International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

A PRAGMATIC REVIEW OF DISTRIBUTED DATABASE SYSTEMS AND RELATED ASPECTS

Hitu Kalra
M.Tech. Research Scholar
Shree Siddhivinayak Group of Institutions
Shahpur-Bilaspur, Distt. Yamuna Nagar, Haryana, India

Prof. Ajay Kumar
Shree Siddhivinayak Group of Institutions
Shahpur-Bilaspur, Distt. Yamuna Nagar, Haryana, India

ABSTRACT

A distributed system is a piece of software that ensures that a collection of independent computers that are interconnected by a computer network, appears to its users as a single coherent system and that cooperate in performing certain assigned tasks. As a general goal, distributed computing systems partition a big, unmanageable problem into smaller pieces and solve it efficiently in a coordinated manner. The economic viability of this approach stems for two reasons: (1) more computer power is harnessed to solve a complex task, and (2) each autonomous processing element can be managed independently and develop its own application. The word distributed referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing. This paper highlights various parameters and aspects of the distributed databases and related arguments.

Keywords – Distributed Databases, Query Optimization, Performance of Distributed Systems

Distributed Database System

A distributed database (DDB) consists of a collection of multiple logically interrelated databases distributed over multiple sites connected by some form of communication network. A distributed database management System (distributed DBMS) is a software system that manages a distributed database while making the distribution transparent to the user. The term distributed database system (DDBS) is typically used to refer to the combination of DDB and the

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

distributed DBMS. In this system each site is able to process local transactions; in addition a site may participate in the execution of global transactions: those transactions that access data in several sites. The execution of global transactions requires communication among sites. Although geographically dispersed, a distributed database system manages and controls the entire database as a single collection of data.

ANSISPARC (American National Standards Institute, Standards Planning And Requirements Committee) represented an abstract design standard for a DDBMS.

