Fragment Allocation in Distributed Database Systems

Reza Basseda

Faculty of Electrical and Computer Eng., School of Engineering, University of Tehran

Database Research Group

R.Basseda@ece.ut.ac.ir

Abstract

Choosing appropriate fragment allocation in Distributed Database Systems is an important issue in design of Distributed Database Management System .Several fragment-allocation algorithms have been suggested in the area of DDBMS design. Each algorithm has its own strengths and shortcomings. Therefore, providing an appropriate algorithm is too critical in DDBMS design and may lead to an increase in Distributed Database Performance. This article presents an overview of research works on fragment allocation algorithms in Database Research Group of University of Tehran during last year. In order to clarify the suggested algorithms and improvements in this area, we conclude discussion with a summary on our evaluations.

1. Introduction

Developments in database and networking technologies in the past two decades led to advances in distributed database systems. A DDS is a collection of sites connected by a communication network, in which each site is a database system in its own right, but the sites have agreed to work together, so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the user's own site [11]. The primary concern of a DDS is to design the fragmentation and allocation of the underlying data. Fragmentation unit can be a file where allocation issue becomes the file allocation problem. File allocation problem is studied extensively in the literature, started by Chu [12] and continued for non-replicated and replicated models [13, 14]. Some studies considered dynamic file allocation [15, 16].

Various approaches have already been described the data allocation technique in distributed systems [1], [4], [5], [6]. Some methods are limited in their theoretical and implementation parts [8], [9]. Other strategies are ignoring the optimization of the transaction response time. The other approaches present exponential time of complexity and test their performance on specific types of network connectivity [2].

Data allocation problem was introduced when Eswaran [17] first proposed the data fragmentation. Studies on vertical fragmentation [18, 19]; horizontal fragmentation [20] and mixed fragmentation [21] were conducted. The allocation of the fragments is also studied extensively.

In these studies, 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 the 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. Initial studies on dynamic data allocation give a framework for data redistribution and demonstrate how to perform the redistribution process in a minimum possible time. In [3] a dynamic data allocation algorithm for non-replicated database systems is proposed named *optimal* algorithm, but no modeling is done to analyze the algorithm. In [5] the *threshold* algorithm is proposed for dynamic data allocation algorithm, which reallocates data with respect to changing data access patterns. It focused on load balancing issues.

A major cost in executing queries in a distributed database system is the data transfer cost incurred in transferring relations (fragments) accessed by a query from different sites to the site where the query is initiated. The objective of a data allocation algorithm is to determine the assignment of fragments at different sites so as to

minimize the total data transfer cost incurred in executing a set of queries. This is equivalent to minimizing the average query execution time, which is of primary importance in a wide class of distributed conventional or multimedia database systems.

2. Near Neighborhood Allocation

The NNA algorithm is basically a variation of the optimal algorithm [3]. In optimal algorithm, all fragments are initially distributed over the nodes according to any static method but afterwards, any node j, runs the optimal algorithm described as follows for every fragment I, that it stores.

(1) For each (locally) stored fragment, initialize the access counter rows to zero. ($S_{ij} = 0$

were i ε fragment indexes and j ε nodes)

(2) Process an access request for the stored fragment

(3) Increase the corresponding access counter of the accessing node for the stored fragment. (If node (x) accesses fragment i, set $S_{ix} = S_{ix} + 1$)

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

(5) If the counter of a remote node is greater than the counter of the current owner node, transfer the ownership of the fragment together with the access counter array to the remote node. (i.e fragment migrates) (If node x accesses fragment i and $S_{ix} > S_{ij}$, send fragment i to node (x))

(6) Repeat from step 2.

