Non-Replicated Dynamic Fragment Allocation in Distributed Database Systems

Nilarun Mukherjee

Senior Lecturer, Department of CSE/IT
Bengal Institute of Technology, Kolkata, West Bengal, India
nilarun.mukherjee@gmail.com

Abstract. Distributed databases have become the most essential technology for recent business organizations. The performance, efficiency and availability of a Distributed Database largely depend on the fragmentation of global relations and the allocation of those fragments at several sites of the network. In Non-Replicated fragment allocation scenario, the optimum allocation of the fragments is the only way, which can be exploited to increase the performance, efficiency, reliability and availability of the Distributed Database. In this paper, a new dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario, which incorporates access threshold, time constraints of database accesses and most importantly the volume of data transmitted to dynamically reallocate fragments to sites at runtime in accordance with the changing access probabilities of nodes to fragments. It will decrease the migration of fragments and data transfer cost and will also improve the overall performance by dynamically reallocating fragments in a most intuitive manner.

Keywords: Distributed Database, Fragmentation, Dynamic Fragment Allocation, Local Agent, Root Agent.

1 Introduction

A Distributed Database is a collection of data, which logically belong to the same system, but are distributed over the sites of a computer network. Each site of the network has autonomous processing capability and can perform local database applications [9]. Each site also participates in the execution of at least one global database application, which requires accessing data residing at several different sites from geographically dispersed locations, using a communication subsystem.

The primal motivations for distributed databases are to improve performance, to increase the availability of data, shareability, expandability and access facility. In a distributed database, maximization of the locality of processing of database applications by allocating data as close as possible to the applications which use them significantly reduces the communication overhead with respect to a centralized database.

In most of the cases Distributed Databases are developed in a Bottom-Up approach, where there already exist several centralized databases located at several geographically dispersed sites, which are integrated to form the Distributed Database. In this scenario, there is no scope of fresh fragmentation, as the fragments exists beforehand and they are analyzed and integrated to form the global relations. Moreover, fragmentation depends highly on the business and organizational requirements. Therefore, the optimum allocation of those fragments is the only way, which can be exploited by the designers to increase the performance, efficiency, reliability and availability of the Distributed Database. The main aim is to maximize the locality of processing for each distributed application by storing the fragments closer to where they are more frequently used in order to achieve best performance. This means placing data i.e. fragments required by the distributed applications at their site of origin or at sites which are closer to their site of origin. I.e. to maximize the "local" references and to minimize "remote" references to the data by each distributed application with respect to its site of origin [9]. This is done by adding the number of local and remote references corresponding to each candidate fragmentation and fragment allocation, and selecting the best solution among them, i.e. the solution providing highest processing locality or Complete Locality to maximum number of distributed applications [9]. The *Complete Locality* means the distributed application can be completely executed at its site of origin; thus reducing overall remote accesses.

Various approaches have already been evolved for dynamic allocation of data in distributed database [1] to improve the database performance. These address the more realistic dynamic environment, where the access probabilities of nodes to fragments change over time. Fragment allocation can further be divided in two ways: Non – Redundant Fragment Allocation: Each fragment of each global relation is allocated to exactly one site. Redundant Fragment Allocation: Fragments of each global relation are allocated to one or more sites introducing replication of the fragments.

In this paper, a new dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario, which incorporates access threshold, time constraints of database accesses and most importantly the volume of data transmitted to dynamically reallocate fragments to sites at runtime in accordance with the changing access pattern i.e. the changing access probabilities of nodes to fragments over time. The proposed algorithm will decrease the movement of fragments over the network and data transfer cost and will also improve the overall performance of the system by dynamically allocating fragments in a most intuitive manner.

2 Related Work

Till date many works have been published on the problem of allocation of data or fragments of the global relations to geographically dispersed sites of a distributed database. [10] considered the problem of file allocation for typical distributed database applications with a simple model of transaction execution. [11] incorporated issues like concurrency and queuing costs. [12] provides an integrated approach for fragmentation and allocation. [12] identified seven criteria that a system designer can use to determine the fragmentation, replication and allocation. [15] presents a replication algorithm that adaptively adjusts to changes in read-write patterns. [16] provides an approach based on Lagrangian relaxation and [17] describes heuristic approaches. More recently [18] has given a high-performance computing method for data allocation in distributed database system.

