Non-Redundant Dynamic Data Allocation in Distributed Database Systems

Raju Kumar
Department of Computer Applications,
Krishna Institute of Engineering & Technology,
Ghaziabad, India

Neena Gupta
Department of Computer Science,
Kanya Gurukul Mahavidyalaya, Dehradun
(Second Campus of Gurukul Kangri
Vishwavidyalaya, Haridwar), India.

ABSTRACT

In the past few decades, the significant developments in database and networking technologies contributed to advances in distributed database systems (DDS). The data allocation is a prominent issue in distributed database systems and is performed on data access static and dynamic patterns. This paper proposes a new strategy named Extended Threshold Algorithm (ETA) for non-redundant dynamic data allocation in distributed database. The proposed algorithm is an extension of Threshold and Time Constraint Algorithm (TTCA) which was based on Optimal and Threshold algorithms. ETA performs relocation of data fragments with respect to changing access patterns to data fragments. It also reduces the space requirement and significantly improves the system performance.

Keywords

Distributed Database System, Static Data Allocation, Dynamic Data Allocation, Non-redundant Database, Redundant Database.

1. INTRODUCTION

The advances in database and communication technologies enhanced the popularity of distributed databases, as it provides high availability, autonomy, and affordability for managing large databases [1]. A distributed database can be considered as a collection of data which are distributed over different sites of a computer network. Each site of the network is capable to perform local applications autonomously. Each site also must participate in the execution of at least one global application, by accessing data at several sites using a communication subsystem [2]. Distributed database systems are used in applications which require access to an integrated database from geographically dispersed locations. The location of data items and the degree of autonomy of individual sites play a prominent role in all aspects of the system.

Data allocation describes the process of deciding where to locate the data. The task of allocating data in a distributed database system is a prominent activity, as it has a critical impact upon the reliability and performance of the system as a whole [4], [5]. The main motivation for developing a distributed database is to decrease the cost of communication by allocating data as close as possible to the applications which use them [1]. Thus in a well-designed distributed database only 10 percent of the overall data should be accessed from remote sites, and the remaining 90 percent of the data should be stored at the local sites [1]. A data allocation which is poorly designed can result to high network loads, and high access cost [6]. Therefore selecting an efficient data allocation method is desirable.

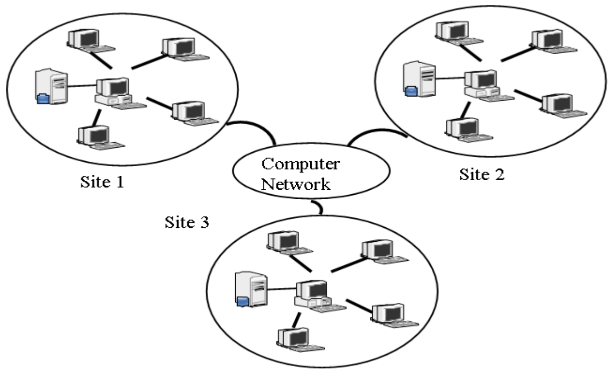

Fig 1: Distributed Database System [3]

A variety of data allocation approaches in distributed databases have already evolved. In most of these approaches, data allocation has been proposed before the design of a database on the basis of some static query/data access patterns. In an environment, where the access probabilities of nodes to fragments never change static data allocation techniques provide the best solution. However, in a dynamic environment where access probabilities of nodes to fragments change over time, the dynamic data allocation techniques provide better solution.

Further more, data allocation can be divided into two different categories: redundant and non-redundant [2],[6],[9]. On the basis of static, dynamic, redundant and non-redundant allocation we have four allocation strategies:

1.1 Static and Non-redundant Allocation

In this strategy, each fragment is allocated on a single site. The location of the fragments never changed, even if the access probability of nodes to fragments is changed.

1.2 Static and Redundant Allocation

In this strategy, same fragments may be allocated on multiple sites. The location of the fragments never changed, even if the access probability of nodes to fragments is changed.