The problem of this approach is that if the changing frequency of access pattern for each fragment is high, it will spend a lot of time for transferring fragments to different nodes. So, the response time and delay will be increased.
In our algorithm, we are going to address the problem of optimal algorithm:
In NNA algorithm, the requirement for moving a fragment is obtained as in optimal algorithm. But, the destination of the moving data is different. In our method we consider the network topology and routing for specifying destination. In other words the destination of the moved fragment is the neighbor of the source, which is in the path from the source to the node with highest access pattern. We have chosen link state routing algorithm for its simplicity of implementation . Any routing algorithm can be used equally.

By using this approach we avoid moving data too frequently because since the fragment will be placed finally in a node which has the average access cost for nodes that use it. So, the delay of movement will be reduced and the response time will also be improved.
Another aspect of NNA algorithm is that the fragments, which are used by a node or neighbors of a node, can be clustered. Using this clustering approach, we can respond to the data requests more effectively.

Evaluation of Algorithm

In our experiments, we consider two factors: average delay for receiving the response (response time) for a fragment request and average time spent for moving data from one node to another (fragment data migration time). We will investigate the effect of different parameters on these factors.
We examined the effect of different parameters on these factors. Findings of our experiments indicated that, the NNA algorithm performs better for larger fragment size and query production rate, but for small fragment size and query production rate, the optimal algorithm performs better. The threshold for fragment size is almost 8000 bytes. For larger networks, by using NNA algorithm we can decrease the delay of response to a fragment regarding to optimal algorithm.

3. BGBR: A variation of NNA which looking for appropriate location

In this algorithm, we proposed an approach based on NNA, but we are going to predicate perfect position of fragment to have less response time and fragment data migration time. BGBR dynamically determine the location of a fragment that provides an overall optimal system performance. This approaches contribution compared to the optimal algorithm is that the optimal algorithm does not consider the complete topology, it only considers the node the fragment is located and any other node with higher access to that fragment. Although it does make sense to move a fragment close to nodes with frequent access to that fragment, the optimal choice is not necessarily determined by selecting a node with higher access to that fragment, a complete analysis of the topology is required to obtain the optimal node. Failure to consider the complete topology will result in frequent oscillation. Although, the NNA algorithm shows improvement in respect to the

optimal algorithm by preventing heavy oscillation, it too does not consider the complete topology and takes too long to converge. The NNA algorithm moves fragments from its located node to a node with higher access in the direction of the shortest path one hop at a time; however, there is no assurance that the shortest path is the best path to be moving a node when considering the global topology and access patterns. Assuming the shortest path is best path, unlike BGBR, NNA converges slowly by taking many hops to find the optimal location. As a fragment is moving along the shortest path towards the optimal location the access patterns could be completely changed, resulting in wasted effort.

The BGBR algorithm is performed in three steps:

1. Determination of Shortest Paths
2. Aggregate Bottom Up
3. Determination of the Minimum Cost

Initially at startup or after a topology change the algorithm begins by running Dijkstra's algorithm to obtain the shortest path from one node to every other node. This algorithm is continuously repeated until we have a shortest path matrix that determines the shortest path from every node to every other node. The BGBR algorithm keeps track of the number of times a fragment has been accessed by each and every node. This is simply implemented with the use of counters. However, once a fragment is relocated then its counters all become zero. The idea in this step is to determine the highest priority fragment to be relocated. This is an important step due to the limited physical hardware space in typical systems. If two fragments want to reside in a location that only has space for one of these fragments, it is important to choose the fragment with higher priority. Algorithms normally define the priority of a fragment based on individually assessing which object has been accessed the most by which node without considering the relationship among the various nodes accessing the same object Once we have determined and sorted the objects based on their priority we need to dequeue objects from the priority queue and select the optimal location for each. The optimal location can be calculated by selecting the location that provides the minimum cost. More importantly to determining the minimal cost, is to determine the cost of placing an object on a node. The idea behind determining the cost is to place objects so that they are close to the nodes that access them.