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

Over past few years, work has been introduced for dynamic fragment allocation in distributed database systems. [15] gives a model for dynamic data allocation for data redistribution and incorporates a concurrency mechanism. In [4] an algorithm is proposed for dynamic data allocation, which reallocates data with respect to the changing data access patterns. [13] provides approach for allocating fragments by adapting a machine learning approach. [7] considers incremental allocation and reallocation based on changes in workload. [19] incorporated security considerations into the dynamic file allocation process. In [20] an optimal fragment allocation algorithm for non-replicated distributed database systems is proposed. [21] has introduced a threshold algorithm for non-replicated fragment allocation in distributed databases. In the threshold algorithm, the fragments are continuously reallocated according to the changing data access patterns.

In this paper, a new dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario which is an extension of the work carried out by [1] and [4, 21, 6]. The proposed algorithm incorporates access threshold, time constraints of database accesses and most importantly the volume of data transmitted to dynamically reallocate fragments to sites at runtime in accordance with the changing access pattern i.e. in accordance with the changing access probabilities of nodes to fragments over time.

3 Design

The main factor that affects the efficiency and turnaround time of the applications of a distributed database is the time delay for transferring the data needed by a certain distributed database query or application over the network from the fragments located at several different geographically dispersed / remote sites to the site of origin of the application or query. Therefore, the primary aim is to allocate fragments accessed or needed by certain distributed database application either at the same site where the distributed database application / query is invoked or at sites closer to the site of origin, so that the data transmission over the network is minimized during the execution of the distributed database application / query. On the other hand, it is not a feasible solution to place the entire data of a distributed database at every site of the system; also it will violate the basic requirements of a distributed database [9]. The placing of fragments to the same sites or to the sites which are closer to the sites, where from the distributed database applications needing those fragments are invoked, is a considerably complex problem. Especially, in the scenario where the probability or pattern of accessing fragments by the distributed database applications at different sites changes dynamically over time.

The proposed algorithm for dynamic fragment allocation in distributed database exploits the concepts of the existing algorithms: the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1]. In distributed database the efficiency and turnaround time of the distributed database applications ultimately depends on the volume of data that is required to get transferred over the network from one site to another for the execution of those applications. Reallocating a fragment from a certain site to another site, which makes the highest number of accesses to that fragment and not to the site, which even though does not make the highest number of accesses to that fragment but results in transmission of maximum volume of data from or to that fragment, will not be always beneficial i.e. will not always improve the overall performance or efficiency of the distributed database.

The main concept of this work is to reallocate a fragment located at a certain site to another site, which not only makes highest number of accesses to that fragment in a specific period of time but also results in transmission of maximum volume of data from or to that fragment in that specific period of time. Moreover, most of the time the site which makes the highest number of accesses to a particular fragment located at a particular site in a specific period of time also results in transmission of maximum volume of data from or to that fragment in that specific period of time. Thus, for most of the cases the proposed algorithm for dynamic fragment allocation will reflect the most realistic scenario and will result in more intuitive and optimum dynamic fragment reallocation.

Furthermore, reallocation of fragments containing huge amount of data from one site to another site over the network just in order to make the distributed database applications to have their required data at their site of origin or at the sites very closer to their site of origin, incurs huge data transmission over the network and results in transmission overload. Because, fragments are expected to contain huge amount of data as compared to the data required by and transmitted due to the execution of those distributed database applications. Thus, it is not desirable to have frequent migrations of fragments over the network, as this can significantly affect or degrade the overall performance or efficiency of the distributed database. The proposed algorithm improves the overall performance or efficiency of the distributed database by imposing a more strict condition for fragment reallocation and results in fewer migrations of fragments from one site to other over the network during the execution of the distributed database application / query, in Non-Replicated allocation scenario as compared to the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1].

