IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 26, Issue 4, Ser. 3 (July. – August. 2024), PP 19-23
www.iosrjournals.org

Study On Performance Improvement Of Distributed Query Optimization Algorithms

Mohammad Kamal Hossain Foraji[1] And Md. Humayun Kabir[2]
[1]Department Of Computer Science And Engineering, Jahangirnagar University, Bangladesh
[2]Department Of Computer Science And Engineering, Jahangirnagar University, Bangladesh

Abstract:
This paper investigates the performance of different distributed query optimization algorithms and presents the outcome of this study to suggest efficient query processing techniques for performance improvement in distributed query processing. In query processing, each query is converted to a series of data manipulation operations. We have analyzed a query in different strategies and showed that efficient technique can be chosen to reduce communication time in distributed database management. The speed of the communication network integrating the Distributed database system (DDBS) environment also affects the performance of the distributed query optimization algorithms. The factors considered here are the distribution of data among the sites, communication costs, and locally available information. Some distributed query optimization algorithms are presented and a few features for improving their performance are reviewed. We have also analyzed the limitations of the existing distributed query optimization techniques to reduce response time and overall communication costs.

***Keyword**: Distributed Database, Distributed Query Optimization, Communication Costs, Data Manipulation Operations, Dynamic Programming, Response Time, Query Execution Plan, Optimization Algorithms, Query Processing Strategies.*

Date Of Submission: 21-07-2024 Date Of Acceptance: 31-07-2024

I. Introduction

A distributed database management system (DDBMS) facilitates communication between various syst em sites by allowing users to store, retrieve, and alter data and information situated at widely distant locations th at are connected via a communication network [1, 2]. Distributed query processing is essential to retrieve and manipulate data and information located at different sites by processing and executing the query in a layered approach using fragmented and replicated data [3].

Any high-level query for a distributed database can be automatically converted by a distributed query processor into a sequence of effective low-level database operations. Afterwards, the local databases that make up the distributed database undergo this sequence of operations. In this research, we investigate various performance improvement techniques for distributed query processing [2, 4].

In query optimization, a query execution plan is produced which represents the query execution strategy with minimum cost [1]. Several issues are investigated. We analyze the factors for improving the efficiency of the existing query optimization techniques, and data distribution techniques. A comparative analysis of various query optimization algorithms is presented. Finally, we suggest how to minimize the query execution cost for efficient query processing.

II. Related Work

Query processing in the distributed database environment involves various factors for execution of the distributed query while executing the sub-queries by the servers of the multiple sites concurrently. These machines of the distributed database environment run the same database management system software [4]. The database which spans over the distributed environment is fragmented and replicated over the sites. The best performance in executing a distributed query will require the minimum cost, i.e., minimum CPU time, minimum transfer time [5]. Searching the optimal tasks for a specific query in the search space and execution of the parts of a query plan by the multiple sites may reduce query execution time by employing parallelism [6] with simple alteration of the dynamic programming (DP) algorithm. Simulated annealing and iterative improvement are the two approaches to search for a plan in a search space [7, 8] of equivalent plans for a given query.

A complex query can be reduced to a simple query by applying simplification and rewriting steps. Redundant predicates in a query are eliminated by applying the idempotency rules to obtain a simple query [9, 10]. One approach is to view the low-cost query as the more efficient query than its original counterpart. By improving the performance of the hardware, software, optimization algorithms and network infrastructure, the overall query execution time, i.e. query execution cost and response time may be reduced in the distributed database environment.

A distributed database is a collection of databases kept in multiple locations within a distributed system. The processing of queries in remote databases is highly difficult, but very important for information management [9]. Some of the major conceptual issues encountered while developing an optimizer for a distributed DBMS have been addressed. First, estimating the response time of a query plan in a distributed database system is far more complicated than in a centralized system due to the possibility of parallelism [5]. To create an accurate estimate of the response time for a specific plan, a distributed system cost model must effectively simulate the sources and barriers to parallelism. Modeling data independent inter-operator parallelism may be considered important.