Evaluation of Algorithm

BGBR outperforms both NNA and the optimal algorithm when considering the Total Query Cost. Comparing the NNA algorithm with Optimal you will notice that NNA starts to perform better when the fragment sizes are greater than 9MB. However, BGBR outperforms both in all fragment sizes. This is precisely because fragments are placed in a location that provides the overall minimum access cost. BGBR has better performance than both NNA and Optimal when considering the Total Fragmentation Migration Cost. The problem with the optimal algorithm is that fragments continuously are moved from one node to the other as the access patterns change. Optimal does not consider moving fragments in a location which is beneficial to all nodes that have accessed a specific object. It only considers the owner of a fragment and the node who has accessed an object more times than the owner. Evidently, this algorithm results in a lot of fragment mobility. NNA on the other hand tries to resolve this problem by not making drastic changes. Fragments are moved hop by hop. The reason that BGBR is much better than the other two is precisely because BGBR picks a location that will not lead to much fragment mobility. This location is determined based on the overall access patterns

4. A Fuzzy approach to improve NNA performance

All of above mentioned algorithms using crisp method to move data along network paths. Estimating time and place (destination) of a data fragment depends on various parameters such as access pattern, bandwidth of network links and etc. The FNA algorithm is basically a variation of the NNA [3].

As mentioned above, detecting oscillation is important in DDBMS. Rapid changing of access pattern for a single fragment may cause problems in DDB system. Fragment migration between two sites leads high delay in accessing fragment. Migrating fragment is inaccessible during fragment migration because fragment is locked when it moves from one site to another. We are going to solve this problem by detection oscillation state and avoiding from moving fragment. In this way, we consider access pattern of site and recognize oscillation state through

differentiation of access pattern. After degrading of access pattern with using mean factor, we use a fuzzy and operator between smoothed access pattern and fuzzy compliment of differentiation of access pattern. The result show revised access pattern, which can be used in deciding of fragment migration. We trace access counts in 20 time sluts. Each time slut comprises of 50-clock cycle. Designed fragment allocation system's architecture is as bellow:

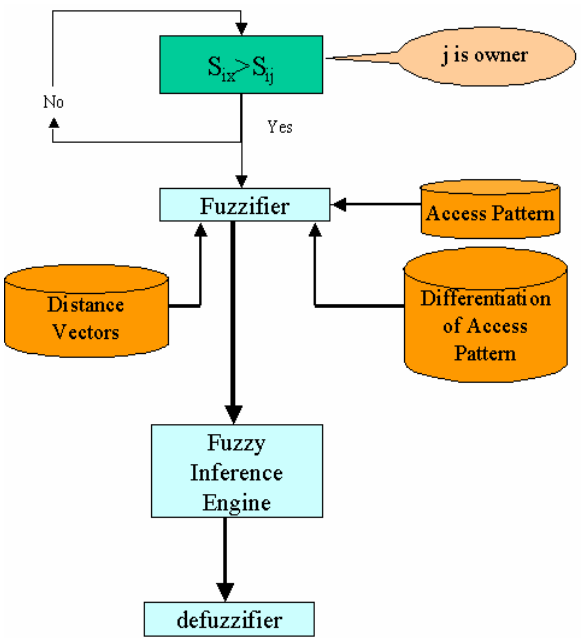


Figure 1 Architecture of Fuzzy Fragment Allocator

Evaluation of Algorithm

We examined the effect of different parameters on these factors. Findings of our experiments indicated that, the FNA algorithm performs better for larger fragment size and query production rate, but for small fragment size and query production rate, the NNA and optimal algorithm performs better. For larger networks, by using FNA algorithm we can decrease the delay of response to a fragment regarding to optimal algorithm.

