Dynamic Distributed Database over Cloud Environment

Ahmed E. Abdel Raouf, Nagwa L. Badr, and Mohamed Fahmy Tolba

Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
Ahmed_ezzat991@yahoo.com, nagwabadr@cis.asu.edu.eg,
fahmytolba@gmail.com

Abstract. An efficient way to improve the performance of database systems is the distributed processing. Therefore, the functionality of any distributed database system is highly dependent on its proper design in terms of adopted fragmentation, allocation, and replication methods. As a result, fragmentation, its allocation and replication is considered as a key research area in the distributed environment. The cloud computing is an emerging distributed environment that uses central remote servers and internet to maintain data and applications. In this paper, we present a dynamic distributed database system over cloud environment. The proposed system allows fragmentation, allocation, and replication decisions to be taken dynamically at run time. It also allows users to access the distributed database from anywhere. Moreover, we present an enhanced allocation and replication technique that can be applied at the initial stage of the distributed database design when no information about the query execution is available.

Keywords: Distributed database management system (DDBMS), fragmentation, replication, allocation, cloud computing.

1 Introduction

Distributed database systems typically consist of a number of distinct database fragments located at different geographic sites which can communicate through a network and they are managed by a distributed database management system (DDBMS) [1].

An efficient support is needed to databases that consist of very large amounts of data which used by applications at different physical locations. Telecom databases, scientific databases, and large distributed enterprise databases are examples of application areas [2]. The main problem of many of these applications is the delay of accessing remote databases. As a result, it is necessary to use a distributed database that employing fragmentation, allocation, and replication [2].

The design of distributed database is one of the major research issues in distributed database system area. The main challenges facing the DDBS design are: how to fragment database tables, which type of fragmentation will be used, when to replicate fragments, what is the optimal number of replica that can be taken for each fragment to enhance system performance and increase availability, how to allocate fragments to sites where they are mostly frequently accessed, do we grouping distributed database sites into disjoint clusters, and which type of clustering will be used. These issues were previously solved either by static and dynamic solution or based on a priori query analysis.

A.E. Hassanien et al. (Eds.): AMLTA 2014, CCIS 488, pp. 67–76, 2014.

Fragmentation, replication, and allocation are considered the most important design issues that lead to optimal solutions particularly in a dynamic distributed environment. They also have a great impact on the Distributed Database Systems (DDBS) performance. In distributed databases, the communication costs can be reduced by partitioning database tables into fragments. The fragments are then allocated to the sites where they are most frequently accessed, aiming at maximizing the number of local accesses compared to accesses from remote sites. The cost of the read operation can be further reduced by the replication of fragments when beneficial. Fragmentation, allocation, and replication will be referred to as FAR in the rest of the paper.

Many applications of DDBS generate very dynamic workloads with frequent changes in access patterns from different sites. Consequently, static/manual FAR may not always be optimal. As a result, FAR should be automatic and completely dynamic. Any change in access patterns should result in re-fragmentation of existing fragments or tables and reallocation of fragments to different sites, as well as creation or removal of fragment replicas [2].

In this paper, we present a dynamic DDBS over the cloud environment that allows dynamic FAR decisions to be taken dynamically over clustered distributed database sites. Dynamic FAR decisions are based on the access pattern and the load of the sites after allocation or migration of fragments or a replica to it. Moreover, we present an optimal allocation and replication technique, which can be applied at the initial stage of the distributed database design when no information about the query execution is available.

The rest of this paper is organized as follows. Section 2 reviews the related work of FAR. Section 3 introduces the proposed system architecture and its components. Section 4 presents an optimal allocation and replication technique that can be applied at the initial stage of distributed database design. Section 5 presents the experimental results. Finally the conclusion and future work are presented in Section 6.