1.3 Dynamic and Non-redundant Allocation

In this strategy, each fragment is allocated on a single site. The location of the fragments is changed, if the access probability of nodes to fragments is changed.

1.4 Dynamic and Redundant Allocation

In this strategy, same fragments may be allocated on multiple sites. The location of the fragments is changed, if the access probability of nodes to fragments is changed.

In this paper a new dynamic data allocation algorithm for non-redundant distributed database systems have been introduced which is an extension of [8]. The aim of this work is to design an efficient algorithm that can generate minimum total data transfer cost allocation schemes in changing load in non-redundant environment.

The rest of the paper is organized as follows: In section 2 the overview of the related work done so far is described. Section 3 provides the proposed new algorithm for non-redundant data allocation. In section 4 comparison of proposed new algorithm with algorithms proposed by [8], [30], [35] is performed. Section 5 concludes the contribution of the study.

2. RELATED WORK

Data allocation in a distributed database is one of the important and critical issues of the distributed database design. It directly affects the performance of the system. Several reports have been published on the problem of data allocation to the nodes. Firstly in [5] file allocation problem was investigated and a global optimization model was introduced to minimize overall operating costs under the constraints of storage capacity with fixed number of copies each file and response time. In [10] the assumption of fixed number of copies is relaxed and stressed the difference between retrieval and updates. In [11] it is proved that [10]'s formulation was NP-Complete and suggested heuristic approaches be investigated rather than deterministic approaches. In [12] a file allocation problem in distributed database environment is analysed for optimization of query processing.

By introducing replicated file, [13] showed how minimization of communication cost attributed to joins can be performed. In [14] the problem of file allocation for complex distributed database applications is considered with a simple model of transaction execution. In [13], [15] it is observed that the fragment allocation problem differs from the well-studied file allocation problem. In [15] the allocation of the distributed database to the sites is considered to minimize total data transfer cost and devised a comprehensive approach to allocate fragments. In [16] issues like queuing costs and concurrency are considered, while [17] presents a max-flow approach. In [18] an integrated approach for data fragmentation and allocation is provided, and seven criteria that a system designer can use to determine the data fragmentation, replication and allocation are identified.

The approach for allocating fragments by adapting a machine learning approach is provided by [19]. In [20] a concurrency mechanism is introduced and [21] presents a replication algorithm that adaptively adjusts to changes in read-write patterns. In [22] an approach based on Lagrangian relaxation is considered and [23] explained heuristic approaches. Besides allocating data, [24] and [25] presented a mathematical modelling approach and a genetic algorithm based approach to allocate operations to nodes. In [26] an integer programming formulation for the non-redundant version of the fragment allocation problem is described. Moreover [27] has given a high-performance computing method for data allocation in distributed database system. In [28] the problem of distributing fragments of virtual XML repositories over the web is described. The problem of distributing the documents of a web site among the server nodes of a geographically distributed web server is considered by [29].

Several works have been introduced for dynamic data allocation in database systems over past few years. A model for dynamic data allocation is introduced by [21]. An algorithm which reallocates data with respect to changing data access pattern is proposed by [30]. In [19] an approach based on machine learning is presented. In [31] incremental allocation and reallocation based on changes in workload is considered. In [32] a dynamic algorithm with centralized control for object allocation and replication is presented. In [33] security considerations into the dynamic file allocation process are considered. An optimal algorithm for non-replicated database systems is proposed by [34]. In [35] a threshold algorithm for non-replicated distributed databases is introduced. In the threshold algorithm, the fragments are continuously reallocated according to the changing data access patterns. In [8] an algorithm namely TTCA (an extension of work carried out by [30] and [35], [36]) is described, which reallocates non-replicated data with respect to the changing data access patterns with time constraint in distributed database systems. In this paper, a new dynamic data allocation algorithm for non-redundant distributed database system has been proposed which is an extension of work carried out by [8]. This new proposed algorithm dynamically reallocates data for non-redundant allocation in distributed database systems.

