Locations (DQP). In order to get a single set of query results, the query processor pulls data from databases spread across several network locations and processes it using numerous CPUs. Executing a Distributed Query involves three distinct steps. Stage of Local Processing: At this point, the first Algebraic Query based on global relations is prepared and made accessible to the relevant sites for data localisation processing, which includes things like local projections and selects. The optimisation of distributed database queries using Ant Colony Optimisation Algorithm hybrids is an emerging area of research. There is ongoing research on developing and using ACO hybrids to address different kinds of optimisation issues. The results demonstrated the efficacy and practicality of ACO hybrids in solving optimisation issues. When the query size and number of joins increase, research shows that these probabilistic algorithms, when implemented, provide viable solutions in both distributed and relational database management systems. Hybrids of ACO for queries in distributed databases still have a lot of room to grow in terms of optimising solutions and refining search techniques, particularly when relation sizes and complexity grow in tandem with the number of factors impacting the query.

KEYWORD: optimisation, Hybrids, environments, query optimisation, Algorithm hybrids, Distributed Query Processing

I. INTRODUCTION

A distributed system has a number of database servers in the various sites to perform the operations pertaining to a query. Following are the approaches for optimal resource utilization **Operation Shipping-** In operation shipping, the operation is run at the site where the data is stored and not at the client site. The results are then transferred to the client site. This is appropriate for operations where the operands are available at the same site. Example: Select and Project operations.

Data Shipping - In data shipping, the data fragments are transferred to the database server, where the operations are executed. This is used in operations where the operands are distributed at different sites. This is also appropriate in systems where the communication costs are low, and local processors are much slower than the client server.

Hybrid Shipping- This is a combination of data and operation shipping. Here, data fragments are transferred to the high-speed processors, where the operation runs. The results are then sent to the client site.

In query trading algorithm for distributed database systems, the controlling/client site for a distributed query is called the buyer and the sites where the local queries execute are called sellers. The buyer formulates a number of alternatives for choosing sellers and for reconstructing the global results. The target of the buyer is to achieve the optimal cost. The algorithm starts with the buyer assigning sub-queries to the seller sites. The optimal plan is created from local

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

optimized query plans proposed by the sellers combined with the communication cost for reconstructing the final result. Once the global optimal plan is formulated, the query is executed.

Reduction of Solution Space of the Query Optimal solution generally involves reduction of solution space so that the cost of query and data transfer is reduced. This can be achieved through a set of heuristic rules, just as heuristics in centralized systems. Query optimization is one of the key factors affecting the performance of database systems that aim to enact the query execution plan with minimum cost. Particularly in distributed database systems, due to the multiple copies of the data that are stored in different data nodes, resulting in the dramatic increase in the feasible query execution plans for a query statement. Because of the increasing volume of stored data, the cluster size of distributed databases also increases, resulting in poor performance of current query optimization algorithms. In this case, a dynamic perturbation-based artificial bee colony algorithm is proposed to solve the query optimization problem in distributed database systems. The improved artificial bee colony algorithm improves the global search capability by combining the selection, crossover, and mutation operators of the genetic algorithm to overcome the problem of falling into the local optimal solution easily. At the same time, the dynamic perturbation factor is introduced so that the algorithm parameters can be dynamically varied along with the process of iteration as well as the convergence degree of the whole population to improve the convergence efficiency of the algorithm. Finally, comparative experiments conducted to assess the average execution cost of Top-k query plans generated by the algorithms and the convergence speed of algorithms under the conditions of query statements in six different dimension sets. The results demonstrate that the Top-k query plans generated by the proposed method have a lower execution cost and a faster convergence speed, which can effectively improve the query efficiency. However, this method requires more execution time.

II. DISTRIBUTED QUERY OPTIMIZATION

Distributed query optimization involves a large number of query trees to be evaluated, each of which produces the necessary query results. This is largely due to the existence of large amounts of repeated and fragmented information. The goal, therefore, is to find an optimal solution rather than the best solution.

The main problems for optimizing distributed queries are −

- Optimal resource usage in the distributed system.
- Trading by query.
- Reduction of the query's solution space.

Optimal Utilization of Resources in the Distributed System

In order to perform the operations pertaining to a query, a distributed system has a variety of database servers on different sites. The methods for efficient use of resources are below. The operation is carried out on the site where the data is processed and not on the site of the client during operation shipping. The findings are then moved to the client site. This is ideal for operations where operands are accessible at the same place. Example: Operations of Select and Project. The data fragments are moved to the database server during data shipping, where the operations are performed. This is used in systems in which the operands are distributed at different sites. In systems where the communication costs are minimal, and local processors are much slower than the client server, this is also acceptable. This is a combination of shipping of data and activities. Here, data fragments are moved where the procedure runs to the high-speed processors. The findings are then sent to the client site.