2 Related Research Work

In [2], the authors present a decentralized approach for dynamic table FAR in DDBS. It performs FAR based on recent access history. It uses cost functions to take decision by estimating the difference in future communication costs between a given replica change and keeping it as is. However, this algorithm doesn't consider site constraints, load of the site after the allocation or migration of replica or fragment to it, and the optimal number of replicas that can be taken for each fragment to enhance system performance and increase availability. The authors of [3] present a synchronized horizontal FAR model. It adopts a new approach to perform horizontal fragmentation of database relation based on the attribute retrieval and update frequency. Both [3] and [18] perform the allocation process based on the fragment access pattern and the cost of moving the data fragments from one site to the other. The authors of [4] propose a new algorithm called region based fragment allocation (RFA). The proposed algorithm considers the frequency of fragment accessed by region as well as individual nodes to move the fragment from source node to target node. The RFA algorithm

decreases the migration of fragments using knowledge of the network topology in comparison to optimal [19] and threshold [20] algorithms. In comparison to the BGBR [21] algorithm, the RFA algorithm reduces the amount of topological data required in decision making. However, this solution will not be the optimal solution in case the node in a region has a high fragment access while the other nodes in that region have low fragment access. As a result, the fragment will allocate to it. In this case, the problem is solved for only one node and is not solved for the other nodes in other regions. The authors of [5] present a model that takes site constraints into account in the process of reallocation. However, this model will be more complicated when information queries continuously change in a faster way and when the number of fragments and sites largely increase. The authors of [6] propose an algorithm named Near Neighborhood Allocation (NNA). It moves data to a neighborhood node that placed in the path to the node with the maximum access counter. The NNA could be more useful in large networks to decrease the delay of response time. The authors of [7] propose an algorithm that dynamically reallocates fragments to sites at runtime. The proposed algorithm takes into account the time constraints of database accesses, volume threshold, and the volume of data transmitted in successive time intervals in accordance with the changing access patterns. However, the volume threshold, the number of time intervals, and its duration are the most important factors that regulate the frequency of fragment reallocations. In [8] the authors perform reallocation process given the changing data access patterns, time, and sites' constraints of the DDBS. That technique reduces data transmission cost compared to the previous methods since it adopts the shortest path algorithm once data movement decision is taken. The main drawback of this algorithm is the more storage required compared to some previous algorithms. The authors of [15] solve the fragment allocation problem using the well-known Quadratic Assignment Problem solution algorithms.

The authors of [9] propose a new technique for horizontal fragmentation of the relations of distributed database. This technique can be applied at the initial stage as well as in later stages of DDBS for partitioning the relations. The authors of [10] address some important scalability issues. They provide some algorithms to ensure generality of the technique developed in [9]. In [11] the authors propose a heuristic technique to satisfy horizontal fragmentation and allocation using a cost model to minimize the total cost of distribution. Furthermore, the authors of [12] and [16] present a new framework for dynamic fragment allocation and replication. In [12], they consider the fragment correlation under a flexible network topology. However, this framework doesn't handle extra characteristics, such as bounds on the capacity of sites and constraints on the number of replicas for each fragment.

Some researches introduce a clustering approach for partitioning database sites and allocate fragments across the sites of each cluster [13]. However, it doesn't address the replication phase of database design. It also wastes a lot of time between node, LCA, LCA Validator, Resource Checker, and GCA until it reaches the required data. The authors of [17] propose a dynamic data replication strategy using historical access record and proactive deletion. The authors of [14] present a novel algorithm for grouping distributed database network sites into disjoint clusters based on communication time. The experimental outcomes confirmed that this approach can be

implemented in a different DDBS environment even if the network sites are enormous. However, this work doesn't mention how to allocate fragments to cluster of sites and which sites in the cluster will hold the fragment and on which bases. Another method of clustering the sites in which low communication cost sites are grouped in one cluster[1]. Furthermore it allows the fragmentation of structured data, fragmentation of unstructured data, and it describes the allocation of fragments to the cluster of sites in order to reduce communication cost. However, this work doesn't mention which sites in the cluster will hold the fragment when the fragment is allocated to it.

3 Dynamic Distributed Database System over Cloud Environment Architecture

To overcome the limitations of existing literature highlighted by the above survey, we propose a DDBS design over cloud environment. The proposed architecture allows users to access a database from anywhere in the world without owning any technology infrastructure. It can be accessed through: a web browser, mobile application or desktop application while the database is stored on servers at remote sites. It also allows FAR decisions to be taken dynamically at run time. These decisions are based on access patterns and load of sites after the allocation and migration of fragments as well as its replicas. The proposed architecture is shown in Fig. 1. It consists of two layers: distributed database system manager and distributed database clusters layers.

