REVISITED PERFORMANCE ISSUES IN CONCURRENT TRANSACTION EXECUTION IN DISTRIBUTED DATABASE MANAGEMENT SYSTEM

[1]Mrs. Shefali Naik, [2]Dr. Samrat Khanna
[1]Assistant Professor, School of Computer Studies, Ahmedabad University
[2]Prof. & Head-MSc.(IT), Inst. Of Science & Technology for Advanced Studies & Research
Sardar Patel University
Email: [1]Shefali.naik@ahduni.edu.in, [2]sonukhanna@yahoo.com

Abstract – When the database is very large and accessed from remote machines, it is to be partitioned and stored on several machines or sites. For the faster data access, the partitions must be replicated and stored on local machines. It is a difficult task to update and synchronize data in all the replicas. There are various methods available for optimization. In the paper, the various issues related to performance issues of concurrent transaction execution in DDBMS are discussed.

Index Terms-Distributed Database, Partition, Performance issues, Replica, Concurrent transaction

I. INTRODUCTION

The database which is fragmented and stored on different machines is called Distributed Database. Different applications which are using this database access data stored in these fragments from different machines through transactions. Applications request data simultaneously which may result in conflict. Conflicting requirements are taken care by the Database Management System. The simultaneous execution of transaction is called concurrent execution. Without proper concurrency control and integration of data, concurrent execution may result into inconsistent data. Transactions in distributed database require distributed processing which is very difficult to manage. The data which are distributed across various machines are stored in different database management systems, it is called heterogeneous distributed database. In DDBMS, for faster data access, sometimes the fragments are replicated. Since database is distributed, there are some performance issues which need to be resolve for better execution of transactions. These issues are discussed in the paper.

II. PRINCIPLES OF DDBMS

Distributed database works on the following principles[1].

A. Partitioning[1] : Data is distributed through partitions across multiple computers. There are different types of partitioning techniques to divide data across the distributed system. The popular methods of partitioning are horizontal partitioning, vertical partitioning and hybrid partitioning. Partition is also called fragmentation.
B. Replication[1] : The partitioned data are replicated or duplicated and stored across multiple computers for fast access. Synchronization is required in all the replicas.

C. Transparency at different levels[1] : When data is distributed, full transparent access should be provided at different levels. Full transparent access means that user can write and execute a query without paying any attention to distribution of data, location, partition or replication. Transparency should be provided in the form of Data independence, Network transparency, Replication transparency and Fragmentation transparency.
D. Distributed Query Processing[1] **:** Query processing deals with optimization. It involves designing of algorithms that analyze queries and convert them into a series of data manipulation operations.
E. Distributed Concurrency Control[1] : Concurrency control involves the synchronization of data accesses so that the integrity of the database is maintained.
F. Replication Protocols[1] : If the distributed database is replicated partially or fully, it is necessary to implement protocols that ensure the consistency of the replicas.
G. Heterogeneity[1] : Heterogeneity may occur in various forms in distributed systems such as hardware, network, DBMS, Data models, query languages, transaction management protocols, etc. Representing data with different modeling tools creates heterogeneity because of the inherent expressive powers and limitations of individual data models.

III. DISCUSSION OF DDBMS ISSUES

A. Replica Synchronization[1] : To provide faster data access, the database fragments are replicated on different machines. When data on any one replica is updated, the changes must also be reflected on other replica to provide up-to-date information. It is very difficult to synchronize the replicas.
B. ACID properties : Many of the present time applications are by nature distributed such Web-based applications, E-commerce applications, multimedia applications such as medical imaging, etc. The transaction in Relation Database Management System should satisfy the ACID(Atomicity, Consistency, Isolation, Durability) properties[2]. But in distributed database it is not possible to get all the four together. Eric Brewer [2010] has given the CAP theorem which states that it is impossible for a distributed database management system to provide Consistency, Availability and Partition tolerance simultaneously. i.e., a distributed database management system cannot satisfy all three of these guarantees at the same time. ACID properties also cannot be maintained across partitions; it is required to restore partitions to ensure it.
C. Data stream management[1] : When data comes in free form from the various machines from the internet, it is very difficult to manage through fixed schema. For these types of streamed data, better data stream management is required.
D. Query Optimization[3] : It is very difficult to optimize the join queries which are distributed in nature.