Query Trading

The control / client site for a distributed query is called the buyer and the sites where the local queries are performed are called sellers in the query trading algorithm for distributed database systems. A variety of alternatives for selecting sellers and reconstructing the global results are formulated by the buyer. The buyer's aim is to achieve the optimal cost. The algorithm begins with the buyer assigning the seller sites to sub-queries. From local optimised query plans proposed by the sellers in conjunction with the communication costs for reconstructing the final result, the optimal plan is generated. The query is performed until the global optimal plan is formulated.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Reduction of Solution Space of the Query

Generally, the optimal solution requires reducing the space of the solution such that the cost of query and data transfer is reduced. This, like heuristics in centralized systems, can be done by a set of heuristic rules.

Some of the rules are below:

- As early as possible, perform selection and projection operations. This reduces the flow of data across communication networks.
- Simplify horizontal fragment operations by eliminating selection criteria that are not relevant to a specific site.
- Move fragmented data to the site where most of the data is present and perform operations there in the case of join and union operations consisting of fragments located in multiple sites.
- Use the semi-join process to qualify the tuples to be joined. This reduces the amount of processing of data , which in turn lowers the cost of communication.
- In a distributed query tree, merge the common leaves and sub-trees.

Optimizing queries in distributed databases is crucial for improving performance and ensuring efficient data retrieval. Here are some effective strategies:

1. Data Distribution and Partitioning

- **Horizontal Partitioning**: Split tables into rows across different nodes based on certain criteria (e.g., range, hash).
- **Vertical Partitioning**: Divide tables into smaller tables that contain a subset of columns, minimizing the data transferred.
- **Replication**: Store copies of data on multiple nodes to reduce latency for read-heavy queries.

2. Query Rewriting

- **Predicate Pushdown**: Move filtering conditions closer to the data source to reduce the amount of data transferred.
- **Subquery Optimization**: Convert subqueries into joins where possible to improve performance.
- **Materialized Views**: Pre-compute and store complex queries to speed up access.

3. Execution Plan Optimization

- **Cost-Based Optimization**: Use statistics to choose the most efficient execution plan based on estimated costs.
- **Join Algorithms**: Choose the most appropriate join method (e.g., hash join, merge join) based on data distribution and size.
- **Distributed Execution**: Ensure that operations are executed where the data resides to minimize data transfer.

4. Indexing Strategies

- **Distributed Indexes**: Create indexes across distributed nodes to speed up query performance.
- **Secondary Indexes**: Use secondary indexes for non-primary key queries to enhance retrieval speeds.
- **Bloom Filters**: Employ Bloom filters to quickly determine if a record might exist in a partition, reducing unnecessary data scans.

5. Data Locality and Access Patterns

- **Local Queries**: Design queries that can operate on local data as much as possible to reduce network overhead.
- **Affinity-based Routing**: Route queries to the node that holds the relevant data to minimize latency.

6. Caching Mechanisms

- **Result Caching**: Cache the results of frequently run queries to improve response times.
- **Data Caching**: Cache frequently accessed data at the application level or database level to reduce load.

7. Concurrency Control

- **Optimistic Concurrency Control**: Allow transactions to proceed without locks and check for conflicts at commit time.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- **Distributed Locking**: Use distributed locking mechanisms judiciously to prevent bottlenecks while ensuring data consistency.

8. Monitoring and Tuning

- **Performance Monitoring**: Continuously monitor query performance and resource usage to identify bottlenecks.
- **Adaptive Query Optimization**: Adjust query execution strategies dynamically based on runtime statistics.
- **9. Batch Processing**
- **Batch Queries**: Combine multiple queries into a single batch request to reduce the number of round trips to the database.
- **Bulk Operations**: Use bulk insert/update operations to minimize the overhead associated with individual transactions.

10. Using Advanced Algorithms

- **Graph-Based Query Optimization**: Utilize graph algorithms for complex data relationships.
- **Machine Learning**: Apply ML techniques to predict query performance and optimize execution plans.