III. Comparative Analysis Of Query Optimization Algorithms

Various techniques involved in distributed query processing. Some of the important features are query optimization, inherent parallelism through intra-query and inter-query parallelism, memory cost-reduction through distributed INGRES and R* algorithms, fragmentation and replication of data, networking and network transparency [4] etc. Query optimization can be achieved employing various algorithms including System R, R*, SDD-1, IDP-M, INGRES, and dynamic programming [11]. A distributed database system is a software solution that allows users to manage dispersed databases while also making the distribution transparent [10]. Distribution transparency takes the notion of data independence a step further by making distribution invisible to users. A brief comparison between distributed INGRES and R* algorithms is given in the following section.

A) Distributed INGRES Algorithm

A dynamic query optimization technique is used by the distributed INGRES algorithm, which recursively divides a query into smaller chunks, or sub-queries [4]. It is used to reduce the overall response time and the communication time. This algorithm makes use of horizontal fragmentation. In broadcast networks, the same data unit can be transferred from one site to all other sites in a single transfer [9]. This algorithm is executed at the master site, which initiates the query.

B) R Algorithm*

R* query optimization algorithm optimizes static queries using the solution space and cost function, i.e., IO Cost + CPU Cost. This algorithm conducts an exhaustive search of all potential techniques to determine the one with the lowest cost [12]. The master site's optimizer makes all inter-site decisions. The other sites, have relations involved in the query, make the local decisions and generate local access plans for the query [13]. The optimizer's objective function is the total time function, which includes local processing and communication costs. The differences between distributed INGRES and R* algorithms are shown in Table 1 [3, 4].

Table no. 1. Distributed INGRES and R* algorithms [3, 4].

Particulars	Distributed INGRES Algorithm	R* Algorithm
Optimization Timing	Dynamic	Static
Objective Function	Response time or total cost	Total Cost
Optimization Factors	Message size, Processor	No. of Msg, Msg Size, IO and CPU
Network Topology	General/Broadcast	General/ Local
Fragments	Horizontal	No

Dynamic query optimization incorporates query deconstruction and optimization, as well as execution [14]. The query optimizer dynamically creates the query execution plan (QEP), which then instructs the DBMS execution engine to execute the query's activities.

IV. Result Analysis

Query processing in distributed databases plays a vital role as we can optimize one query in multiple different ways. It also depends on the speed of the communication network. Consider the following example configured between 2 sites A and B as shown in Figure 1.

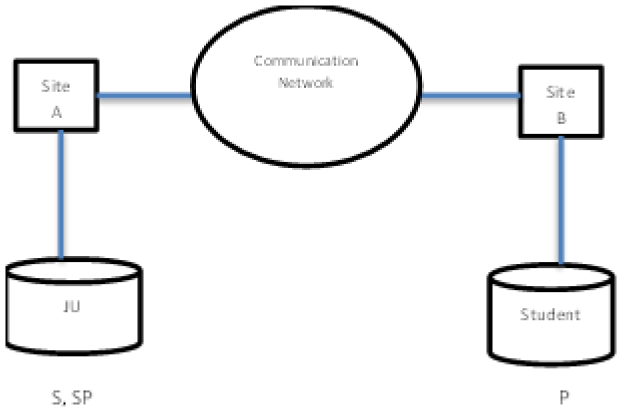


Figure 1: DDBS Environment.

In the above configuration, the database JU has user JU1 and database STUDENT has user STUDENT1. S represents the supplier table with two attributes-supplier number (SN) and City. It contains 10,000 tuples stored at site A. P represents the part table with the attributes part number (PN), and Color. It contains 1,00,000 tuples stored at Site B. Another table is Shipment denoted by SP with two attributes- supplier number (SN), and part number (PN). It contains 1,00,000 tuples located at Site A. There are 3 tables distributed in Site A and B. Assume that every stored tuple is 25 bytes (200 bits) long. The data distribution in the sites is shown in Table 2.

Table no. 2. Data Distribution Among the Sites.

