Computing, Information Systems, Development Informatics & Allied Research Journal
Vol 7 No 1 March 2016 - www.cisdijournal.net

Query Processing and Optimization in a Distributed Database System

[1]Daodu, S. S., [2]Akhatuamen, S., [3]Folorunso, O. and [4]Ukaoha, K.C.
[1&4]Department of Computer Science, University of Benin, Nigeria
[2]Department of Mathematics and Computer Science, Federal Polythecnic, Auchi.
[3]Department of Computer Science, Federal University, Oye-Ekiti, Nigeria.

ABSTRACT

The concept of distributed database systems has emerged as a natural solution to the information problems of geographical dispersed organization. In this work, we are concerned with processing a query in a distributed relational database system implemented on ping functionality in the communication network. Optimizing the response time of such queries is a challenging task due to the unpredictability of server performance and network traffic at the time data shipment. This may result in the selection of an expensive query plan using a static query optimizer. We developed a method which accurately and efficiently estimates the size of an intermediate result of a query. This method provides the basis of the query optimization algorithm, since the distributed query optimizer is known to be intractable.

Keywords: Distributed database, communication network, query optimizer, intractable

1. INTRODUCTION

The recent telecommunication boom has encouraged business expansion resulting in the decentralization of data while increasing the needs for instant information access. A distributed database system is a collection of multiple and logically interrelated databases distributed over a computer network, (Sartawi and Jafar, 2003). Logically, data belong to the system but physically is spread over the sites of the network making the distribution invisible to the user (Haas and Swami, 1992). Each site acts as an autonomous database with its processing capability and data storage capacity. The advantage of this distribution results in achieving availability, modularity, improved performance and reliability. As in traditional centralized database systems, distributed database systems (DDBSs) must provide an efficient user interface that hides all of the underlying data distribution details of the DDB from the users. The use of a relational query allows the user to specify a description of the data that is required without having to know where the data is physically located.

Distributed query processing is the process of selecting data from database located at multiple sites in a network and distributed processing performs computations on multiple CPUs to achieve a single result. Any Structured Query Language data manipulation statement that references tables at sites other than the site an application program is submitted for compilation is called distributed query and needs to be processed. Query optimization is a difficult task in a distributed client/server environment as data location becomes a major factor (Alom et al., 2009). In order to optimize queries accurately, sufficient information must be available to determine which data access techniques are most effective. Query processing is observed to be more difficult in distributed environment than in centralized environment due to the reasons stated below:

i. A large number of parameters effect the performance of distributed queries
ii. Relations involved in distributed query may be fragmented and/or replicated
iii. With many sites to access, query response time may invariably become high

The performance of DDBS is dependent on the ability of the query optimization algorithm to derive efficient query processing strategies, while distributed database management system (DDBMS) query optimization algorithm attempts to reduce the quantity of data transferred. The act of minimizing the quantity of data transferred is a desirable optimization criterion since more data transported across telecommunication networks requires more time and labour. The distributed query optimization has several problems that relate to:

- Cost model
- Larger set of queries
- Optimization and execution cost tradeoff
- Optimization/ re-optimization interval

A distributed database system is the combination of two different technologies used for data processing: Database Systems and Computer Networks. The main component of a database is the data which is basically collection of facts about something. This something may be the business data in case of a business corporation, strategic data in case of a military database etc.

For the efficient continuity of distributed database, there are two processes : replication and duplication. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be very complex and time consuming depending on the size and number of the distributive databases. This process can also require a lot of time and computer resources. Duplication on the other hand is not as complicated. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, changes to the master database only are allowed so as to ensure that local data will not be overwritten (Raipurkar and Bamnote, 2013)

2. HISTORICAL BACKGROUND AND REVIEW

Raipurkar and Barnote (2013) studied query processing in distributed database through data distribution and observed that distributive database makes use of two processes in order to ensure that its data are up to date and current. These processes are: replication process and duplication. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be very complex and time consuming depending on the size and number of the distributive databases. This process can also require a lot of time and computer resources. Duplication on the other hand is not as complicated. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, changes to the master database only are allowed. This is to ensure that local data will not be overwritten. Both of the processes can keep the data current in all distributive locations.