5. Conclusion

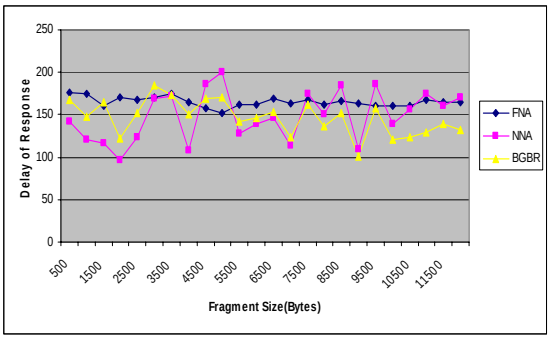
In this article, we introduce some method to distribute data fragment of Distributed Database Systems over the sites. As mentioned above, NNA algorithm is a simple variation of optimal algorithm which can be used in a simple design of Distributed Database Systems.

BGBR is more complicated variation of NNA which shows better results but it requires more calculations and it needs to capture more data. Consequently, it can be used in systems which less response time is more vital.

FNA needs more space to capture history of access pattern than other algorithms and it requires complicated calculations such as fuzzification of access pattern and detection of oscillation in access pattern. Despite of these complexities in space and time of Fuzzy approach, it seems better than other algorithms in oscillation conditions.

We compared these algorithms in detail. We developed software to simulate algorithms and compare them. This simulator is configurable for testing different network topologies and different data requests and/or allocation conditions. In our simulator we mark each packet's send and receive time. Using time stamp, we could compare algorithms in different factors. Detailed information regarding to the implementation of this software is available in [24] and [25].

In our experiments, we consider two factors: average delay for receiving the response (response time) for a fragment request and average time spent for moving data from one node to another (fragment data migration time). We will investigate the effect of different fragment size on these factors. Figures 2 and 3 show the effect of fragment size on these factors. According to Figure 2, for small fragment size, the average time spent for response in FNA and BGBR algorithms is larger than NNA and for larger fragments this is reversed. In larger fragments, BGBR acts better than FNA. The reason is that for small fragments the cost of moving data to destination node is low and so, the movement cost does not exceed the access cost. In the case of large fragments the movement of fragments takes more time and also increase the network traffic. So, less movement will produce some advantages that overcome the access cost. Avoidance of oscillation condition in FNA leads to have less traffic and saving in network resources such as bandwidths. In BGBR, we predicate an appropriate location for fragment and this fact prevents unnecessary fragment migration and leads to have less traffic. In FNA destination of a data fragment is chosen according to access pattern of over all system. So, we direct our fragments more effective and this will be valuable in larger fragments. Results have been reported in different conferences.



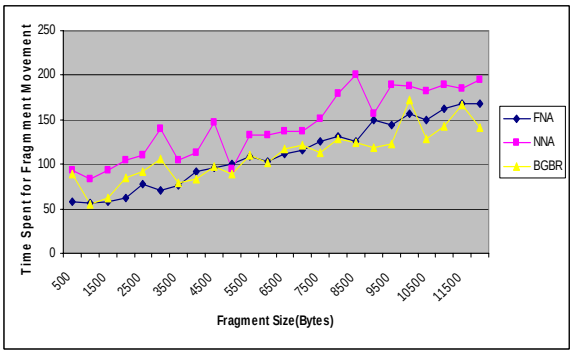


Figure 3 shows that time spent for fragment movement is larger in NNA and BGBR than FNA. This is caused by preventing unnecessary fragment movement especially in oscillation conditions.

Here we just studied these algorithms on non-replicated distributed database systems. Further studies are needed to test FNA, BGBR, NNA and optimal algorithms in replicated distributed database systems.

References

[1] Berkan, R. C., Trubatch, S. L., Fuzzy Systems Design Principles, IEEE Press, New York, 1997

[2] Basseda, R., Tasharofi, S., Rahgozar, M., Near Neighborhood Allocation (NNA): A Novel Dynamic Data Allocation Algorithm in DDB, In proceedings of 11th Computer Society of Iran Computer Conference (CSICC2006), Tehran, 2006

[3] John, L. C., A Generic Algorithm for Fragment Allocation in Distributed Database Systems, ACM, 1994.