Algorithm:

1. Initially all the fragments of all the global relations of the distributed database are distributed over different sites using any static allocation method in non-replicated manner. Let there be total N number of fragments of global relations distributed among the total M number of sites in the distributed database system. Each site has one or more fragments allocated to it. A distributed database query or application may require accessing several different fragments allocated at several different sites for execution.
2. The proposed algorithm needs each site to maintain a separate data structure named Access Log, which stores certain information regarding each access to the fragments allocated at that site, by the distributed database queries or applications invoked at the same or different sites. Each Access Log record denoted by A_k^h (i.e. kth access at site h, where k = 1,2,3,…to infinity and h = 1,2,3,...M) stores the following information regarding each access:

   a. Name or Identifier of the Fragment accessed.
   b. Address of the accessing site.
   c. Date and Time of the access.
   d. The volume of data transmitted from or to that fragment in Bytes.

3. Two parameters are used to tune the performance of the algorithm:
   a. The Time Constraint for Fragment Reallocation (τ). It determines the duration of the time intervals.
   b. The Access Threshold for Fragment Reallocation (η).

The following Steps [4 to 6] are performed at each site for each individual access to a fragment allocated at that site by a certain distributed database query or application invoked at the same or different site. Suppose at site h an access is made and processed for fragment i allocated at that site from site j at time t, where h = 1,2,3,…M, i = 1,2,3,...N and j = 1,2,3,...M and h = j or h $\neq$ j. The local agent at site h performs the following operations:

4. Write a log record A_k^h in Access Log at site h.
5. If the address of the accessing site in the log record A_k^h is same as the address of site h, i.e. the access is made from the same site (h = j), then do nothing.
6. Else if the address of the accessing site in the log record A_k^h is different from the address of site h, i.e. the access is made from a different site (h $\neq$ j); then:
   a. Calculate the total number of accesses made from all the sites (including site h) to the fragment i located at site h within the time interval τ up to current access time t. Let n_i^m, denotes the total number of accesses from the site m to the fragment i allocated at site h, within the time interval τ up to current access time t, where m = 1,2,3,...M.
   b. Calculate the Average volume of data transmitted (in bytes) in between the fragment i and all the sites (including site h) where from the accesses to the fragment i located at site h are made, through the accesses occurred within the time interval τ up to current access time t. Let $A_k^h V_i^m$ denotes the volume of data transmitted (in bytes) in between the fragment i allocated at site h and the site m in the access A_k^h at certain point of time, where m = 1,2,3,...M. The Average volume of data transmitted in bytes (denoted by $V_i^m t$) in between the fragment i allocated at site h and the site m through the accesses occurred within the time interval τ up to current access time t can be calculated as:

$$V_i^m t = \frac{\sum A_k^h V_i^m}{n_i^j} \quad (1)$$

   Where A_k^h occurred within the time interval τ up to current access time t. $V_i^m t$ is calculated for all the sites (including site h) where from the accesses to the fragment h are made, through the accesses occurred within the time interval τ up to current access time t.
   c. If the total number of accesses from the site j to the fragment i allocated at site h, within the time interval τ up to current access time t is greater than the access threshold for Fragment Reallocation (η) and is greater than the total number of accesses made from all other sites (including site h) to the fragment i

located at site h within the time interval τ up to current access time t i.e. if $n_i^j > \eta$ and $n_i^j > n_i^m$ where j, m = 1,2,3,...M and $m \neq j$ and $h \neq j$, then the fragment i is migrated and reallocated to site j and removed from the current site h, catalogs are updated accordingly, if and only if the Average volume of data transmitted (in bytes) in between the fragment i allocated at site h and the site j through the accesses occurred within the time interval τ up to current access time t is greater than the Average volume of data transmitted (in bytes) in between the fragment i and all other sites (including site h) where from the accesses to the fragment i located at site h are made, through the accesses occurred within the time interval τ up to current access time t, i.e. if and only if $V_i^j t > V_i^m t$ where j, m = 1,2,3,...M and $m \neq j$ and $h \neq j$. Otherwise, do nothing.