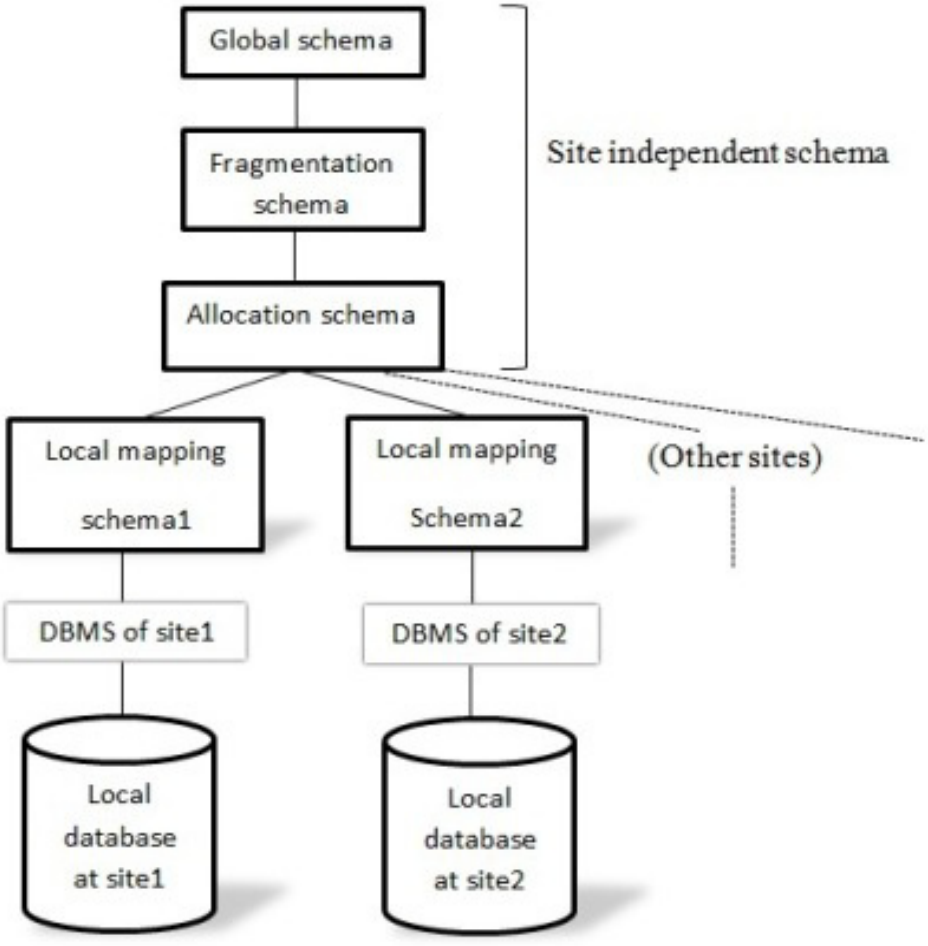

Figure 1 - Design Standard for Distributed Database System

The global level permits the integration of local databases in a global database by using the global scheme, the fragmentation scheme and the allocation scheme. The global scheme defines all the information contained in a network distributed database and is described by a set of global

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

relations. Every global relation can be split in many disjunctive parts called fragments.

The fragmentation scheme describes the connections between the global relation and its fragments. This is a one-to-many type of scheme and has the form of a hierarchy. The fragments are logical parts of the global relations that can be physically allocated on one or more nodes of the network.

The allocation scheme describes the distribution mode of the segments on the nodes of the network. Every segment will have a physical allocation on one or more nodes. The allocation scheme introduces a minimum controlled redundancy, so that a certain segment can be located on multiple nodes of the network.

The local level treats every local database like a centralized database. At this level, the local scheme, which depends on local DBMS, makes the correspondence between the physical global relations on that node and the objects manipulated by the local DBMS.

Distributed database may be homogeneous or heterogeneous database system.

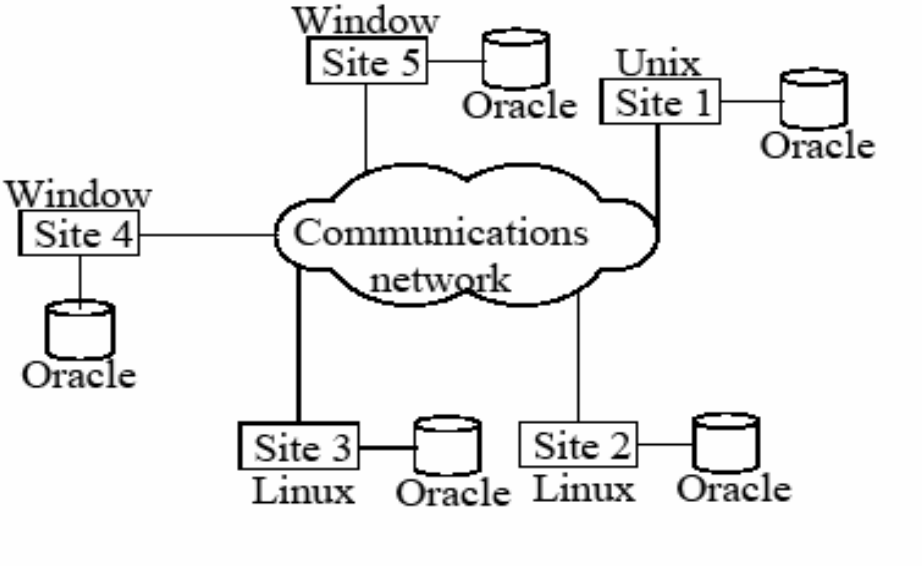

Figure 2 - Homogeneous Distributed Database System

In homogeneous distributed database system, all sites of the database system have identical setup, i.e., same database system software, common schema and database system code but the underlying operating system may be different

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

In heterogeneous distributed database system, each site may run under the control of different database system/software where the schemas and system code may differ.

Distributed Database System Architecture

Following three architectures are used in distributed database systems:

- Client/server architecture
- Collaborating server system
- Middleware system

Client/server Architecture

Client/server architectures are those in which a DBMS- related workload is split into two logical components namely client and server, each of which typically executes on different systems

Client is the user of the resource whereas the server is a provider of the resource. Client/Server architecture has one or more client processes and one or more server processes. Distributed database systems traditionally adopt a client-server architecture consisting of client and server processes. In a client-server architecture there exists a clean separation of responsibility between client processes and server processes. Data is stored at server sites that run server processes that provide access to their local data to client processes. Client sites run client processes that provide an interface between users and the data residing in the system.