IV. RELATED WORK AND FUTURE SCOPE

M. Ozsu and P. Valduriez state that there are algorithms[1] given for partitioning or fragmentation. In internet-based applications data comes in the form of audio, video, documents and other formats which are called stream data. Real time data comes in the form of stream (unbounded sequence) from internet. These data are distributed across many machines, which are accessed by many users from their own machines.

D. Wang has proposed Cluster-and-Conquer algorithm[4] for optimizing distributed join over database federation with efficiently considering run-time conditions. Cluster-and-Conquer algorithm is motivated from real world observation in which author has proposed to view the whole database federation as clustered

system, and provide each cluster of data sources with its cluster mediator. Finally author has implemented the prototype federation system with the proposed architecture and optimization algorithm. The experimental results showed the capabilities and efficiency of Cluster-and-Conquer algorithm and gave the target environment where the algorithm performs better than other related approaches. Currently the prototype system has two levels of mediators, but it is necessary to extend the system in order to support multi-level mediators whenever the environment demands. Another possible extension is to employ this algorithm to other distributed systems, such as distributed databases and grid computing systems. The philosophy of cluster-and-conquer is expected to be useful for large-scale distributed computing environments. So the algorithm can be extended for the processing of other types of operations, like aggregate (such as group-by, max and min), top-K, etc.

R. Taylor proposed a cost model[3] that allows inter-operator parallelism opportunities to be identified within query execution plans. This allows the **response time** of a query to be estimated more accurately. The author has merged two existing centralized optimization algorithms DPccp and IDP1 to create a practically more efficient algorithm IDP1ccp. He proposed the novel **Multilevel optimization algorithm** framework that **combines heuristics with existing centralized optimization algorithms.** The distributed multilevel optimization algorithm (DistML) proposed in this paper uses the idea of distributing the optimization phase across multiple optimization sites in order to fully utilize the available system resources. The future work on this cost model could be done to make it capable of handling pipelining between operators, which means that one operator feeds its output tuples directly into a parent operator when they become available without writing them to disk.

S. blanas has has evaluated the join methods on a 100-node system and shown the unique tradeoffs of these join algorithms[5] in the context of MapReduce. They have also explored how their join algorithms can benefit from certain types of practical preprocessing techniques. The valuable insights obtained from their study can help an optimizer select the appropriate algorithm
based on a few data and manageability characteristics.The proposed methods can be evaluated for multi-way joins, exploring indexing methods to speedup join queries, and designing an optimization module that can automatically select the appropriate join algorithms. Another important future direction is to design new programming models to extend the MapReduce
framework for more advanced analytic techniques.

For computations of cost from the optimization process, the optimizer must consult the data sources involved in an operation to find the cost of that operation. The mentioned analytical process in [6] indicate that, in many cases, especially when the physical database design is known to the optimizer, this query optimization algorithm works very well. But in absence of physical database design, more aggressive optimization techniques must be required.

In paper [7], through the research on query optimization technology, based on a number of optimization algorithms commonly used in distributed query, a new algorithm is designed, and experiments show that this algorithm can significantly reduce the amount of intermediate result data, effectively reduce the network communication cost, to improve the optimization efficiency. As a future work, the algorithm can be extended for distributed file system.

CONCLUSION AND RESEARCH DIRECTIONS

It is very difficult to find an ideal optimum solution for the distributed data. To obtain optimum solution, the cost of network, resources, response time, access time, memory

usage, processing time, etc. should be minimized which could be done with the use of better algorithms for different principles of DDBMS, materialized views, on-the fly schema, caching of frequently used data, etc.

REFERENCES

1. M. T. Ozsu, and P. Valduriez, "Book : Principles of Distributed Database Systems, Third Edition, Springer
2. S. Naik, "Book-Concepts of Database Management System", First Edition, Pearson
3. R. Taylor, "Thesis on Query Optimization for Distributed Database Systems", University of Oxford, 2010
4. D. Wang, "Thesis on Efficient Query Optimization for Distributed Join in Database Federation", Worcester Polytechnic Institute, March 2009
5. S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita and Y. Tian, "A Comparison of Join Algorithms for Log Processing in MapReduce", SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA.
6. D. Sukheja and U. Singh, "A Novel Approach of Query Optimization for Distributed Database Systems", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July, 2011
7. F. Yuanyuan and M. Xifeng, "Distributed Database System Query Optimization Algorithm Research", IEEE, 2010