This dynamic fragment allocation algorithm migrates a fragment located at a certain site to another site, which not only makes number of accesses to that Fragment greater than the Access Threshold for Reallocation (η) in the specific period of time determined by the Time Constraint for Fragment Reallocation (τ) up to current access time, but also results in transmission of maximum volume of data from or to that fragment in that specific period of time.

In distributed database, at each site the local agent of a remote or local distributed transaction accesses or interacts with the local database where the fragments of the global relations are actually stored [9]. In case of insertion or modification operations on a fragment, the local agent receives the data to be inserted or to be used to modify the existing data in the particular fragment from the root agent of the distributed transaction. Thereafter, the local agent sends those data along with the command (insert or update) to the local transaction manager, which actually accesses the database table corresponding to that fragment stored at the local database to perform insertion or modification. In case of retrieval, the local agent receives the data retrieved from the database table stored at the local database corresponding to a particular fragment through the local transaction manager. Thereafter, it sends those retrieved data to the root agent of the distributed transaction executing at the same or different site in the distributed database network. In all the cases the local agent of the distributed database transaction temporarily retains the data to be inserted or to be used to modify the existing data in the particular fragment or being retrieved from the particular fragment during the execution of a distributed database application or query. Moreover, the local agent belongs to the distributed database transaction management system and it is designed and programmed by the developer of the distributed database management system. Therefore, it is possible to enhance the transaction management system and the local agents of the distributed database transactions to add the functionality of writing Access Log Records and to calculate the volume of data transmitted from or to each fragment (in Bytes) at each fragment access, i.e. to implement the proposed dynamic fragment allocation algorithm.

4 Comparative Study

In Optimal Algorithm [4], initially all the fragments are distributed over the different sites using any static data allocation method. After the initial allocation, system maintains an access counter matrix for each locally stored fragment at each site or node.

Every time an access request is made for the locally stored fragment, the access counter of the accessing site for that fragment is increases by one. If the counter of a remote node becomes greater than the counter of the current owner, then the fragment is moved to the accessing node. The problem of Optimal Algorithm [4] 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 and will incur huge data transmission overload. Threshold algorithm [21] solves the problem of optimal algorithm and decreases the migration of fragments over different sites and reduces the transmission overload as compare to the Optimal Algorithm [4]. Threshold algorithm guarantees the stay of the fragment for at least (η +1) accesses at the new node after a migration, where η 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 resets the counter of local fragment to zero, every time a node is going for a local access and it does not specify which node will be the fragment's new owner when the counter exceeds the threshold value, also it does not give the information about past accesses of the fragments and does not consider the time variable of the access pattern.

The TTCA [1] algorithm removes all the above problems of threshold algorithm [21] by adding a time constraint to consider the time of the accesses made to a particular fragment. TTCA [1] decreases the migration of fragments over different sites, thus reduces the transmission overload due to fragment migrations as compare to simple Threshold algorithm [21]. TTCA [1] maintains a counter matrix at each site to store the number of accesses made by the sites to a particular fragment located at that site, which is incremented on each access. It reallocates data with respect to the changing data access patterns with time constraint. It migrates a fragment from one site to another site, which has most recently accessed that fragment for η + 1 numbers of times in a time period τ up to the current access time, where η is the Threshold Value and τ is the Time Constraint. If the value of time constraint increases then the migration of fragment will be more. But, if the value of time variable decreases then there will be less migration of fragments. But TTCA [1] does not store the time of those accesses in that counter matrix. Therefore, it does not provide any conceivable method to determine within which time period the certain number of accesses (determined by the value of the counter) made by a certain site to a particular fragment located at another site have occurred.