Techniques for Distributed Database Design

Data Fragmentation

Technique of breaking up the database into logical units, which may be assigned for storage at the various sites, is called data fragmentation. In the data fragmentation, a relation can be partitioned (or fragmented) into several fragments for physical storage purposes.[6] These fragments contain sufficient information to allow reconstruction of the original relation. All fragments of a given relation will be independent. There are three different schemes for fragmenting a relation:

- Horizontal fragmentation
- Vertical fragmentation
- Mixed fragmentation

Horizontal fragmentation

A horizontal fragment of a relation is a subset of the tuples (rows) with all attributes in that relation. Horizontal fragmentation splits the relation "horizontally" by assigning each tuple or group (subset) of tuples of a relation to one or more fragments, where each tuple or a subset has a certain logic meaning i.e. tuples that belong to horizontal fragment are specified by a condition on one or more attributes.

Horizontal Fragmentation

It is defined using the select operation of the relational algebra. The original relation is obtained by UNION operation.

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

Vertical Fragmentation

Vertical fragmentation splits the relation by decomposing "vertically" by columns (attributes). A vertical fragment of a relation keeps only certain attributes of the relation at a particular site because each site may not need all the attribute of a relation. JOIN operation is done to reconstruct the original relation.

Mixed Fragmentation

Sometimes, horizontal or vertical fragmentation of a database schema by itself is insufficient to adequately distribute the data for some applications so, mixed or hybrid fragmentation is required. Thus, Horizontal (or vertical) fragmentation of a relation, followed by further vertical (or horizontal) fragmentation of some of the fragments, is called mixed fragmentation. A mixed fragmentation is defined using the select and project operations of the relational algebra.

Data Replication

Data replication is a technique that permits storage of certain data in more than one site. The system maintains several identical replicas (copies) of the relation and stores each replica at a different site. Typically, data replication is introduced to increase the availability of the system. When a copy is not available due to site failure(s), it should be possible to access another copy. The most extreme case is replication of the whole database at every site in the distributed system, thus creating a fully replicated distributed database. Full replication makes the concurrency control and recovery techniques more expensive than they would be if there was no replication. Like fragmentation, data replication should also support replication independence. The other extreme from full replication involves having no replication - that is, each fragment is stored at exactly one site. In this case, all fragments must be disjoint, except for the repetition of primary keys among vertical (or mixed) fragments. This is also called non redundant allocation.

Data Allocation

Each fragment – or each copy of a fragment – must be assigned to a particular site in the distributed system. This process is called data distribution (or data allocation). The choice of sites and the degree of replication depend on the performance and availability goals of the system and on the types and frequencies of transactions submitted at each site. For example, if high availability is required and transactions can be submitted at any site and if most transactions are retrieval only, a fully replicated database is a good choice. However, if certain transactions that access particular parts of the database are mostly submitted at a particular site, the corresponding set of fragments can be allocated at that site only. Data that is accessed at multiple sites can be replicated at those sites. If many updates are performed, it may be useful to limit replication. Finding an optimal or even a good solution to distributed data allocation is a complex optimization problem..

Advantages

- **Management of distributed data with different levels of transparency** like

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

Distribution or network transparency, fragmentation transparency, replication transparency. It makes the user unaware of operational details of the network, existence of copies of data and existence of fragments.

- **Availability and Reliability**: In distributed database systems, the availability of the data increases because the replicas of the data are distributed at different sites. It also increases the reliability because if the one site fails, then data can be accessed from the other site where its replica is present. So, in distributed environment, failure of one site does not result in unavailability of the data.
- **Localization**: means the data is present as close to the site where it is needed, therefore data can be accessed in less time and data transfer time also reduces.
- **Improved Performance**: Because the data is stored on multiple sites, so the overhead on one machine decreases which improves the performance.
- **Easier expansion:** In a distributed environment, expansion of the system in terms of adding more data, increasing database sizes, or adding more processors is much easier.
- **Local autonomy or site autonomy**: A department can control the data about them (as they are the ones familiar with it.)
- **Protection of valuable data**: If there were ever a catastrophic event such as a fire, all of the data would not be in one place, but distributed in multiple locations.
- **Economics**: It costs less to create a network of smaller computers with the power of a single large computer.
- **Modularity**: systems can be modified, added and removed from the distributed database without affecting other modules (systems).
- **Reliable transactions**: Due to replication of database.