Hazra et al (2009) studied the Network Issues in Clock Synchronization on Distributed Database and found that there are two major issues associated with distributed databases. These are:

a) Issue with sensor data acquisition: In distributed databases, data acquisition which is carried out in one processor must always maintain a fixed timing relationship with the algorithms which process this sensor data. These algorithms may typically be performed in a set of different processors. This fixed timing relationship cannot be maintained unless all clocks of the processors partitioning in the execution of the algorithm are synchronized. Similar arguments hold for the processors, which are entrusted with the job of data distribution to launch vehicle control system actuators.
b) Precedence Issue; There may exist a precedence relationship among tasks in different processors. Say, task A in one processor can start execution only upon the completion of Task B in another processor. Since the pre-run time scheduling algorithms of Xu and Parnas (1993), which are commonly used to schedule tasks in different processors use the logical clocks maintained by each individual processor, the only way to guarantee the precedence relationship among tasks distributed across processors is to maintain good clock synchronization among processors. The authors maintained that clock synchronization can be achieved normally by two distinct methods which can either be external synchronization or through internal synchronization. External synchronization tries to maintain processor clock within a specified deviation from an externally maintained time reference, using phase locked oscillators Shin and Ramamathan, (1988).

For this purpose, each processor should be connected to a stable clock through an external links, to provide nanosecond level accuracy to physical clocks. One needs to execute one pass of a synchronization algorithm initially or whenever a new processor joins in. But Cesium or Rubidium-based stable automatic clocks are expensive and to carry them on board satellite launch vehicles makes the overall cost prohibitive. These issues are also highly relevant in the case of automotive industry where expensive hardware clocks in each node may not be cost-effective. Moreover, in a distributed system, it may well be near impossible to use external clocks, as the cables interconnecting these clocks introduce distortions, which could be higher than the inherent stability of the external clock. Internal clock synchronization addresses these problems by using software algorithms which ensure that the logical clocks used by the processors in different nodes are consistent within limits, irrespective of the drift in the physical clocks.

Jagannatha et al, (2013) studied load balancing in Distributed Database System using resource allocation approach. They proposed a methodology for performance analysis of load balancing by sharing of resources, allocation of fragment replicas and transaction in distributed database system. They observed that some research work has been done on the issue of load balancing, pointing that Wiesmann and Schiper (2005) addressed the performance comparison of database replication techniques based on total order broadcast technique on classical replication scheme like distributed locking. Jagannatha et al, (2013) maintained that performance has little influence in a LAN setting; total order broadcast-based techniques are very promising as they minimize synchronization between replicas. The authors also observed that Gu et al (2006) described the design and analyzed the data replication strategies with the model of Dynamic Window Mechanism algorithm jointly implemented with different types of object replacement strategies with limited buffer capacities. In Jagannatha et al, (2013) proposal, the authors assumed that application is model using use-case diagram, UML. Each use case U_i consisting of k parallel and dependent sub queries, some queries are assigned same resources for load sharing and replica available.

They gave an example of a use-case with-draw case which has: withdrawn, checks balance, overdraw, and are parallel sub queries. They snoted that these tasks share R_j resources and share the R_j capacity and expense P_j according to its processing capacity. They also noted that the tasks allocate R_j resources for load sharing and that each sub-query may have to share R_j and may be waiting to share R_j. where the assignment of these queries into these resources are based on data replica available. The entry a_{ij} is the allocation of query in the U_i use case to the R_j resources and is based on data availability.

3. SYSTEM SEQUENCE DIAGRAM

System state diagrams are a standard modeling technique for modeling system of varying complexities. A sequence diagram describes the interaction between classes. The interaction represents the order of messages that are exchanged between classes. Sequence diagrams show interaction between classes arranged in a time sequence. In sequence diagram, interactions are message exchanges that take place between classes to accomplish a purpose.