The proposed algorithm will remove all the above problems of the Threshold [21] and TTCA [1] algorithm. It will dynamically reallocate fragments at runtime in accordance with the changing access pattern i.e. the changing access probabilities of sites to fragments over time. While migrating fragments from one node to another node, this algorithm not only considers the threshold value and time constraint of database accesses and the time of the accesses made to a particular fragment, but also considers the volume of data transmitted. This dynamic fragment allocation algorithm migrates a fragment located at a certain site to another site, which not only makes number of accesses to that fragment greater than the Access Threshold for Reallocation (η) in the specific period of time determined by the Time Constraint for Fragment Reallocation (τ) up to current access time, but also results in transmission of maximum volume of data from or to that fragment in that specific period of time. It recognizes the fact that reallocating a fragment from a certain site to another site, which makes the highest number of accesses to that fragment and not to the site, which even though does not make the highest number of accesses to that fragment but results in transmission of maximum volume of data from or to that fragment, will not be always beneficial i.e. will not always improve the overall performance or efficiency of the distributed database. The proposed algorithm for dynamic fragment allocation reflects the most realistic scenario and results in the optimum dynamic fragment reallocation, in Non-Replicated allocation scenario. The proposed algorithm improves the overall performance or efficiency of the distributed database by imposing a more strict condition for fragment reallocation and resulting in fewer migrations of fragments from one site to other over the network and further reduces the transmission overload due to fragment migrations as compared to the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1].

Moreover, the administrators of the distributed database implementing this dynamic fragment allocation algorithm can easily relax or tighten the condition for fragment reallocation by regulating the values of the Time Constraint for Fragment Reallocation (τ) and the Access Threshold for Fragment Reallocation (η), to tune the algorithm to work best to fulfill the functional, infrastructural and business requirements of the organization implementing the distributed database. The choice of the most appropriate values of τ and η serves as the key factors for determining the optimum performance of the algorithm, to minimize the frequency of unnecessary fragment migrations and to reduce the movement of data over the network required during the execution of the distributed database queries or applications by increasing their locality of processing i.e. to maximize the overall throughput, performance and efficiency of the distributed database applications. For a fixed value of τ, if the value of η is increased, then the migration of fragments from one site to other will decrease vice versa. For a fixed value of η, if the value of τ is increased, then the migration of fragments from one site to other will increase vice versa. I.e.

$$\begin{array}{c}\text{Fragments Migration /}\\ \text{Reallocation Frequency}\end{array} \propto \frac{\tau}{\eta} \tag{2}$$

The proposed algorithm requires more storage space and increases the storage cost due to the maintenance of the Access Log at each site, in which log records are created for each access. Although, it reduces some storage space requirement by eliminating the usage of the counter matrix at each site, but it incurs more computational overhead per database access at each site due to the calculation of the total number of accesses made from all other sites to a particular fragment located at that site within the time interval τ up to current access time and due to the calculation of the Average volume of data transmitted (in bytes) in between that fragment and all other sites where from the accesses to that fragment are made, through the accesses occurred within the time interval τ up to current access time. But these drawbacks are well compensated by the advantage gained from the reduction in the frequency of unnecessary fragment migrations as compared to the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1].

5 Conclusion

In this paper a new sophisticated dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario which is an extension of the work carried out by [1] and [4, 21, 6]. The proposed algorithm incorporates access threshold, time constraints of database accesses and most importantly the volume of data transmitted to dynamically reallocate fragments to sites at runtime in accordance with the changing access pattern i.e. in accordance with the changing access probabilities of nodes to fragments over time. This dynamic fragment allocation algorithm migrates a fragment located at a certain site to another site, which not only makes number of accesses to that Fragment greater than the Access Threshold for Reallocation (η) in the specific period of time determined by the Time Constraint for Fragment Reallocation (τ) up to current access time, but also results in transmission of maximum volume of data from or to that fragment in that specific period of time. It reflects the most realistic scenario and results in the more intuitive and optimum dynamic fragment reallocation, in Non-Replicated allocation scenario. The proposed algorithm improves the overall performance or efficiency of the distributed database by imposing a more strict condition for fragment reallocation in distributed database and resulting in fewer migrations of fragments from one site to other over the network as compared to the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1]. Moreover, the administrators of the distributed database implementing this dynamic fragment allocation algorithm can easily relax or tighten the condition for fragment reallocation by regulating the values of the Time Constraint for Fragment Reallocation (τ) and the Access Threshold for Fragment Reallocation (η), to tune the algorithm to work best for the functional, infrastructural and business needs of the organization. The choice of the most appropriate values of τ and η serves as the key factors for determining the optimum performance of the algorithm, to minimize the frequency of unnecessary fragment migrations and to minimize the movement of data over the network required during the execution of the distributed database queries or applications by increasing their locality of processing i.e. to maximize the overall throughput, performance and efficiency of the distributed database applications.