Distributed query processing and optimization

Query processing is an important concern in the field of distributed databases. The main problem is: if a query can be decomposed into sub queries that require operations at geographically separated databases, determine the sequence and the sites for performing this set of operations such that the operating cost (communication cost and processing cost) for processing this query is minimized. The problem is complicated by the fact that query processing not only depends on the operations of the query, but also on the parameter values associated with the query. Distributed query processing is an important factor in the overall performance of a distributed database system. Therefore, it is important here to optimize on the time required to access such a query, which will be largely comprised of the time spent in transmitting data between sites rather and not the time spent on retrieval from the disk storage or computation.

P. Valduriez has described a methodology of distributed query processing. The input is a query on distributed data expressed in relational calculus.

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

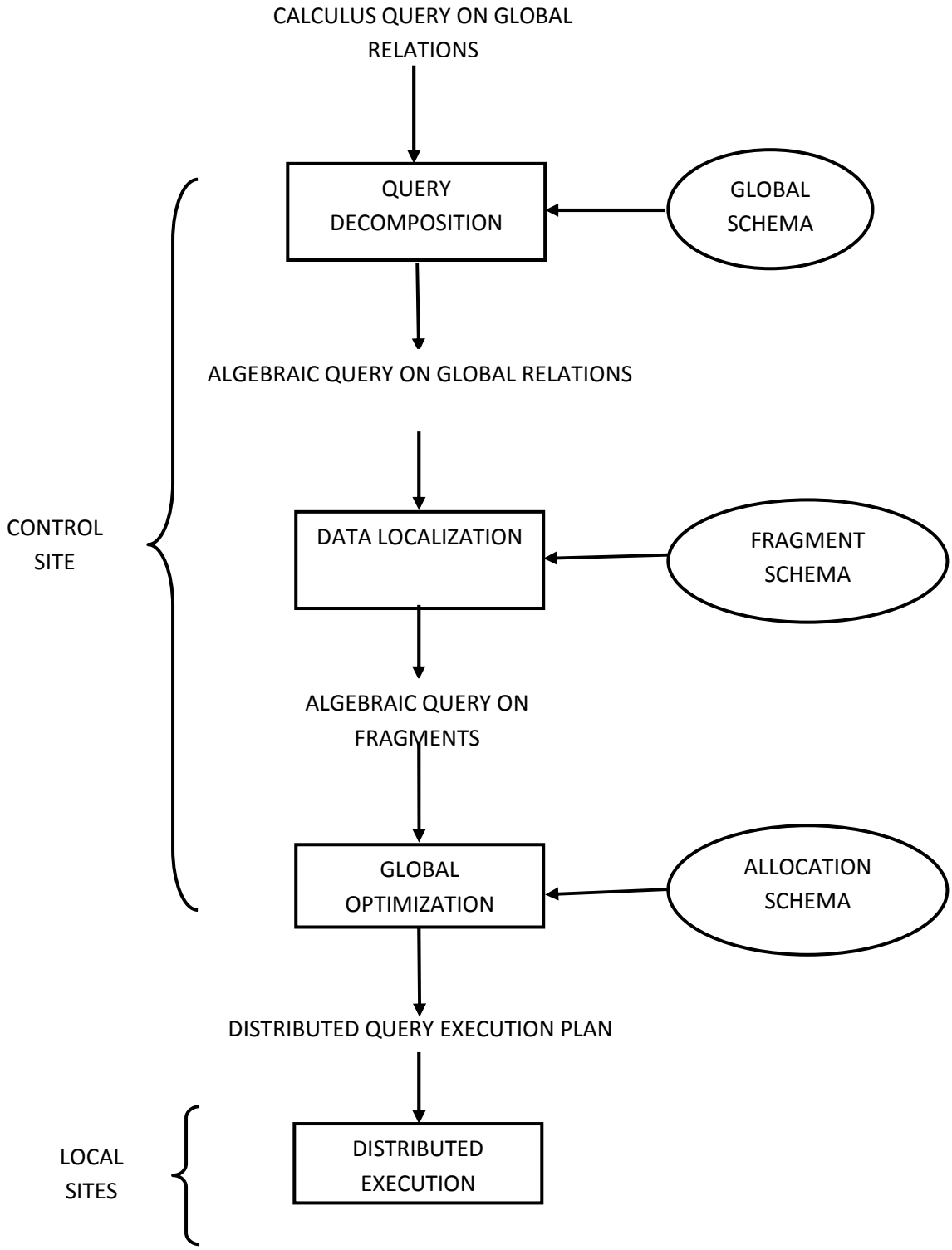

Figure 3 - Distributed Query Processing

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

Four main layers are involved in distributed query processing. The first three layers map the input query into an optimized sequence of distributed query execution plan They perform the functions of query decomposition, data localization, and global query optimization. Query decomposition and data localization correspond to query rewriting.