Table	Attributes	No. of Tuples	Site
S	SN, City	10,000	A
P	PN, Color	1,00,000	B
SP	SN, PN	1,00,000	A

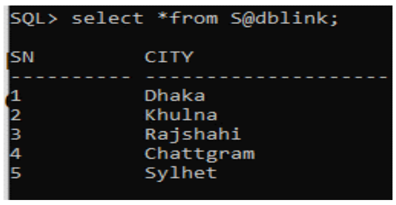
Consider the following query to be processed.
Query 1. *"Find supplier numbers for Dhaka supplier of red parts."*

So, if we write the query using relational algebra, we get it as follows:
$(S \quad _{SN} SP \ltimes_{PN} P) \ltimes$
$\Pi_{SN} (\sigma_{City} = \text{"Dhaka" and Color} = \text{"Red"})$

Supplier (S) operated on natural join Shipment (SP) natural join Parts (P) where these two natural joins take place SN and PN. Here PN is a part number and SN is a supplier number.

To run the following SQL query, a database link '*dblink'* is created between the two databases 'JU' in site A and 'STUDENT' in site B. The following SQL statement is used for creation of database link on student database so that the user of 'STUDENT' database can access to the 'JU' database:
create database link dblink connect to ju1 identified by oracle using 'JU';

Here, 'dblink' is the name of database link, 'oracle' is the password of the 'ju1' user. For the link testing, we can run the SQL query: *select * from S@dblink*. We obtain the following data from table S using the remote site A as shown in Figure 2.

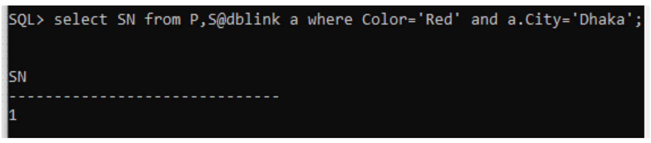


```
SQL> select *from S@dblink;

SN         CITY
---------- --------------------
1          Dhaka
2          Khulna
3          Rajshahi
4          Chattgram
5          Sylhet
```

Figure 2: SQL Query Result.

At the same time, we write the above query in Oracle SQL statement on distributed environment and get it as follows as shown in Figure 3.
select SN from P, S@dblink a where Color = 'Red' and a.City = 'Dhaka';



```
SQL> select SN from P,S@dblink a where Color='Red' and a.City='Dhaka';

SN
-----------------------------
1
```

Figure 3: SQL Query Result.

Consider the following attribute values of the table P stored at site B for the different strategies used for analyzing query 1.
No. of red parts = 10
No. of shipments by Dhaka supplier = 1,00,000

Hence, out of these 1,00,000 tuples, only 10 tuples are red in color, and 1,00,000 number of shipments have been done by Dhaka supplier.

Communication assumptions:
Data rate = 50,000 bits per second
Access delay = 0.1 seconds
Data volume = No. of tuples * 200 bits

Total Communication time [4]
= (Total access delay) + (Total data volume / Data rate)

Strategy 1: Move P to A, i.e., we see table P was only at Site B. So, let us move it to A so that all the information will move to Site A. If the data processing is done, the estimated communication time will be 6.67 minutes.

Estimation:
Total Communication time = 0.1+ (1,00,000 * 200) / 50,000
= 400.1 seconds = 6.67 minutes

Strategy 2: Move S and SP to B. Here two tables are to be moved.

Estimation:
Total Communication time
*= 0.2+((10,000 + 1,00,000) * 200) / 50,000*
= 440.2 seconds = 7.34 minutes
Here, communication time is more than Strategy 1, which is 7.34 minutes.

Strategy 3: Move Dhaka shipment to B. Only Dhaka shipments will be moved to table B.

Estimation:
Total Communication time
*= 0.1+(1,00,000*200) / 50,000*
= 400.1 seconds = 6.67 minutes

Strategy 4: Move red part to A. This is having part with 1,00,000 tuples. But only 10 tuples are with having red color. It is better to move the red part to this Site A.

Estimation:
Total Communication time
*= 0.1+(10*200) / 50,000*
= 0.14 seconds = 0.0023 minutes

So, the minimum data communication time is 0.14 seconds, and it seems the best of all solutions mentioned above. The result statistics is shown in Table 3 and Figure 4.

Table no. 3: Query Execution Statistics.

Strategy	Access Time (Sec)	Data Volume (Bits)	Communication Time (Minutes)
1	0.1	20000000	6.67
2	0.2	22000000	7.34
3	0.1	20000000	6.67
4	0.1	2000	0.0023

Figure 4 shows the access time, data volume and communication time using Table 3.

