Distributed Database System (DSS) Design Over a Cloud Environment

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

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 including its allocation and replication is considered as a key research area in the distributed environment. Cloud computing is an emerging distributed environment that uses central remote servers and the internet to maintain data and applications. This research presents an enhanced dynamic distributed database system over a 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, this research presents 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. It also presents different clustering techniques and their advantages and disadvantages.

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 are managed by a distributed database management system (DDBMS) [15].

A.E. Abdel Raouf (✉) · N.L. Badr · M.F. Tolba
Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
e-mail: ahmed_ezzat991@yahoo.com

N.L. Badr
e-mail: dr.nagwabadr@gmail.com

M.F. Tolba
e-mail: fahmytolba@gmail.com

A.E. Hassanien et al. (eds.), *Multimedia Forensics and Security*,
Intelligent Systems Reference Library 115, DOI 10.1007/978-3-319-44270-9_5

An efficient support is needed to databases that consist of very large amounts of data which are used by applications at different physical locations. Telecom databases, scientific databases, and large distributed enterprise databases are examples of application areas [18]. 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 employs fragmentation, allocation and replication [18].

Fragmentation is the process of dividing a single database into two or more pieces; known as database fragment, the combinations of the pieces yielded from the original database are without any loss of information [15].

The process of placing data fragments at the sites are in order to, minimize the overall data transmission cost required to answer the query known as the allocation process [15].

To enhance the system performance and increase the availability, copies of the fragments are allocated to other locations within the distributed database. This process is known as the replication process. Too many copies of a fragment will enhance the performance of read only queries and increase the availability; however, it will also slow down updates. While too fewer copies of a fragment will decrease the performance of read only queries and the availability.

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

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 proposed research.

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 [18].

This research presents an enhanced dynamic DDBS over the cloud environment that allows dynamic FAR decisions to be taken dynamically over a 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, this research presents 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. It also presents different clustering techniques and their advantages and disadvantages and which one we will use in our system.

The rest of this paper is organized as follows; Sect. 2 reviews the related works of dynamic fragmentation, allocation and replication. The research issues are presented in Sect. 3. The solutions and recommendations are given in Sect. 4 and its subsections. Section 5 presents the enhanced allocation and replication technique. The experimental results are given in Sect. 6. Section 7 presents the previous works done in clustering distributed database sites. Finally, the future research directions and the conclusions are given in Sects. 8 and 9.

2 Background

The authors of [18] present a decentralized approach for dynamic table FAR in distributed database systems. It performs FAR based on recent access history, aiming at maximizing the number of local accesses compared to accesses from remote sites. In this approach FAR decisions are fully decentralized. Each site decides over its own fragments to split, migrate and/or replicate independently of other sites. The FAR decisions are made on the fly based on current operations and recent history of local reads and writes. This makes it possible to use this approach without communication overhead. This approach has two main components first, detecting replica access patterns. Second, based on these statistics to decide on re-fragmentation and reallocation. This approach ensures that decisions taken are not in conflict with each other by handling master replica and read replicas differently. It use cost functions to take decision by estimate the difference in future communication costs between a given replica change and keeping it as is. However, this algorithm doesn't consider site constraints, the load of the site after the allocation or migration of replica or fragments to it and the optimal number of replicas that can be taken for each fragment to enhance the system performance and increase availability.

The author of [3] presents a synchronized horizontal FAR model. It adopts a new approach to perform horizontal fragmentation of database relation based on attribute retrieval and update frequency. It proposes a new heuristic technique to satisfy horizontal fragmentation and allocation using a cost model to minimize the total cost of distribution. This technique performs the allocation process based on the fragment access pattern and the cost of moving the data fragments from one site to the other. It provides an optimal allocation that leads to best fragments distribution which avoids frequent remote accesses. In this technique first, performance and cost implications of different allocation choices are evaluated. Second, the replication

decision is taken based on the computed threshold values for the average retrieval and update costs of all fragments individually. This technique use site constraints to improve the efficiency which has been confirmed by the produced empirical results.

The process of placing data fragments at the sites in order to minimize the overall data transmission costs that are required to answer the query is known as fragment allocation. The fragment allocation has two types: Non-redundant fragment allocation and redundant fragment allocation. In non-redundant fragment allocation, each fragment of each global relation is allocated to exactly one site. As a result, 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 redundant fragment allocation, the fragments of each global relation are allocated to one or more sites introducing replication of the fragments.

The work of [1] proposes an approach that contributes in determining best possible allocation of a data fragment in a distributed environment. This approach is based on the fragment access patterns and the cost of moving data fragments from one site to the other. Different SAGA methods for data allocation were employed. However, the implementation confirmed that SAGA 100 outperformed all other SAGA.