The first three layers are performed by a central control site and use schema information stored in the global directory. The fourth layer performs distributed query execution by executing the plan and returns the answer to the query. It is done by the local sites.

The general scheme for the execution of a distributed query is as follow:

- The execution of a request begins with the global request represented using the relational computations. This request is addressed to some global relations, data distribution being invisible at this level.
- This request is decomposed in sub requests expressed with relational algebra operators. These sub requests change from global sub requests in requests addressed to the global relations fragments, using information about the fragmentation scheme.
- The global optimization tries to find the best order of operations within sub requests addressed to the fragments, including the communication operations that minimize cost function.
- Local optimization of every sub request at a certain node is reviewed using information from local scheme.

These four phases realize decomposition of the global request into a sequence of local optimized operations, each of them acting on a local database.

LITERATURE REVIEW

Distributed Databases are becoming very popular now a day. Today's business environment has an increasing need for distributed database and Client/server applications as the desire for reliable, scalable and accessible information is Steadily rising. Distributed database systems provide an improvement on communication and data processing due to its data distribution throughout different network sites. Not Only is data access faster, but a single-point of failure is less likely to occur, and it provides local control of data for users, as suggested by Swati Gupta et al.

Query optimization is an important part of database management system. Query optimization has been one of the research focuses of database area, although many researchers have done a lot of work, but the not commensurate with the successful application of relational database technology in data processing is multi-join query optimization has been a problem not well resolved in relational database systems, as suggested by Fan Yuanyuan et al. .

E-business sites are increasingly utilizing dynamic web pages. Dynamic page generation technologies allow a Web site to generate pages at run-time. But very little work has been done so far to address the dynamic page generation delays. Proposed approach is based on concept of caching entire pages of dynamically generated content. The

novelty of our approach lies not only in the caching of dynamic page fragments, but also in the utilization of intelligent cache management strategies as suggested by Neera Batra et al.

Reza Ghaemi, Amin Milani Fard, et al. suggested that due to new distributed database applications such as huge deductive database systems, the search complexity is constantly increasing and we need better algorithms to speedup traditional relational database queries. An optimal dynamic programming method for such high dimensional queries has the big disadvantage of its exponential order and thus we are interested in semi-optimal but faster approaches.

Task allocation in Distributed computing systems (DCS) is an important research problem. When resource to be shared in DCS is a database that system is classified as Distributed database system (DDBS). In DDBS systems Data & operation allocation are both closely interrelated & highly dependent on each other. General models and objective function can be treated as basic platform for research in this area of task allocation An objective function can be derived by modifying the terms present in general model ,which in turn depend on characteristics of the system concerned ex. Distributed computing system, distributed database system, parallel system & multiprocessors etc. In DCS the software application is called a task and is a set of cooperating modules. For achieving a fast response time from such systems, an efficient assignment of the application tasks to the processors is imperative as suggested by Suchita Upadhyaya et al.

A distributed system database performance is strongly related to the fragment allocation in the nodes of the network. A heuristic algorithm for redistributing the fragments is proposed by Ţambulea L., et al.. The algorithm uses the statistical information relative to the requests send to a distributed database. This algorithm minimizes the size of the data transferred for solving a request. Assuming that a distribution of the fragments in the nodes of a network is known, the algorithm generates a plan to transfer data fragments, plan that will be used to evaluate a request.

Semi-join reducers were introduced in the late seventies as a means to reduce the communication costs of distributed database systems. Subsequent work done by Stocker, Kossman et al in the eighties showed, however, that semi-join reducers are rarely beneficial for the distributed systems of that time. They showed that semi-join reducers can indeed be beneficial in modern client-server or middleware systems -- either to reduce communication costs or to better exploit all the resources of a system. Furthermore, they presented and evaluated alternative ways to extend state-of-the-art (dynamic programming) query optimizers in order to generate good query plans with semi-join reducers. They presented two variants, called Access Root and Join Root, which differ in their implementation complexity, their running times, and the quality of plans they produce. They presented the results of performance experiments that compare both variants with a traditional query optimizer.