Implementing these strategies requires a thorough understanding of the underlying data, query patterns, and system architecture. Regularly revisiting optimization strategies can lead to significant performance improvements over time. Query optimization is used for accessing the database in an efficient manner. It is an art of obtaining desired information in a predictable, reliable and timely manner. Formally defines query optimization as a process of transforming a query into an equivalent form which can be evaluated more efficiently. The essence of query optimization is to find an execution plan that minimizes time needed to evaluate a query. To achieve this optimization goal, we need to accomplish two main tasks. First one is to find out the best plan and the second one is to reduce the time involved in executing the query plan.

After parsing and translation into relational algebra expression, the query is then transformed into a form which is usually query tree or graph that can be handled by the optimization engine. Query representation During the optimization phase, the optimization engine performs various analyses on the query data. It applies various rules to the internal data structures of the query to transform these structures into equivalent and efficient representation. It then generates valid evaluation plans based upon the rules applied. From the generated evaluation plans, the best evaluation plan to be executed is determined and passed onto the query execution engine. The final phase in processing a query is the evaluation phase. During the evaluation phase, the best evaluation plan generated by the optimization engine is selected and then executed. The next step is an optimization step that transforms the initial algebraic query using relational algebra transformation into other algebraic queries until the best one is found. A query execution plan is then founded which represented as a query tree includes information about the access method available for each relation as well as the algorithms used in computing the relational operations in the tree. The next step is called code generator, where we generate code for the selected query execution plan. This code is then executed by the run time database processor to produce the query result. The run time database processor has the task of running the query code, whether in compiled or interpreted mode, to produce the query result. If a run time error results, an error message is generated by the run time database processor. Distributed database system vs. Centralized database system Distributed database system: Distributed database management system is basically a set of multiple and logical interrelated database which is distributed over the network. It includes single database which is further divided into sub fragments. Each fragment is integrated with each other and is controlled by individual database. It provides a mechanism that helps the users in distributing the data transparently. Distributed database management system is mostly used in warehouse to access and process the database of the clients at single time. Centralized database management system: Centralized data base is another type of database system which is located, maintained and stored in a single location such as mainframe computer. Data stored in the centralized DBMS is distributed across the network computers. It includes set of records which can easily be accessed from any location by using internet connection such as WAN and LAN. Centralized database system is commonly used in the organizations such as banks, schools, colleges etc to manage all their data in an appropriate manner.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.625| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

III. CONCLUSION

- With the help of centralized database management system, organizations an easily communicate with each other in less time. This approach basically allows the team members of an organization to work on cross-functional projects. It becomes easy for the team members to analyse the data and complete the tasks with good quality.
- By using centralized database system, individuals and teams can easily share their ideas with each other. It becomes easy for the organization to co-ordinate their work with the team members and achieve their business goals.
- Centralized database system also provides high level of security.
- Most of the organizations prefer to use centralized database to reduce the conflicts within the organization. Sharing the information with each other leads to a happier working environment
- Organizations may face issues while using centralized database due to heavy workload requirements.
- While using centralized database system, organizations may have to spend more money to manage and store the data.
- Distributed database system increases the complexity and cost of the organization. It becomes difficult for the organization to maintain and manage the local database management system due to which organizations may face difficulty to establish a network between the sites.
- In distributed database system, it becomes difficult for the organizations to control the replicate data.
- While using distributed database system, organizations cannot use static SQL.

REFERENCES

[1] P. G. Selinger et al., "Access path selection in a relational database management system," 1979.
[2] D. Kossmann, "The state of the art in distributed query processing," 2000.
[3] S. Ding et al., "A distributed query optimizer for cloud databases," 2018.
[4] A. Deshpande et al., "Adaptive query processing," 2007.
[5] V. Markl et al., "Progressive query optimization for federated queries," 2004.
[6] K. Karanasos et al., "Distributed query processing on big data systems," 2017.
[7] T. Kraska et al., "The case for learned index structures," 2018. [8] R. Marcus et al., "Deep reinforcement learning for join order enumeration," 2018.
[9] J. Ortiz et al., "Learning state representations for query optimization with deep reinforcement learning," 2018.
[10] S. Wu et al., "Query routing in a peer-to-peer semantic overlay network," 2004.
[11] A. Vulimiri et al., "Global analytics in the face of bandwidth and regulatory constraints," 2015.
[12] B. Ding et al., "Improving optimizer cost models with machine learning," 2019.
[13] M. Serafini et al., "Elastic database partitioning with latency guarantees," 2016.
[14] H. Mao et al., "Learning scheduling algorithms for data processing clusters," 2019.
[15] J. Zhu et al., "Pythia: A customizable hardware prefetching framework using online reinforcement learning," 2020.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details