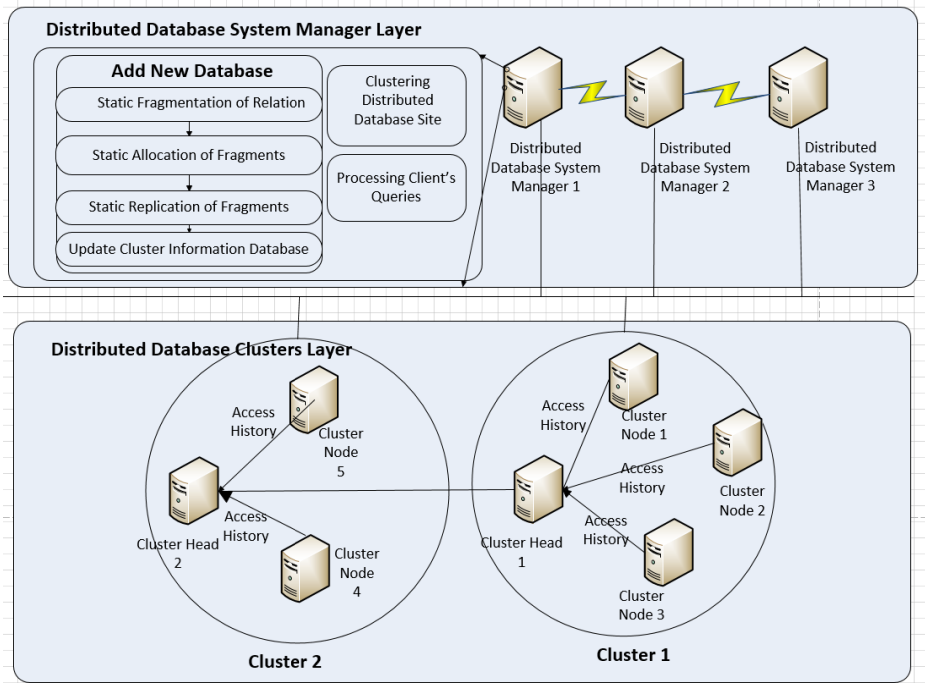

Fig. 1. Dynamic Distributed Database System over Cloud Environment Architecture

3.1 Distributed Database System Manager Layer

This layer is composed of three modules: add new database, clustering distributed database sites, and client queries processing.

Add New Database Module. This module is used to add the database tables to DDBS. Firstly, the new tables are fragmented using the static fragmentation technique that can be used at the initial stage of the DDBS design. Secondly, that fragments are allocated to the sites of each cluster. Thirdly, some fragments are replicated to different sites of each cluster in order to enhance the system performance and increase the availability. Finally, the cluster information database is updated to save the locations of each fragment and replica in the distributed database system. Cluster information database is a database that holds complete information about each fragment or replica in which site or cluster.

Clustering Distributed Database Sites Module. Distributed database system manager will use clustering technique to cluster distributed database sites into disjoint clusters. It also allocates sites to each cluster. Finally, it updates cluster information database with the recent location of each site in DDBS.

Client Queries Processing Module. When the client sends a query to the distributed database system manager firstly, the distributed database system manager uses the cluster information database to determine the closest site that holds the required data. Then, it redirects the user to that site to fetch the needed data.

3.2 Distributed Database Clusters Layer

The distributed database clusters layer consists of more than one cluster. Each cluster has one cluster head and more than one cluster node. The cluster head is a cluster node has especial and additional operations to manage the other cluster nodes. At each access to the cluster node, firstly, the cluster node checks whether it is a local access or remote. Secondly, allows the user to access local database of the node to run the query and fetch the required data [22]. Thirdly, the site access record is updated to save the information about the user access. Each cluster node sends the access history to the cluster head to take any suitable decision such as: create or delete replica, re-fragmentation or re-allocation of the fragments. The cluster head takes the decisions based on access history and load of each site after the allocation or migration of the fragments and replicas of it. The fragments and replicas will send to the site with less workload. The cluster head also collects the data that each node in its cluster holds. Afterwards, it sends it to the distributed database system manager to keep the cluster information database updated.