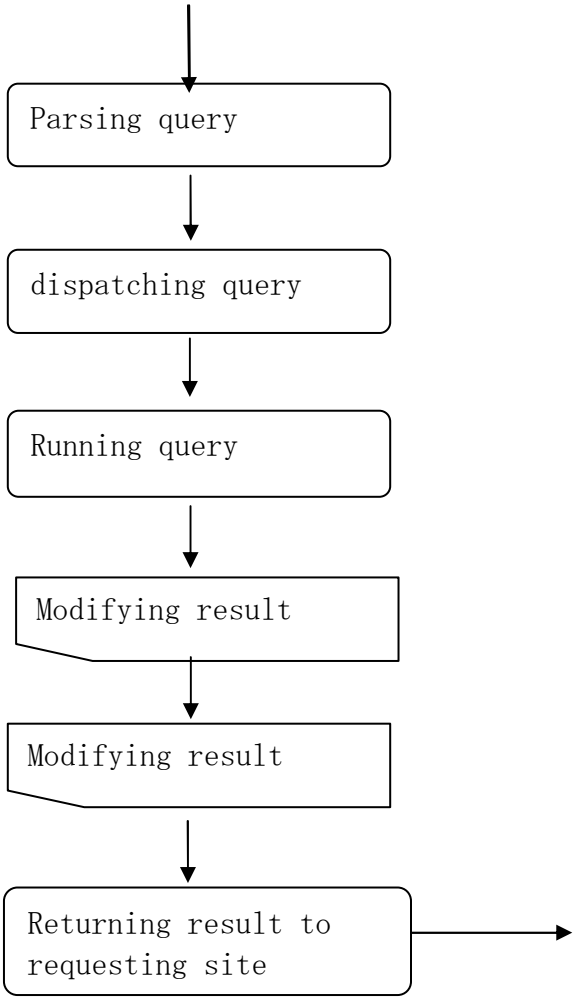

Figure 1: System state design diagram of a query processing and optimizer in a DDMS

To design this model in a distributed database environment the following schema is drawn to show how the various sites in the network can actually send requests to the database and such queries are executed optionally. It should be noted that in a distributed database system, the data may be in the different sites replicated or fragmented. Based on this it is left for the query optimizer to look at which of the alternative is more cost effective.

