Non-replicated Dynamic Data Allocation in Distributed Database Systems

Arjan Singh† and K.S. Kahlon††

†Department of Computer Science & Engineering, BBSB Engineering College, Fatehgarh Sahib, Punjab, India
††Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, Punjab, India

Summary
Allocation of data or fragments in distributed database is a critical design issue and requires the most effort. It has the greater impact on the quality of the final solution and hence the operational efficiency of the system. Performance of the distributed database system is heavily dependent on allocation of data among the different sites over the network. The static allocation provides only the limited response to the change in workload. So, choosing an appropriate technique for allocation in the distributed database system is an important design issue. In this paper, a new dynamic data allocation algorithm for non-replicated distributed database system has been proposed. The proposed algorithm reallocates data with respect to the changing data access patterns with time constraint. This algorithm will decrease the movement of data over the network and also improve the overall performance of the system.

Key words:
Distributed Databases, Static Data Allocation, Dynamic Data Allocation

1. Introduction

Due to demand for system availability, autonomy, and enabled by advances in database and communication technology, distributed database systems are becoming wide-spread [8]. Distributed database technology is one of the most important developments of the past two decades. The maturation of database management systems technology has coincided with significant developments in distributed computing. It resulted in the emergence of Distributed Database Management Systems (DDBMS) [23].Distributed database is a collection of multiple, logically interrelated database distributed over computer network and distributed database management system is a database management system capable of supporting and manipulating distributed database [23]. The basic motivations for distributed databases are to improve system performance, to increase the availability of data, shareability, expandability and access facility. Distributed database systems are used in applications requiring access to an integrated database from geographically dispersed locations.

The design of distributed databases is an optimization problem requiring solutions to following two problems [7]:

- Designing the fragmentation of global relations
- Designing the allocation of fragments to the sites of communication network

The problem of fragmenting the database is difficult one in itself and variety of approaches exist for fragmenting the database. But, this study concentrates only on data (fragments) allocation problem, assuming that the database is already fragmented.

The problem of allocating data in a distributed database system has an important impact upon the performance and reliability of the system as a whole [2,10]. The aim is to store the fragments closer to where they are more frequently used in order to achieve best performance. So, one key principle in distribution design is to achieve maximum locality of data and applications. Since, distributed databases enable more sophisticated communication between sites; the major motivation for developing a distributed database is to reduce communication by allocating data as close as possible to the applications which use them [8]. Thus in a well-designed distributed database 90 percent of the data should be found at the local site, and only 10 percent of the data should be accessed on a remote site [8]. A poorly designed data allocation can lead to inefficient computation, high access cost and high network loads [23].

Various approaches have already evolved for allocation of data in distributed database. In most of these approaches, data allocation has been proposed prior to the design of a database depending on some static data access patterns and/or static query patterns. In a static environment, where the access probabilities of nodes to fragments never change, a static allocation of fragments provides the best solution. However, in a dynamic environment where these probabilities change over time, the static allocation solution would degrade the database performance.

Fragment allocation can further be divided into two different categories: redundant or non-redundant [7, 23]. In a non-redundant allocation exactly one copy of each fragment will exist across all the sites, while under a redundant allocation, fragments are replicated over multiple sites. In this paper a new dynamic data allocation

Manuscript received September 5, 2009
Manuscript revised September 20, 2009

algorithm for non-replicated database systems have been introduced which is an extension of [4] and [29] .The objective of this work is to design an effective algorithm that can generate minimum total data transfer cost allocation schemes in changing load environments. The rest of the paper is organized as follows.

Section 2 provides an overview of the related work done so far. Section 3 provides the proposed new algorithm for non-replicated dynamic allocation of data. Section 4 describes the comparison of proposed new algorithm with other dynamic approaches. Finally, Section 5 summarizes the contribution of the study and point out direction for further research.

2. Related Work

Many reports have been published on the problem of allocation of data to nodes. The file allocation problem was first investigated by [10]. [10] developed a global optimization model to minimize overall operating costs under the constraints of response time and storage capacity with fixed number of copies each file. [5] relaxed the assumption of fixed number of copies and stressed the difference between updates and retrieval. [14] proved that [5]'s formulation was NP-Complete and suggested heuristic rather than deterministic approaches be investigated. [26] analyzed a file allocation problem in the environment of a distributed database for optimization of query processing. By introducing replicated file, [27] showed how communication cost attributed to joins can be minimized. [6] considered the problem of file allocation for typical distributed database applications with a simple model of transaction execution.