The authors of [30] give the brief overview of dynamic fragment allocation in non-replicated distributed database system algorithms. They also propose 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 fragment from source node to target node. The RFA algorithm is designed to address the issues in existing approaches where fragments movement depends only on the frequency of access to the fragments. The RFA algorithm decreases the migration of fragments using knowledge of the network topology in comparison to optimal [14] and threshold [29] algorithms. In comparison to the BGBR algorithm [11], the RFA algorithm reduces the amount of topological data required in decision making. The proposed algorithm first, chooses the region that has high fragment access. Second, chooses node in that region that has high fragment access. Third, allocates the fragment to that node. However, this solution will not be the optimal solution in the case of a node in a region having 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 author of [2] presents a new data reallocation model for replicated and non-replicated constrained DDBSs. The proposed model takes site constraints into account in the process of reallocation. It reallocates data fragments across sites based on communication and update cost for each fragment individually. The reallocation process performed by selecting the site that has the highest query updates cost for fragment Fi to be chosen as the candidate site to store fragment Fi in order to, minimize communication cost. If the candidate site that chosen to store fragment violate site constraints then, the fragment will be migrate to the site with the next highest update cost value. Moreover, if more than one site has the same update cost for a certain fragment. In this case, this model will use fragment priority

(FP) procedure to allocate fragment to the site with the highest FP value in order to, avoid fragment duplication over sites. The main advantages of this model are that any change in the site queries and their frequency will have an effect on the reallocation process. In addition, this model guarantees that no fragment duplication at post allocation. However, this model will be more complicated when queries information continuously changes in faster way or when the number of fragments and sites largely increase.

The work of [16] proposes a new dynamic data allocation algorithm for non-replicated distributed database system. The proposed algorithm named Near Neighborhood Allocation (NNA). The NNA algorithm reallocates data with respect to the changing in data access pattern with time constraint. This algorithm is based on optimal algorithm, but with different strategy for selecting nodes for data movement. It moves data to a node which is the neighborhood and also placed in the path to the node with the maximum access counter. The experiments of this algorithm find that the NNA algorithm performs better for large fragment size and query production. However, the optimal algorithm performs better for small query production and fragment size. It also finds that the threshold for the fragment size is almost 8000 byte. The NNA can be used for larger networks to decrease the delay of response to a fragment regarding to optimal algorithm.

The work of [22] proposes a new integer programming formulations for the non-redundant version of the fragment allocation problem. This work assumes that the fragments already determined, and focuses on the problem of allocating them in such a way as to minimize the total cost resulting from transmissions generated by user queries.

The authors of [10] present and analyze a new approach for dynamic data allocation algorithm named Fuzzy Neighborhood Allocation (FNA) algorithm. This algorithm is based on NNA in [27]. It is different with NNA in selecting nodes for data movements and different strategy in migrating fragments. The proposed algorithm uses fuzzy method to prevent redundant data fragment migration and avoid oscillation condition. The experiments results indicated that, the proposed algorithm performs better for larger fragment size. However, the NNA and optimal algorithm performs better for small fragment size. The future of this work is to test FNA, NNA and optimal algorithms in replicated distributed database systems.

The author of [23] proposes a new dynamic fragment allocation algorithm in non-replicated allocation scenario. 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 order to, dynamically reallocate fragments to sites at runtime in accordance with the changing access patterns. The proposed 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 in the 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. The proposed algorithm improves the overall performance of the distributed database by imposing a more strict condition for fragment reallocation in distributed database. It results in fewer migrations of fragments from one site to

other sites over the network. However, the volume threshold, the number of time intervals and duration are the most important factors that regulate the frequency of fragment reallocations.

The authors of [5] propose a new dynamic data allocation algorithm for non-replicated DDBS named Performance Optimality Enhancement Algorithm (POEA). This works explores and improves some concepts used in previously developed algorithms to reallocates fragments to different sites given the changing data access patterns, time, and sites constraints of the DDBS. When the migration decisions are made it adopts the shortest path between the old location and the new anticipated location for the transferred fragments. The POEA algorithm is the most efficient one among all previous algorithms for dynamic data allocation as, it has certainly improves the DDBS performance by further minimizing network traffic. It also reduces data transmission cost compared to the previous methods, since it adopts the shortest path algorithm once data movement decisions are taken. The only drawback of the POEA algorithm is that it requires more storage compared to some previous algorithms. However, this is compensated by DDBS performance enhancement and is considered as a very trivial drawback due to the dramatic fall down in the storage hardware prices.

The authors in [28] solve the fragment allocation problem by using the well-known Quadratic Assignment Problem solution algorithms. The authors of [21] propose a new technique for horizontal fragmentations of the relations of distributed databases. This technique can be applied at the initial stage as well as in later stages of DDBS for partitioning the relations. The authors of [20] address some important scalability issues. They provide some algorithms to ensure generality of the technique developed in [21].