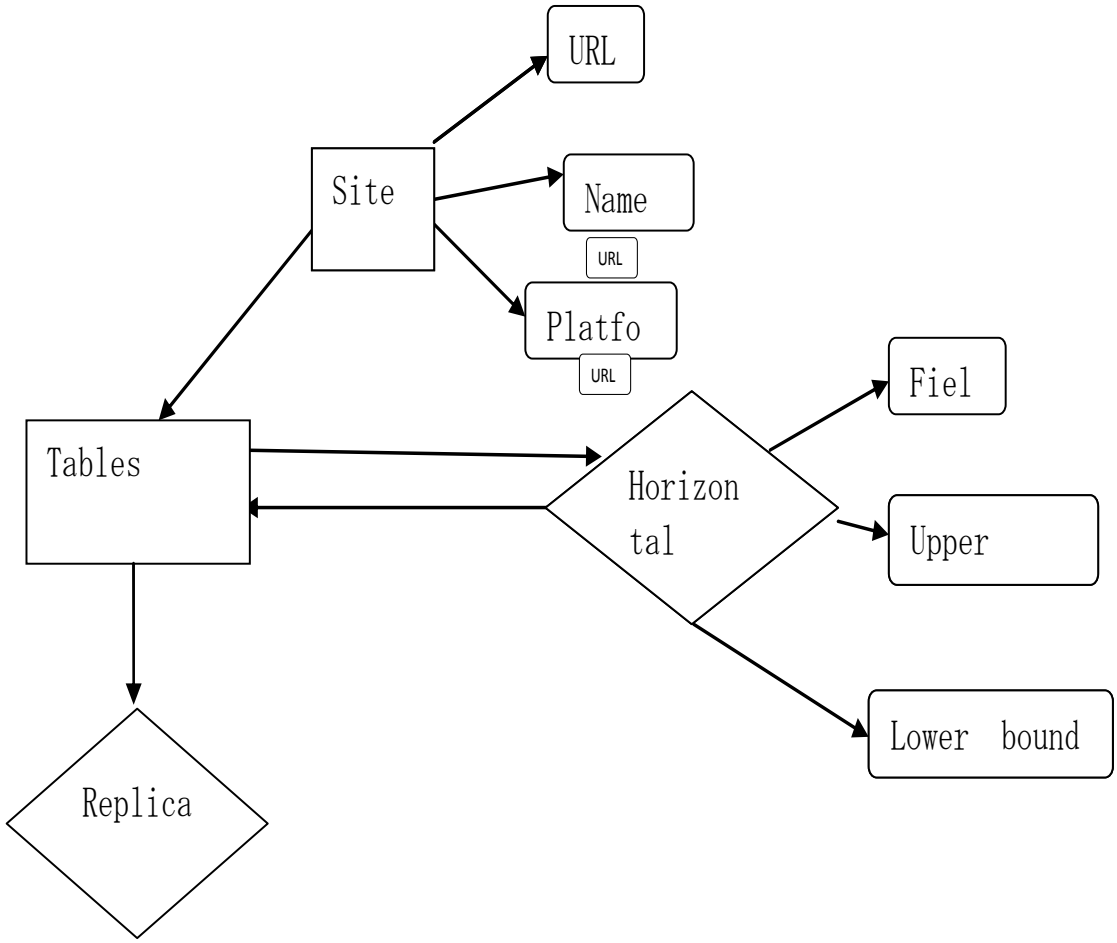

Figure 2: DDMS Schema

From the above schema, one site contains many tables and a table can have many replica of that data. More also, a table can have many fragments as the case may be. This is so because in a distributed environment more than one site can have the same data in different fragments so that when queries are requested, the underlying algorithm for the query processing is used to give the optimal solution to return the request in terms of cost and response time

4. DESIGN

In the design, this optimization of processing of queries in a distributed database system, the ping functionality was used. This was done by running queries at the closest site from the site that initiates the query.In this research work, only simple queries like SELECT*FROM were used, but others like ADD, AVG were embedded within the queries. We also considered only horizontal fragmentation with two sites in the network, and these fragments were based on one field. A table is only queried if it has no fragments, otherwise the fragments are used for the query.

4.1 Ingres Algorithm

1. Decompose each multi-variable query into a sequence into a sequence of mono-variable queries with a common variable.
2. Process each by a one variable query processor by:
   i. choosing an initial execution plan(heuristics)
   ii. order the rest by considering intermediate relation sizes
   iii. replace an n variable q by a series of queries

$q_1 \longrightarrow q_2 \longrightarrow \cdots \longrightarrow q_n$

where q_i uses the result of q_{i-1}

5. CONCLUSION

To a large extent, the success of a DBMS lies in the quality, functionality, and sophistication of its query optimizer, since that determine much of the system's performance. In this work, we have presented an abstraction of the architecture of a query optimizer and focused on the techniques currently used in most commercial systems for systems for various modules. Optimization is much more than transformations and query equivalences. The infrastructure for optimization is significant. Designing effective and correct SQL transformations is hard, developing a robust cost metric is elusive, and building extensive enumeration architecture is a significant understanding. Despite many years of work, significant open problems remain. However, an understanding of the existing engineering framework is necessary for making effective contribution to the area of query optimization.

REFERENCES

1. Alom, B. M., Henskens, F. and Hannaford, M. (2009). Query Processing and Optimization in Distributed Database Systems. International Journal of Computer Science and Network Security, 9(9), 143- 152.
2. Gu, X. Lin, W. and Veeravalli, B. (2006). Practically Realizable Efficient Data Allocation and Replication Strategies for Distributed Databases with Buffer Constraints IEEE Transaction on Parallel and distributed systems, 17(9).
3. Haas, P. and Swami, A. (1992): Sequential sampling procedures for query size estimation. In proceedings of the ACM-SIGMODD Conference on the Management of Data, pp. 341-350.
4. Hazra, R., Bhattacharyya, D., Dey, S., Feruza, S. Y.,and Furkhat, T. A. (2009). Network Issues in Clock Synchronization on Distributed Database. *International Journal of Database Theory and Application.* 2(2).
5. Jagannatha, S., Geetha, D. E., Suresh Kumar, T. V. and Kauth, K. R. (2013). Load Balancing in Distributed Database System Using Resource Allocation Approach. International Journal of Advanced Research in Computer and Communication Engineering. 2(7). 2529-2535
6. Raipurkar, A and Bamnote, G.R. (2013). Query Processing In Distributed Database through Data Distribution. International Journal of Advanced Research in Computer and Communication Engineering. 2, Issue 2.
7. Sartawi, B. and Jafar, H. (2003). Distributed Query Processing, Addison Wesley
8. Shin, K. G. and Ramamathan,P. (1988). Transmission Delay in Hardware Clock *Synchronization*. IEEE Transactions on Computers, 37(11), 1465-1467.
9. Wiesmann, M. and Schiper, A. (2005). Comparison of Database Replication Technique Based on Total Order Broadcast. IEEE Transactions on Knowledge and Data engineering, 17(4).
10. Xu., J. and Parnas,D. L. (1993). On Satisfying Timing Constraints in Hard Real-Time Systems, IEEE Transactions on Software Engineering, 19(1) , 70-84.