International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 4 Issue 3 May 2014
International Manuscript ID : 2249054XV4I3052014-30

A multi database model of distributed information retrieval is proposed by J. Callan et al in which people are assumed to have access to many searchable text databases. In such an environment, full text information retrieval consists of discovering databases contents, ranking databases by their expected ability to satisfy the query, searching a small number of databases and merging results returned by different databases.

Distributed data processing is becoming a reality. Businesses want to do it for many reasons, and they often must do it in order to stay competitive. While much of the infrastructure for distributed data processing is already there (e.g., modern network technology), a number of issues make distributed data processing still a complex undertaking, as suggested by Donald Kossman, et al.[9]: (1) distributed systems can become very large, involving thousands of heterogeneous sites including PCs and mainframe server machines; (2) the state of a distributed system changes rapidly because the load of sites varies over time and new sites are added to the system; (3) legacy systems need to be integrated—such legacy systems usually have not been designed for distributed data processing and now need to interact with other (modern) systems in a distributed environment. He presented the state of the art of query processing for distributed database and information systems. He presented the "textbook" architecture for distributed query processing and a series of techniques that are particularly useful for distributed database systems. These techniques include special join techniques, techniques to exploit intra query parallel, techniques to reduce communication costs, and techniques to exploit caching and replication of data. Furthermore, the paper discusses different kinds of distributed systems such as client-server, middleware (multi tier), and heterogeneous database systems, and shows how query processing works in these systems.

P. Griffiths Selinger, et al. claimed that high level query and data manipulation languages such as SQL, requests are stated non-procedurally, without reference to access paths. They described how System R chooses access paths for both simple (single relation) and complex queries (such as joins), given a user specification of desired data as a Boolean expression of predicates. System R is an experimental database management system developed to carry out research on the relational model of data. System R was designed and built by members of the IBM San Jose Research Laboratory.

Yannis E. loannidis et al. described query optimization for relational database systems as a combinatorial optimization problem, which makes exhaustive search unacceptable as the query size grows. Randomized algorithms, such as Simulated Annealing (SA) and Iterative Improvement (II), are viable alternatives to exhaustive search. They adapted these algorithms to the optimization of project-select-join queries. They tested them on large queries of various types with different databases, concluding that in most cases SA identifies a lower cost access plan than II. To explain this result, they studied the shape of the cost function over the solution space associated

with such queries and we have conjectured that it resembles a 'cup' with relatively small variations at the bottom. This has inspired a new Two Phase Optimization algorithm, which is a combination of Simulated Annealing and Iterative Improvement. Experimental results show that Two Phase Optimization outperforms the original algorithms in terms of both output quality and running time.

Chin-Wan Chung, et al. addressed the processing of a query in distributed database systems using a sequence of semi joins. The objective was to minimize the intersite data traffic incurred by a distributed query. He developed a method which accurately and efficiently estimates the size of an intermediate result of a query. This method provided the basis of the query optimization algorithm. He presented a heuristic algorithm to determine a low-cost sequence of semi joins. Furthermore he provided the cost comparison with an existing algorithm. He measured the scheduling time for sequences of semi joins for example queries using the PASCAL program which implements the algorithm.

Philip A Bernstein, et al. described the techniques used to optimize relational queries in the SDD-1 distributed database system. They translated each Data language query into a relational calculus form called an envelope and concerned with the optimization of envelopes. They proposed envelopes in two phases. The first phase executes relational operations at various sites of the distributed database in order to delimit a subset of the database that contains all data relevant to the envelope. This subset is called a reduction of the database. The second phase transmits the reduction to one designated site, and the query is executed locally at that site. They presented an algorithm that constructs a cost-effective program of semi joins, given an envelope and a database.

CONCLUSION

The optimization of queries in distributed systems is a complex activity that depends on many factors. In a certain percent it is performed by the DBMS, but there are situations when the user applications must contain algorithms for the query optimization. I studied many related work and found out that very few work addressed the problem of considering run-time conditions in query optimization. By analyzing both theoretically and experimentally, I have presented the need to take run-time conditions, including CPU utilities in the data sources and network environment, into account in optimization process.