[3,27] have observed that the fragment allocation problem differs from the well-studied file allocation problem. [3] considered the allocation of the distributed database to the sites so as to minimize total data transfer cost and devised a comprehensive approach to allocate relations. [27] provides a nonlinear integer programming formulation and its linearization. [25] incorporated issues like concurrency and queuing costs, while [18] presents a max-flow approach. [28] provides an integrated approach for fragmentation and allocation. [28] identified seven criteria that a system designer can use to determine the fragmentation, replication and allocation. [9] provides approach for allocating fragments by adapting a machine learning approach. [24] incorporates a concurrency mechanism, and [31] present a replication algorithm that adaptively adjusts to changes in read-write patterns.

[12] provides an approach based on Lagrangian relaxation and [16] describes heuristic approaches. Beside allocating data, [13] and [20] present a mathematical modeling approach and a genetic algorithm-based approach to allocate operations to nodes. [22] has presented an integer programming formulation for the non-redundant version of the fragment allocation problem. More recently [15] has given a high-performance computing method for data allocation in distributed database system. The problem of distributing fragments of virtual XML repositories over the Web is considered by [1]. [32] has considered the related problem of distributing the documents of a Web site among the server nodes of a geographically distributed Web server.

In most of the above approaches, data allocation has been proposed prior to the design of a database depending on some static data access patterns and/or static query patterns. Static allocation of fragments provides the best solution where the access probabilities of nodes to fragments never change. But, the static allocation solution would degrade the database performance in a dynamic environment, where these probability change over time.

Over past few years, work has been introduced for dynamic data allocation in database systems. [31] give a model for dynamic data allocation for data redistribution. In [4] an algorithm is proposed for dynamic data allocation algorithm, which reallocates data with respect to bringing changing data access pattern. [9] presents an approach based on machine learning, [11] considers incremental allocation and reallocation based on changes in workload. [19] has given a dynamic object allocation and replication algorithm with centralized control. [21] incorporated security considerations into the dynamic file allocation process. In [17] an optimal algorithm for non-replicated database systems is proposed. [29] has introduced a threshold algorithm for non-replicated distributed databases. In the threshold algorithm, the fragments are continuously reallocated according to the changing data access patterns. In this paper, a new dynamic data allocation algorithm for non-replicated distributed database system has been proposed which is an extension of work carried out by [4] and [29, 30]. The proposed algorithm reallocates data with respect to the changing data access patterns with time constraint.

3. NEW NON-REPLICATED DYNAMIC ALLOCATION OF DATA

A major cost in executing queries in a distributed database system is the data transfer cost incurred in transferring fragments accessed by a query from different sites to the site where the query is initiated [7, 23]. The main objective of a data allocation algorithm is to allocate fragments at different sites in such a way that the total data transfer cost during the execution of a query can be minimized. The proposed work is an attempt to decrease the data transfer during the execution of a query.

The proposed algorithm for dynamic allocation of data in distributed database system is a variation of existing two approaches: Optimal algorithm [4] and Threshold algorithm [29].

In optimal algorithm [4], initially all the fragments are distributed over the different sites using any static data allocation method. After the initial allocation, optimal algorithm maintains access counters matrix for each locally stored fragment at each node or site. Every time an access request is made for the stored fragment then the access counter of the accessing node for the stored fragment is increases by one. If the accessing node is the current owner then there is no problem, otherwise if the counter of a remote node is greater than the counter of the current owner than move the fragment to the accessing node. So, a node having the highest access value for a particular fragment is the primary candidate for the fragment. The problem of this technique is that if the changing frequency of access pattern for each fragment is high, then it will spend more time for transferring fragments to different sites.

Threshold algorithm [29] has solved the problem of optimal algorithm. In threshold algorithm only one counter per fragment is maintained. Threshold algorithm guarantees the stay of the fragment for at least (t+1) accesses at the new node after a migration, where t is the value of threshold. The most important point in this algorithm is the choice of threshold value. If the threshold value increases then the migration of fragment will be less. But, if the threshold value decreases then there will be more migration of fragments. But threshold algorithm has following problem with its approach:

- Every time a node is going for a local access, it reset the counter of local fragment to zero.
- Whenever the counter exceeds the threshold value, the ownership of the fragment is transferred to another node. But, it does not specify which node will be the fragment's new owner.
- It does not give the information about past accesses of the fragments.
- It is not considering the time variable of the access pattern.