4 Enhanced Allocation and Replication Technique

The authors of [9] proposed a new technique of horizontal fragmentation. This technique helps in taking the fragmentation decision at the initial stage of designing distributed database. It uses knowledge gathered during the requirement analysis phase using the enhanced CRUD (Create, Read, Update, and Delete) matrix without the help of empirical data about query execution. This technique performs data allocation according to the maximum attribute locality precedence (ALP) value and the locations of sites. The attribute locality precedence (ALP) can be defined as the importance of an attribute with respect to each site. However, the allocation strategy of this technique doesn't meet the goal of the data allocation. The goal of the data allocation can be achieved by allocating the fragments to the sites that require it only. As a result the user can access data with low cost and time. The proposed module enhanced the allocation strategy of this technique by performing data allocation according to the site that has the maximum ALP single value. That methodology guarantees that no fragment duplication will happen during the allocation process. In the case we got two sites that have the maximum ALP single value, the fragment will be allocated to the site that performs more data manipulations and less read operations. Consequently, the replica is sent to the site which performs less data manipulations and more read operations. After the process of data allocation, we replicate the fragment to the site that performs more read operation than other sites. For example, if we have two sites: site1 and site2. Site1 performs CUD operations. Site2 performs R operation. The replica will be sent to site2 although site1 has the maximum ALP single value. If the replica is sent to site1, it will not be used because there is no data manipulation on the replica. The pseudo code for the enhanced allocation and replication technique is shown in Fig. 2.

5 The Experimental Results

We have implemented our technique on an HP Compaq computer with Core-two Duo 2.33 processors and 2 GB RAM using SQL Server as DBMS. We have implemented the modified technique on the MCRUD matrix of the bank account table shown in table 1. We performed the enhanced technique on account relations shown in table 2. The ALP table is generated after applying the modified technique on the MCRUD matrix. The ALP table is shown in table 3. From the ALP table, the attribute that has a maximum ALP value is the branch name, so its predicates will be used to perform horizontal fragmentation. The resulted fragments are shown in tables 4, 5, and 6 and are allocated to sites that already need it without taking the location of sites into account. Fragment1 has been allocated to site1 and fragment2 has been allocated to site2 because they have the maximum ALP single value for the fragmentation attribute. However, the last predicate of the branch name attribute has two sites which have the same maximum value. In this case, the fragment will be allocated to the site that performs more data manipulation and less read

operations. In our experiments, site1 performs two read operations and site3 performs three read operations. As a result, fragment3 will be allocated to site1 and its replica will be sent to site3.

```
Input:  Number of attributes, number of predicates of each attribute [], number of sites, number of
applications of each site [], and MCRUD matrix [total number of predicates, total number of applications]
Output:  The ALP table, the sites that contain maximum ALP single value for each predicate of each
attribute (Position of MAX [attribute, predicate]), and the sites that performs more read operations for
each predicate of each attribute (Position of next Max [attribute, predicate]).
Foreach attribute in Number of attributes do
     Foreach predicate in number of predicates [attribute] do
          Number of read operation of max = 0
          Number of read operation of next max = 0
          Application number = 0
          Total sum = 0
            Foreeach site in number of sites do
                 Application sum = 0
                 Number read operation = 0
                     Foreeach application in number of application [site] do
                       Application sum += Calculate MCRUD (MCRUD [Predicate number, application number])
                       Application number++     End
                     Total sum += application sum
                     If application sum == MAX [attribute, predicate] then
                           If Number read operation > Number of read operation of max then
                               Position of next Max [attribute, predicate] = site
                               Number of read operation of next max = Number read operation   End
                           Else if Number read operation < Number of read operation of max then
                                Position of next Max [attribute, predicate] =
                                    Position of MAX [attribute, predicate]
                                Number of read operation of next max = Number of read operation of max
                               MAX [attribute, predicate] = application sum
                               Position of MAX [attribute, predicate] = site
                               Number of read operation of max = Number read operation  End End
                     If application sum > MAX [attribute, predicate] and application sum > 0 then
                         If Number of read operation of max > Number of read operation of next max then
                              Position of next Max [attribute, predicate] =
                                   Position of MAX [attribute, predicate]
                             Number of read operation of next max = Number of read operation of max End
                           MAX [attribute, predicate] = application sum
                           Position of MAX [attribute, predicate] = site
                           Number of read operation of max = Number read operation   End
                     Else if application sum > 0 and Number read operation > 0 and
                             Number read operation > Number of read operation of next max then
                           Position of next Max [attribute, predicate] = site
                           Number of read operation of next max = Number read operation;    End End
                  Predicate number++
                  ALP Single [attribute, predicate] = MAX [attribute, predicate] -
                            (Total sum - MAX [attribute, predicate])        End
              Foreach predicate in number of predicates [attribute] do
                  ALP FINAL [attribute] += ALP Single [attribute, predicate]  End End
```

Fig. 2. Pseudo Code for the enhanced allocation and replication technique

After the allocation process, we replicate the fragments to enhance the system performance of reading only queries and increase the availability. The replicas will be allocated to the site that performs more read operations than other sites. In our scenario, replica1 of fragment1 is allocated to site3, replica2 of fragment2 is allocated to site1, and replica3 of fragment3 is allocated to site3. The final results of the allocation and the replication processes are shown in table 7.