This work studies underlines the distributed database systems and related aspects. In the future scope, this work shall implement the prototype system with the proposed architecture and optimization algorithm. The experimental results shall show the capabilities and efficiency of join query optimization algorithm and give the target environment where the algorithm performs better than other related approaches and predict the best plan for a join query.

REFERENCES

[1] Swati Gupta, Kuntal Saroba, Bhawna, “Fundamental Research of Distributed Database”, International Journal of

Computer Science and Management Studies, pp. 138-146, 2011

[2] Fan Yuanyuan, Mi Xifeng, "Distributed Database System Query Optimization Algorithm Research", IEEE international conference on Computer Science and Information Technology (ICECT), pp.145-149, 2011

[3] Neera Batra, A. K. Kapil, "Three Tier Cache Based Query Optimization Model in Distributed Database", IJEST, vol. 2, 2010, pp. 3206-3212

[4] Reza Ghaemi, Amin Milani Fard, Hamid Tabatabee, Mahdi Sadaghizadeh, "Evolutionary Query optimization for Heterogeneous Distributed Database Systems" ,World Academy of Science, Engineering and Technology, pp. 43-49, 2008

[5] S. Upadhyaya and S. Lata, "Task Allocation in Distributed Computing VS Distributed Database Systems: A Comparative Study", IJCNS (International Journal of Computer Science and Network Security),vol. 8:3, pp. 338-346, 2008.

[6] Ţambulea L., Horvat-Petrescu M., "Redistributing Fragments into a Distributed Database, International Journal of Computers Communications & Control", ISSN 1841-9836, 3(4):384-394, 2008.

[7] Stocker, Kossman, Braumandl, Kemper, "Integrating Semi Join Reducers into state of the art query processors", ICDE, pp. 143-156 , 2001

[8] J. Callan, "Distributed Information Retrieval " , W. B. Croft, Ed. Kluwer Academic Publishers, pp. 127-150, 2000

[9] D. Kossman, "The state of the art in distributed query processing" , ACM Computing Surveys, pp. 422-469, 1998

[10] P. Griffiths, Selinger, M. M. Astrahan, D. D. Chamberlin, R.A. Lorie, T. G. Price, "Access path selection in a rational database management system", Morgan Kauffman series in Data Management Systems, pp. 141-152, 1998

[11] Yannis. E. Loannidis and Youngkyung Cha Kang, "Randomized Algorithms for optimizing large Join Queries", ACM Computing Surveys, pp. 47-53, 1990

[12] Chin-Wan Chung, "An Optimization of Queries in Distributed Database Systems", Journal of Parallel and Distributed Computing 3, pp. 137-157, 1986

[13] Philip A Bernstein and Nathan Goodman, Engene Wong, Christopher L. Reeve and James B. Rothnie, "Query Processing in a System for Distributed Databases(SDD – 1), ACM Transactions on Database Systems, vol. 6, no. 4, 1981, pp. 602-625

[14] Li, Vector O. K. , "Query Processing in distributed databases" , MIT. Lab. For Information and Decision Systems, pp. 1107, 1981

[15] Huang, Kuan – Tsae, Davenport, Wilbur B., "Query Processing in Distributed Heterogeneous Systems", MIT Laboratory for information and Decision Systems, pp. 45-49 , 1981

[16] B.M. Monjurul Alom, Frans Henskens and Michael Hannaford, "Query Processing and Optimization in Distributed Database Systems", IJCSNS International Journal of Computer Science and Network Security, vol.9 no.9, 2009, pp. 143-152

[17] Syam Menon, "Allocating fragments in distributed Database", IEEE Transactions on Parallel and Distributed Systems, pp. 577-585, 2005

[18] D. Kossmann, K. Stocker, "Iterative Dynamic Programming: A New Class of Query optimization Algorithms", ACM Computing Surveys, pp. 422-469, 2000

[19] Lee Chiang, Chi-sheng shin, Yaw-huei chen, "Optimizing large join queries using a graph based approach", IEEE Transactions on Knowledge and Data Engineering, pp. 441-450, 2006

[20] Tsai, P.S.M, Chen A.L.P, "Optimizing queries with foreign function in a distributed environment", IEEE Transactions on Knowledge and Data Engineering, pp.809-824, 2002