3. NEW PROPOSED DYNAMIC DATA ALLOCATION ALGORITHM

In distributed database system the cost of executing queries is heavily depend on the data transfer cost which occurred in transferring fragments accessed by a query from different sites to the site where the query is initiated [2], [6]. The key objective of any data allocation algorithm in distributed environment is to place fragments at different sites in a way so that the total cost of data transfer during the execution of a query can be minimized.

The Optimal Algorithm [30] begins with the distribution of fragments in non-replicated manner over the different sites using a static data allocation method. Thereafter for each locally stored fragment the algorithm maintains access counters matrix at each site. Whenever a node made an access request for the stored fragment then access counter of the accessing node for the stored fragment is incremented by one. No movement of fragment is required if the accessing node is the current owner. In a case if the counter of a remote node is greater than the counter of the current owner, then fragment has to move to the accessing node. The main drawback of this algorithm is that if the changing frequency of access pattern for each fragment is high, then it will spend more time for fragment transferring to different sites.

Threshold Algorithm [35] overcomes the problem of optimal algorithm. In threshold algorithm initially the fragments are distributed in non-replicated manner over the different sites using a static data allocation method. In this algorithm only one counter per fragment is maintained and initial value of the counter is zero. The counter value is increased by one for each remote access to the fragment. It is reset to zero for a local access. In other words, the counter always shows the number of successive remote accesses. Whenever the counter exceeds a predetermined threshold value, the ownership of the fragment is transferred to another node. This algorithm delays the migration of the fragment from any node for at least (t+1) accesses, where t is the value of the threshold. Migration of fragments depends on the value of the threshold. If the threshold value decreases then migration of the fragment will be more. In case the threshold value increases then there will be less migration of the fragments. The main drawback of this algorithm is that 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.

The Threshold and Time Constraint Algorithm (TTCA) [8] solved the problems of threshold algorithm. In TTCA initially all the fragments are distributed over different nodes using any static allocation method in non-replicated manner. After the initial allocation, TTCA maintains access counters matrix for each locally stored fragment at each node with initial value set to 0. Every time an access request is made for the stored fragment then the access counter of the accessing node is incremented by one. If the counter of the remote node is greater than the threshold value "t" and all the last "t+1" accesses are made in a specified time "T" then reset the corresponding fragments counter to zero for all the node and transfer the fragment to the node who's counter value was greater than threshold value. But TTCA has following problems with its approach:

- If owner site counter is increasing as a consequence of several local accesses and its value becomes greater than the threshold value "t" and all other remote nodes counter values are less than the threshold value, then fragment is not migrated and corresponding fragment's counter value is not reset to zero for all the nodes. If there is further several local accesses are made then its counter value is continuously increasing, this may result in scaling problem. For example, if one byte is chosen to store the counter values, then a value greater than 255 cannot be stored in this data type.
- If all the "t+1" accesses by the remote node is not made in a specified time "**T**", then the fragment will not be migrated and corresponding fragment's counter value is not reset to zero for all the nodes. If there is further several remote accesses are made then its counter value is continuously increasing, this may again result in scaling problem.
- The time constraint of TTCA says that if all the "t+1" accesses by the remote node is not made in a specified time "**T**" (say 01 day / 01 week / 01 month…), then the fragment will not be migrated to the remote node who's counter value is greater than the threshold value "t". It shows that over same time span fragment is more required by the remote node as compared to local node. Even then fragment is not migrated to remote site, as a consequence more remote references has to be performed, that is against the main aim of dynamic data allocation.

The new proposed algorithm named as Extended Threshold Algorithm (ETA) will remove all the above problems of threshold and time constrained algorithm. The ETA is illustrated as follow:
Initially all the fragments are distributed over different nodes using any static allocation method in non-redundant manner. ETA maintains an **m×n** counter matrix **M**, where **m** denotes the total number of fragments and **n** denotes the total number of nodes or sites. $\mathbf{M}_{ij}$ is the number of accesses to fragment i by node j.