[4] Ahmad, I., Karlapalem, K., Kwok, Y. K., and So, S. K. Evolutionary Algorithms for Allocating Data in Distributed Database Systems, International Journal of Distributed and Parallel Databases, 11: 5-32, The Netherlands, 2002.

[5] Brunstroml, A., Leutenegger, S. T. and Simhal, R., Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database with changing Workloads, *ACM Transactions on Database Systems*, 1995.

[6] Chin, A. G., Incremental Data Allocation and ReAllocation in Distributed Database Systems, Journal of Database Management; Jan-Mar 2001; 12, 1; ABI/INFORM Global pg. 35.

[7] Ulus, T., and Uysal, M., Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems, Pakistan Journal of Information and Technology 2 (3), 2003, ISSN 1682-6027, 231-239.

[8] Voulgaris, S., Steen, M. V., Baggio, A., and Ballintjn, G., Transparent Data Relocation in Highly Available Distributed Systems. Studia Informatica Universalis. 2002.

[9] Navathe, S. B., Ceri, S., Wiederhold, G. and Dou, J., Vertical Partitioning Algorithms for Database Design, ACM Transaction on Database Systems, 1984, 680-710.

[10] Apers, P. M. G. , "Data allocation in distributed database systems," ACM Transactions on Database Systems, vol. 13, no. 3, 1988, 263–304.

[11] Huang, Y. F. and Chen, J. H., Fragment Allocation in Distributed Database Design, Journal of Information Science and Engineering 17, 2001, 491-506.

[12] Hababeh, I. O., A Method for Fragment Allocation Design in the Distributed Database Systems, The Sixth Annual U.A.E. University Research Conference, 2005.

[13] Özsu, T., and Valduriez, P., Principles of Distributed Database Systems. Prentice-Hall Book Co., Englewood Cliff, USA, 1991.

[14] Chu, W. W., Optimal File Allocation in a Multiple Computer System, IEEE Transactions on Computers, C-18, 1969, 885-889.

[15] Morgan, H.L., and Levin, K. D., Optimal Program and Data Locations in Computer Networks, Communications of ACM, 20, 1977, 315-321.

[16] Azoulay-Schwartz, R., and Kraus, S., Negotiation on Data Allocation in Multi-Agent Environments, Autonomous Agents and Multi-Agent Systems, 5, 2002, 123-172.

[17] Wah, B. W., Data Management in Distributed Systems and Distributed Data Bases, Ph.D. Dissertation, University of California, Berkeley, CA, USA, 1979.

[18] Smith, A. J., Long-term File Migration: Development and Evaluation of Algorithms, Communications of ACM, 24, 1981, 512-532.

[19] Eswaran, K. P., Placement of Records in a File and File Allocation in a Computer Network, in Proceedings of IFIP Congress on Information Processing, Stockholm, Sweden, 1974, 304-307.

[20] Navathe, S. B., Ceri, S., Wiederhold, G., and Dou, J., Vertical Partitioning Algorithms for Database Design, ACM Transaction on Database Systems, 9, 1984, 680-710.

[21] Ceri, S., Pernici, B., and Wiederhold, G., Optimization Problems and Solution Methods in the Design of Data Distribution, Information Systems, 14, 1989, 261-272.

[22] Ceri, S., Navathe, S. B., and Wiederhold, G., Distribution Design of Logical Database Schemas, IEEE Transactions on Software Engineering, 9 , 1983, 487-503.

[23] Zhang, Y., and Orlowska, M. E., On Fragmentation Approaches for Distributed Database Design, Information Sci., 1, 1994, 117-132.

[24] Basseda, R. and Tasharofi, S., Design and Implementation of an Environment for Simulation and Evaluation of Data Allocation Models in Distributed Database Systems, Technical Report No. DBRG.RB-ST.A50701, 2005.

[25] Basseda, R. and Tasharofi, S., Data Allocation in Distributed Database Systems, Technical Report No. DBRG.RB-ST.A50715, 2005.