Table 1. MCRUD Matrix

	Site1			Site2			Site3		
	AP1	AP2	AP3	AP1	AP2	AP3	AP1	AP2	AP3
Account. Account id > 20	C		RU						C
Account. Account id <= 20		R							
Account .Type= ind	CRD	RU	RUD		R				
Account .Type= cor		RU	R				CRUD	RU	R
Account .customer id > 5	C		RU						R
Account .customer id <= 5		R							
Account .open date > 1-1-2008	CRD	RU	RU		R				
Account .open date <= 1-1-2008		RU	R				CRUD	RU	R
Account .Balance < 10000	R		R			CRUD			R
Account .Balance >= 10000		CR							
Account .Branch Name = dhk	CRUD	RU	CRUD		CUD		R		
Account. Branch Name = ctg		R		CRUD	CRUD	R		R	
Account. Branch Name = khl	CRUD	CRU	U				CRUD	CRU	CR

Table 2. Account Relation

account_no	Account Type	customer id	open date	Account Balance	Account Br Name
3	ind	1	20/01/2009 ...	12500.0000	dhk
4	cor	2	20/05/2009 ...	12000.0000	dhk
7	ind	2	05/03/2009 ...	11000.0000	ctg
15	ind	3	08/05/2009 ...	11000.0000	khl
20	cor	2	08/05/2010 ...	15000.0000	ctg
21	ind	1	09/05/2012 ...	9000.0000	khl
22	cor	8	20/09/2011 ...	8000.0000	dhk
23	ind	5	06/08/2011 ...	6000.0000	khl
24	ind	9	08/09/2006 ...	15000.0000	khl
28	cor	5	07/05/2009 ...	16000.0000	ctg

Table 3. ALP TABLE

Attribute name	ALP Value
Account id	6
Type	22
customer id	6
open date	22
Balance	8
Branch Name	27

Table 4. Fragment 1

account_no	Account Type	customer id	open date	Account Balance	Account Br Name
3	ind	1	20/01/2009 ...	12500.0000	dhk
4	cor	2	20/05/2009 ...	12000.0000	dhk
22	cor	8	20/09/2011 ...	8000.0000	dhk

Table 5. Fragment 2

account_no	Account Type	customer id	open date	Account Balance	Account Br Name
7	ind	2	05/03/2009 ...	11000.0000	ctg
20	cor	2	08/05/2010 ...	15000.0000	ctg
28	cor	5	07/05/2009 ...	16000.0000	ctg

Table 6. Fragment 3

account_no	Account Type	customer id	open date	Account Balance	Account Br Name
15	ind	3	08/05/2009 ...	11000.0000	khl
21	ind	1	09/05/2012 ...	9000.0000	khl
23	ind	5	06/08/2011 ...	6000.0000	khl
24	ind	9	08/09/2006 ...	15000.0000	khl

Table 7. Final Result of Allocation and Replication

Fragment Number	Allocated to	Replicated to
Fragment 1	1	3
Fragment 2	2	1
Fragment 3	1	3

6 Conclusion

Efficient distribution of the distributed database fragments and replicas to various sites play a critical role in the function of the database in terms of performance and cost. In this paper, we present a dynamic DDBS over cloud environment. The proposed system allows FAR decisions to be taken based on access history and the load of the site. Moreover, we present enhanced allocation and replication technique, which allocates the fragments and replicas to the sites that already, requires it without taking into account the location of the sites. The enhanced technique aims at maximizing the number of local access compared to access from the remote sites, enhance the system performance, and increase the availability.

As proposed future work, we plan to use an enhanced clustering technique to cluster distributed database sites into disjoint clusters then do allocation and replication

of, the fragments to sites of each cluster. Second, implement the remaining parts of the proposed architecture to efficiently allow FAR decisions to be taken automatically at run time.