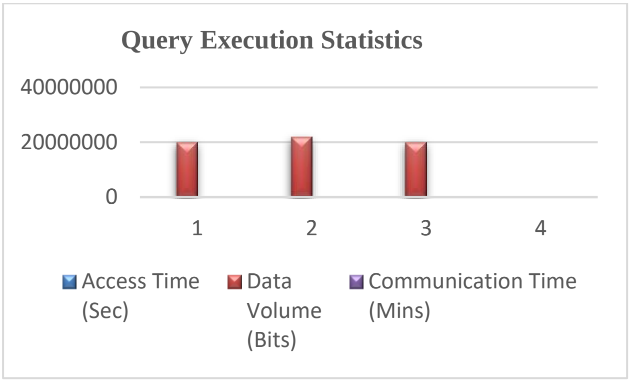


Figure 4: Query Execution Statistics Chart.

V. Conclusion And Future Work

The network transmission rate is reliant on network traffic and hence varies over time. The cost of sending data tables between sites over the network contributes significantly to the entire execution time and should be factored into the cost model [4, 5]. So, the performance of query processing in distributed database system relies on the query execution strategy, query performance and query optimization. Several strategies are examined to show how choice of data movement among the sites can reduce query execution time. Research will be continued further to improve the performance of distributed query execution in DDBS environment applying the distributed INGRES algorithm and R* algorithm.

References

[1] S. Padia, S. Khulge, A. Gupta, P. Khadilikar. Query Optimization Strategies In Distributed Databases. International Journal Of Computer Science And Information Technologies. Vol 6(5), 2015.

[2] B. Johnsirani And M. Natarajan. An Overview Of Distributed Database Management System. International Journal Of Trend In Research And Development. Vol 2(5), Sep-Oct 2015.

[3] D. Sukheja And U. K. Singh. Novel Distributed Query Optimization Model And Hybrid Query Optimization Algorithm. International Journal Of Computer Applications, Vol. 75(17), 2013.

[4] M. Tamer Ozsu And Patrick Valduriez, Principles Of Distributed Database Systems, Prentice Hall, 3rd Edition, 2011.

[5] William T. Bealor. Semi-Join Strategies For Total Cost Minimization In Distributed Query Processing. M.Sc. Thesis, University Of Windsor, 1995.

[6] B.M. Monjurul Alom, Frans Hanskens And Michael Hannaford. Query Processing And Optimizations In Distributed Database Systems. International Journal Of Computer Science And Network Security (Ijcsns), Vol. 9(9), 2009.

[7] Subir Bandyopadhyay, Qiuling Fu Joan Momssey, And A. Sengupta. A Multiattribute Semijoin Operation For Query Optimization In Distributed Databases, 1996.

[8] Rosana S. G. Lanzellote And Atric Valdureiz. Extending The Search Strategy In A Query Optimizer. Proceedings Of The 17th International Conference On Very Large Databases. Barcelona, September 1991.

[9] G. M. Thomas Neumann. Analysis Of Two Existing And One New Dynamic Programming Algorithm For The Generation Of Optimal Bushy Join Trees Without Cross Products. In Proceedings Of The 32nd International Conference On Very Large Data Bases, Pages 930–941. Vldb Endowment, 2006.

[10] Norvald H. Ryeng. Improving Query Processing Performance In Large Distributed Database Management Systems, Phd Thesis, Norwegian University Of Science And Technology, 2011.

[11] Epstein, R., Stonebraker, M. And Wong, E. Query Processing In A Distributed Relational Database System. Proc. Acm Sigmod Int. Conf. Management Of Data, 1978 , Pp. 169–180.

[12] Guy M. Lohman, C. Mohon, Laura M. Haas, Bruce G. Lindsay, Patrica G. Salinger And Paul F Wilms Query Processing In R*. Ibm Research Laboratory, Rj 4272 (46860) 4/23/83, Computer Science, Carnegie-Mellon University, 1984.

[13] P. G. Selinger And M. Adiba. Access Path Selection In Distributed Database Management Systems. In Proc. First Int. Conf. On Databases, 1980, Pp. 204-315.

[14] Abhayanand And M. M. Rahman. An Overview Of Query Optimization In Distributed Relational Databases. Int. Journal Of Creative Thoughts, 2024, Pp. 204-315.