The new algorithm will remove all the above problems of threshold algorithm. It will reallocate data with respect to the changing data access patterns with time constraint. The new algorithm will add time constraint to the existing threshold technique. While migrating fragment from one node to another node, this algorithm will not only consider the threshold value but will also consider the time of access made to a particular fragment. This algorithm is called Threshold and Time Constraint Algorithm (TTCA). TTCA is illustrated as follow:

TTCA Algorithm:- Initially all the fragments are distributed over different nodes using any static allocation method in non-redundant manner. TTCA maintain a n×m counter matrix M, where n denotes the total number of fragments and m denotes the total number of nodes or sites. M_{ij} is the number of accesses to fragment i by node j.

Step 1: For each fragment, initialize the counter values equal to zero (i.e. set $M_{ij} = 0$, where i = 1,2,----,n and j = 1,2,----,m)

Step 2: Process an access request for the stored fragment.

Step 3: Increase the corresponding access counter of the accessing node by one for the stored fragment and also store the time of corresponding access.

Step 4: If the accessing node is the current owner, go to Step 2. (i.e. Local access, otherwise it is remote access).

Step 5: If the counter of the remote node is greater than the threshold value (t) and all last "t+1" accesses are made in a specified time (T) then reset corresponding fragment's counter to zero for all the node and transfer the fragment to the node who's counter value was greater than threshold value (t).

Step 6: Go to step 2.

TTCA will further decrease the migration of fragments over different sites as compare to simple threshold algorithm. It will migrate only that fragment which is most recently accessed for t numbers of times, where t is the threshold value. So if we increase the value of time variable then the migration of fragment will be more. But, if the value of time variable decreases then there will be less migration of fragments.

4. COMPARISON

Comparison of TTCA with the optimal and threshold has been made on the following three different parameters:

- Migration Condition
- Network overhead
- Storage Cost

Optimal algorithm migrate the fragment when the counter for remote node in the access matrix is greater than the counter of the owning node. Threshold algorithm migrate the fragment when the counter with particular fragment is greater than the threshold value. TTCA migrate fragment when the counter of the remote node is greater than the threshold value (t) and all last "t+1" accesses are made in a specified time (T).

Optimal algorithm increases the traffic on network when changing frequency of access pattern for

each fragment is high. Threshold algorithm decreases the overhead on the network by limiting the migration of fragment. TTCA further decreases the network overhead as compare to both optimal and threshold algorithms by including both the number of access and most recent access conditions for migration of fragments.

Optimal algorithm uses extra storage cost for access counter matrix. Threshold algorithm required less storage cost as compare to optimal algorithm, because it stores only one counter for each fragment. But TTCA requires more storage as compare to both optimal and threshold algorithms, as it stores access counter matrix and respective time of particular access.

5. CONCLUSION

Distributed databases are being increasingly used in various organizations, supported by availability of various distributed database management system software products. The decision on how to distribute organizational database using distributed database management system is an important issue, affecting both cost and performance. Performance of distributed database system is heavily dependent on allocation of data among different sites, because major cost in executing queries in a distributed database system is the data transfer cost incurred in moving data accessed by a query from different sites to site where the query is initiated. The static allocation provides only limited response to changing workload. This paper provides a dynamic algorithm for non-replicated distributed database systems. Purposed algorithm will decrease the movement of fragment during the reallocation process as compare to optimal and threshold algorithms. This work is a significant effort to minimize the amount of data transferred during processing the application.

References

[1] S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu, and T. Milo, "Dynamic XML Documents with Distribution and Replication," Proc. 2003 ACM SIGMOD Int'l Conf. Management of Data, pp. 527-538, 2003.

[2] S. Agrawal, V. Narasayya, and B. Yang, "Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design," Proc. 2004 ACM SIGMOD International Conf. Management of Data, pp. 359-370, 2004.

[3] P. Apers, "Data Allocation in Distributed Databases," ACM Trans. Database Systems, vol. 13, no. 3, pp. 263-304, Sept. 1988.

[4] A. Brunstroml, S.T. Leutenegger and R. Simhal, "Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database with changing Workload", ACM Trans. Database Systems, 1995.

[5] R. G. Casey, " Allocation of Copies of a File in an Information Network", in Proc. AFIPC 1972 SJCC, Vol 40, 1972, pp. 617-625.

[6] S. Ceri, G. Martella, and G. Pelagatti, "Optimal file Allocation for a Distributed Database on a Network of Minicomputers", in Proc. International Conference on Database, Aberdeen, July 1980, British Computer Society Hayden.

[7] S. Ceri and G. Pelagatti, "Distributed Databases: Principles & Systems", McGraw-Hill International Editions.

[8] S. Ceri, B. Pernici and G. Wiederhold, "Distributed Database Methodologies", Proceedings of IEEE, Vol. 75, No. 7, May 1987.