References

1. Suganya, A., Science, C., Kalaiselvi, R.: Efficient Fragmentation and Allocation in Distributed Databases. Int. J. Eng. Res. Technol. 2, 1–7 (2013)
2. Hauglid, J.O., Ryeng, N.H., Nørvåg, K.: DYFRAM: dynamic fragmentation and replica management in distributed database systems. Distrib. Parallel Databases 28, 157–185 (2010)
3. Abdalla, H.I.: A synchronized design technique for efficient data distribution. Comput. Human Behav. 30, 427–435 (2014)
4. Varghese, P.P., Gulyani, T.: Region based Fragment Allocation in Non-Replicated Distributed Database System. Int. J. Adv. Comput. Theory Eng. 1, 62–70 (2012)
5. Abdalla, H.: A New Data Re-Allocation Model for Distributed Database Systems. Int. J. Database Theory 5, 45–60 (2012)
6. Gope, D.: Dynamic Data Allocation Methods in Distributed Database System. American Academic & Scholarly Research Journal 4, 1–8 (2012)
7. Mukherjee, N.: Synthesis of Non-Replicated Dynamic Fragment Allocation Algorithm in Distributed Database Systems. Int. J. on Information Technology 1, 36–41 (2011)
8. Abdallaha, H.I., Amer, A.A., Mathkour, H.: Performance optimality enhancement algorithm in DDBS (POEA). Comput. Human Behav. 30, 419–426 (2014)
9. Khan, S., Hoque, A.: A new technique for database fragmentation in distributed systems. Int. J. Comput. Appl. 5, 20–24 (2010)
10. Khan, S., Hoque, A.: Scalability and performance analysis of CRUD matrix based fragmentation technique for distributed database. In: 15th International Conference on Computer and Information Technology (ICCIT), pp. 557–562. IEEE, Chittagong (2012)
11. Abdalla, H.I., Amer, A.A.: Dynamic horizontal fragmentation, replication and allocation model in DDBSs. In: 2012 Int. Conf. Inf. Technol. e-Services, pp. 1–7. IEEE, Sousse (2012)
12. Kamali, S., Ghodsnia, P., Daudjee, K.: Dynamic data allocation with replication in distributed systems. In: 30th IEEE Int. Perform. Comput. Commun. Conf., pp. 1–8. IEEE, Orlando (2011)
13. Amalarethinam, D., Balakrishnan, C.: A Study on Performance Evaluation of Peer-to-Peer Distributed Databases. IOSR J. Eng. 2, 1168–1176 (2012)
14. Hababeh, I.: Improving network systems performance by clustering distributed database sites. J. Supercomput. 59, 249–267 (2010)
15. Tosun, U., Dokeroglu, T., Cosar, A.: Heuristic Algorithms for Fragment Allocation in a Distributed Database System. In: Gelenbe, E., Lent, R. (eds.) Computer and Information Sciences III, pp. 401–408. Springer, London (2013)
16. Amalarethinam, D., Balakrishnan, C.: oDASuANCO-Ant Colony Optimization based Data Allocation Strategy in Peer-to-Peer Distributed Databases. Int. J. Enhanc. Res. Sci. Technol. Eng. 2, 1–8 (2013)
17. Wang, Z., Li, T., Xiong, N., Pan, Y.: A novel dynamic network data replication scheme based on historical access record and proactive deletion. J. Supercomput. 62, 227–250 (2011)

18. Abdalla, H.I.: An Efficient Approach for Data Placement in Distributed Systems. In: 2011 Fifth FTRA Int. Conf. Multimed. Ubiquitous Eng., pp. 297–301. IEEE, Loutraki (2011)
19. Corcoran, L.: A Genetic Algorithm for Fragment Allocation a Distributed Database System. In: Proc. 1994 ACM Symp. Appl. Comput., SAC 1994, pp. 247–250. ACM, USA (1994)
20. Ulus, T., Uysal, M.: Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems. Inf. Technol. J. 2, 231–239 (2003)
21. Bayati, A., Ghodsnia, P.: A Novel Way of Determining the Optimal Location of a Fragment in a DDBS: BGBR. In: Syst. Networks, pp. 64–69. IEEE Computer Society, Washington (2006)
22. Maghawry, E.A., Ismail, R.M., Badr, N.L., Tolba, M.F.: An Enhanced Resource Allocation Approach for Optimizing Sub Query on Cloud. In: Hassanien, A.E., Salem, A.-B.M., Ramadan, R., Kim, T.-h. (eds.) AMLTA 2012. CCIS, vol. 322, pp. 413–422. Springer, Heidelberg (2012)