References

1. Singh, A., Kahlon, K.S.: Non-replicated Dynamic Data Allocation in Distributed Database Systems. IJCSNS International Journal of Computer Science and Network Security 9(9), 176–180 (2009)
2. Vasileva, S., Milev, P., Stoyanov, B.: Some Models of a Distributed Database Management System with data Replication. In: International Conference on Computer Systems and Technologies-CompSysTech 2007, vol. II, pp. II.12.1—II.12.6 (2007)
3. Agrawal, S., Narasayya, V., Yang, B.: Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design. In: Proc. 2004, ACM SIGMOD International Conf. Management of Data, pp. 359–370 (2004)
4. Brunstroml, A., Leutenegger, S.T., Simhal, R.: Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database with changing Workload. ACM Trans. Database Systems (1995)

5. March, S., Rho, S.: Allocating Data and Operations to Nodes in Distributed Database Design. IEEE Trans. Knowledge and Data Eng. 7(2), 305–317 (1995)
6. Ulus, T., Uysal, M.: A Threshold Based Dynamic Data Allocation Algorithm- A Markove Chain Model Approach. Journal of Applied Science 7(2), 165–174 (2007)
7. Chin, A.: Incremental Data Allocation and Reallocation in Distributed Database Systems. Journal of Database Management 12(1), 35–45 (2001)
8. Hanamura, H., Kaji, I., Mori, K.: Autonomous Consistency Technique in Distributed Database with Heterogeneous Requirements. In: IPDPS Workshops 2000, pp. 706–712 (2000)
9. Ceri, S., Pelegatti, G.: Distributed Database Principles & Systems. McGraw-Hill International Editions
10. Ceri, S., Martella, G., Pelagatti, G.: Optimal file Allocation for a Distributed Database on a Network of Minicomputers. In: Proc. International Conference on Database, Aberdeen, British Computer Society, Hayden (July 1980)
11. Ram, S., Narasimhan, S.: Database Allocation in a Distributed Environment: Incorporating a Concurrency Control Mechanism and Queuing Costs. Management Science 40(8), 969–983 (1994)
12. Karlaplem, K., Pun, N.: Query-Driven Data Allocation Algorithms for Distributed Database Systems. In: Tjoa, A.M. (ed.) DEXA 1997. LNCS, vol. 1308, pp. 347–356. Springer, Heidelberg (1997)
13. Tamhankar, A., Ram, S.: Database Fragmentation and Allocation: An Integrated Methodology and Case Study. IEEE Trans. Systems, Man and Cybernetics Part A 28(3) (May 1998)
14. Chaturvedi, A., Choubey, A., Roan, J.: Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach. IEEE Trans. Eng. Management 41(2), 194–207 (1994)
15. Wolfson, O., Jajodia, S., Huang, Y.: An Adaptive Data Replication Algorithm. ACM Trans. Database Systems 22(2), 255–314 (1997)
16. Chiu, G., Raghavendra, C.: A Model for Optimal Database Allocation in Distributed Computing Systems. In: Proc. IEEE INFOCOM 1990, vol. 3, pp. 827–833 (June 1990)
17. Huang, Y., Chen, J.: Fragment Allocation in Distributed Database Design. J. Information Science and Eng. 17, 491–506 (2001)
18. Hababeh, I.O., Ramachandran, M., Bowring, N.: A high-performance computing method for data allocation in distributed database systems. J. Supercomput. 39, 3–18 (2007)
19. Mei, A., Mancini, L., Jajodia, S.: Secure Dynamic Fragment and Replica Allocation in Large-Scale Distributed File Systems. IEEE Trans. Parallel and Distributed Systems 14(9), 885–896 (2003)
20. John, L.S.: A Generic Algorithm for Fragment Allocation in Distributed Database System. ACM, New York (1994)
21. Ulus, T., Uysal, M.: Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems. Pakistan Journal of Information and Technology 2(3), 231–239 (2003)