[9] A. Chaturvedi, A. Choubey, and J. Roan, "Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach," IEEE Trans. Eng. Management, vol. 41, no. 2, pp. 194-207, 1994.

[10] W.W. Chu, "Optimal File Allocation in Multiple Computer Systems" IEEE Transaction on Computers, Vol. C-18, No.10, 1969.

[11] A. Chin, "Incremental Data Allocation and Reallocation in Distributed Database Systems," Journal of Database Management, Vol. 12, No. 1, pp. 35-45, 2001.

[12] G. Chiu and C. Raghavendra, "A Model for Optimal Database Allocation in Distributed Computing Systems," Proc. IEEE INFOCOM 1990, vol. 3, pp. 827-833, June 1990.

[13] A. Corcoran and J. Hale, "A Genetic Algorithm for Fragment Allocation in a Distributed Database System," Proc. 1994 ACM Symp. Applied Computing, pp. 247-250, 1994.

[14] K.P. Eswaran, "Placement of Records in a File and File Allocation in a Computer Network", on Proc. IFIP Congr. North-Holland, 1974.

[15] I.O. Hababeh, M. Ramachandran and N. Bowring, " A high-performance computing method for data allocation in distributed database systems", Springer, J Supercomput (2007) 39:3-18.

[16] Y. Huang and J. Chen, "Fragment Allocation in Distributed Database Design," J. Information Science and Eng., vol. 17, pp. 491- 506, 2001.

[17] L.S. John, "A Generic Algorithm for Fragment Allocation in Distributed Database System", ACM 1994.

[18] K. Karlaplem and N. Pun, "Query-Driven Data Allocation Algorithms for Distributed Database Systems," Proc. Eighth International Conf. Database and Expert Systems Applications (DEXA '97), pp. 347- 356, Sept. 1997.

[19] W.J. Lin and B. Veeravalli, "A Dynamic Object Allocation and Replication Algorithm for Distributed System with Centralized Control," International Journal of Computer and Application, Vol. 28, no. 1, pp. 26-34, 2006.

[20] S. March and S. Rho, "Allocating Data and Operations to Nodes in Distributed Database

Design," IEEE Trans. Knowledge and Data Eng., vol. 7, no. 2, pp. 305-317, 1995.
[21] A. Mei, L. Mancini, and S. Jajodia, "Secure Dynamic Fragment and Replica Allocation in Large-Scale Distributed File Systems," IEEE Trans. Parallel and Distributed Systems, vol. 14, no. 9, pp. 885-896, Sept. 2003.
[22] S. Menon, "Allocating Fragments in Distributed Databases", IEEE Transactions on Parallel and Distributed Systems, Vol. 16, No. 7, July 2005.
[23] M. Ozsu and P. Valduriez, "Principles of Distributed Database Systems", Prentice Hall, second ed. 1999.
[24] S. Ram and R. Marsten, "A Model for Database Allocation Incorporating a Concurrency Control Mechanism," IEEE Trans. Knowledge and Data Eng., vol. 3, no. 3, pp. 389-395, 1991.
[25] S. Ram and S. Narasimhan, "Database Allocation in a Distributed Environment: Incorporating a Concurrency Control Mechanism and Queuing Costs," Management Science, vol. 40, no. 8, pp. 969-983, 1994.
[26] C.V. Ramamoorthy and B.W. Wah, "The Placement of Relations on a Distributed Relational Database", in Proc. 1st Conf. On Distributed Computing System 1979.
[27] R. Sarathy, B. Shetty, and A. Sen, "A Constrained Nonlinear 0-1 Program for Data Allocation," European J. Operational Research, vol. 102, pp. 626-647, 1997.
[28] A. Tamhankar and S. Ram, "Database Fragmentation and Allocation: An Integrated Methodology and Case Study," IEEE Trans. Systems, Man and Cybernetics—Part A, vol. 28, no. 3, May 1998.
[29] T. Ulus and M. Uysal, "Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems", Pakistan Journal of Information and Technology, 2(3): pp. 231-239, 2003.
[30] T. Ulus and M. Uysal, "A Threshold Based Dynamic Data Allocation Algorithm- A Markove Chain Model Approach", Journal of Applied Science 7(2), pp 165-174, 2007.
[31] O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm," ACM Trans. Database Systems, vol. 22, no. 2, pp. 255-314, 1997.
[32] L. Zhuo, C. Wang, and F. Lau, "Document Replication and Distribution in Extensible Geographically Distributed Web Server," J. Parallel and Distributed Computing, vol. 63, no. 10, pp. 927-944, 2003.