The matrix M is decomposed into rows and each row is stored together with its associated fragment in the same node. In this way, whenever the fragment migrates, its associated counters migrate as well. Fig. 2 shows fragment 'i' with its associated counters, M_0 through M_n.

Fragment_i	$M_0\ M_1\ M_2\ \ldots\ M_n$

Fig 2: Any fragment 'i' in extended threshold algorithm

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

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.

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 owner node is greater than the threshold value "t", then reset the corresponding fragment's counter to zero for all the node, and go to Step 2.

Step 6: If the counter of remote node is greater than the threshold value "t", then reset the corresponding fragment's counter to zero for all the node and transfer the fragment to the node whose counter value was greater than the threshold value "t".

Step 7: Go to step 2.
The ETA will further decrease the space requirement as time constraint is not stored. It well suits the main aim of dynamic data allocation in distributed database.

4. COMPARISON

Comparison of proposed algorithm - ETA with algorithms Optimal, Threshold and TTCA has been made on the following four different parameters:

- Storage Cost
- Migration Condition
- Network Overhead
- Scaling Problem

4.1 Storage Cost

Optimal algorithm and ETA use extra storage cost for access counter matrix. Threshold algorithm required less storage cost as compared to ETA and Optimal algorithms, because it stores only one counter for each fragment. TTCA requires more storage as compare to optimal, ETA and threshold algorithms, as it stores not only access counter matrix but respective time of particular access also.

4.2 Migration Condition

Optimal algorithm migrate the fragment when the counter value of the remote node in the access matrix is greater than the counter value of the owning node. TTCA migrates the 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". Threshold algorithm and ETA migrates the fragment when the counter value of the remote node is greater than the threshold value.

4.3 Network Overhead

Optimal algorithm increases the traffic on the network when changing frequency of access pattern for each fragment is high. Threshold algorithm, TTCA and ETA decrease the network overhead as compared to optimal algorithm by limiting the migration of fragments.

4.4 Scaling Problem

Optimal and TTCA algorithms may suffer from scaling problem (data type range overflow for the counter). But Threshold algorithm and ETA are not suffered from scaling problem.

5. CONCLUSION

In the age of globalization, distributed databases are used by almost all the organizations across the globe. Deciding the technique by which organizational database is distributed in a distributed environment is an important issue, as it affects both cost and system performance.

The allocation of data is traditionally static and determined off-line, using estimates of access frequencies. The static data allocation techniques provide only limited response to changing workload. The proposed new dynamic data allocation algorithm- ETA for non-redundant distributed database systems theoretically shows an edge over Optimal, Threshold and TTCA and improves the overall performance of the system. In future, we can practically implement ETA for non-redundant distributed database and further enhanced it for redundant distributed databases.

REFERENCES

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

[2] S. Ceri and G. Pelagatti, "Distributed Databases: Principles & Systems", McGraw-Hill International Editions, 1985.

[3] http://www.vocw.edu.vn

[4] 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.

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

[6] M. Ozsu and P. Valduriez, "Principles of Distributed Database Systems", Prentice Hall, second ed. 1999.

[7] http://el.mdu.edu.tw/datacos//0941231101A/Lecture 03.doc

[8] Arjan Singh and K.S. Kahlon, "Non-replicated Dynamic Data Allocation in Distributed Database Systems", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.9, September 2009

[9] http://courses.washington.edu/tcss545/tcss545A_ 14.ppt

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

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm," ACM Trans. Database Systems, vol. 22, no. 2, pp. 255-314, 1997.

[22] 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.

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

[24] 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.

[25] 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.

[26] S. Menon, "Allocating Fragments in Distributed Databases", IEEE Transactions on Parallel and Distributed Systems, Vol. 16, No. 7, July 2005.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] L.S. John, “A Generic Algorithm for Fragment Allocation in Distributed Database System”, ACM 1994.

[35] 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.

[36] T. Ulus and M. Uysal, “A Threshold Based Dynamic Data Allocation Algorithm- A Markove Chain Model