A new vertical fragmentation, allocation and replication scheme of a distributed database called (VFAR) was proposed by our previous work in [7]. The proposed scheme partitions the distributed database relations vertically at the initial stage of the database design by using the enhanced minimum spanning tree (MST) Prim's algorithm. In addition, it allocates and replicates the resulted fragments to the sites that require it.

The authors of [4] propose a heuristic technique to satisfy horizontal fragmentations and allocations using a cost model to minimize the total cost of distribution. Furthermore, the authors of [9, 19] present a new framework for dynamic fragment allocation and replication. The authors of [19] consider replication and fragment correlation under a flexible network topology. This framework tackles multiple issues in this system, including lazy replication strategy, fragments' correlation, on the-fly fragment allocation in the face of changing query access patterns, and non-uniform distances between network sites. In this framework the correlation between fragments is modeled and an algorithm to find near optimal dynamic allocation is presented. It also provides a simple methodology to update the allocation when access patterns change. The experiments demonstrate that this algorithm provides efficient solutions for the fragment allocation problem in distributed database systems. The future of this work is firstly, leveraging fragment ordering to further improve the performance of the algorithms. Secondly, generalize

reallocation by employing a mechanism that does data mining of the access patterns to detect and decide on a reallocation schedule. Finally, incorporate into the algorithms the ability to handle extra characteristics, such as bounds on the capacity of sites, constraints on the number of replicas for each fragment, and constraints on fragments that are not allowed to be replicated, e.g., due to access control or security constraints.

The author of [33] proposes two algorithms for dynamic data redistribution: Part of the redistribution (Partial Reallocate) and full redistribution (Full Reallocate) algorithm. The two algorithms are linear complexity and can be used in a variety of different sizes distributed database system. The authors of [25] propose a new dynamic data allocation algorithm for non-replicated distributed database system. This algorithm is called Threshold and Time Constraint Algorithm (TTCA), which is an extension of the optimal algorithm [12] and threshold algorithm [29]. The TTCA algorithm removes problems of threshold algorithm by adding time constraint to the existing threshold technique.

The authors of [8] introduced cluster based peer to peer architecture for distributed databases. The proposed architecture is named flexipeer. This work uses predicate based fragmentation of previous work done in [21]. It is used to address the fragmentation and allocation of database designs. The proposed work introduces a clustering approach for partitioning database sites. The clustering process is done based on the unique numbers of region sites. It also allocates fragments across the sites of each cluster. This paper tries to implement the concept of chord in peer to peer based data management. The sites of each cluster are managed by the local cluster administrator LCA and the whole architecture is managed by a global cluster administrator GCA. 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 work of [17] presents a novel algorithm for grouping distributed database network sites into disjoint clusters based on communication time. The authors see that performing clustering algorithms after fragmentation will speed up the process of data allocation by eliminating extra communication costs between sites. The proposed algorithm creates disjoint clusters according to the least average communication cost between network sites. It also distributes the DDBS sites over the clusters. In addition, it generates near optimal numbers of clusters required to achieve high network system performance. The results obtained from the simulation demonstrated the significant network server's load balance and network delay. The experimental outcomes confirmed that this approach can be implemented in different DDBS environments even if the network sites are enormous.

Another method of clustering the sites in which low communication cost sites are grouped in one cluster was proposed in [15]. 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.

The authors of [31] proposed a dynamic data replication strategy using historical access records and proactive deletions called Closest Access Greatest Weight with Proactive Deletion (CAGW_PD). The authors of [13, 24, 26, 32] believe that a replication method has three important issues to consider: When the new replica should be replicated, which file should be replicated, and where the new replica should be placed. However, the authors of [31] see that there is still one additional issue which should be resolved, i.e., how to control the number of replicas.

3 Research Issues

Based on the above survey, the following key findings are highlighted. First, no previous works handle dynamic fragmentation, allocation, and replication decisions of distributed database fragments and replica at cloud environment.

Second, no existing work takes the fragmentation, allocation, and replication decisions dynamically over a clustered distributed database sites. In spite of, performing clustering algorithm after fragmentation will speed up the process of data allocation by eliminate extra communication costs between sites.

Third, few existing works handle FAR decisions dynamically at run time. However, these works does not consider site constraints, load of the sites after allocate or migrate a fragment or replica to it, and the limit of the replica numbers into account in order to take the dynamic decisions.

4 Solutions and Recommendations

To overcome the limitations of existing literature highlighted by the above survey, this research proposes an enhanced DDBS design over a cloud environment. In our previous work in [6], a DDBS design over the cloud was introduced.

To extend this work, contributions in this proposed research includes adding new layers and modules to enhance the DDBS design and 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 a remote site. It also allows FAR decisions to be taken dynamically at run time. These decisions are based on access patterns and the 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 three layers: Application client layer, distributed database system manager layer and distributed database clusters layer.

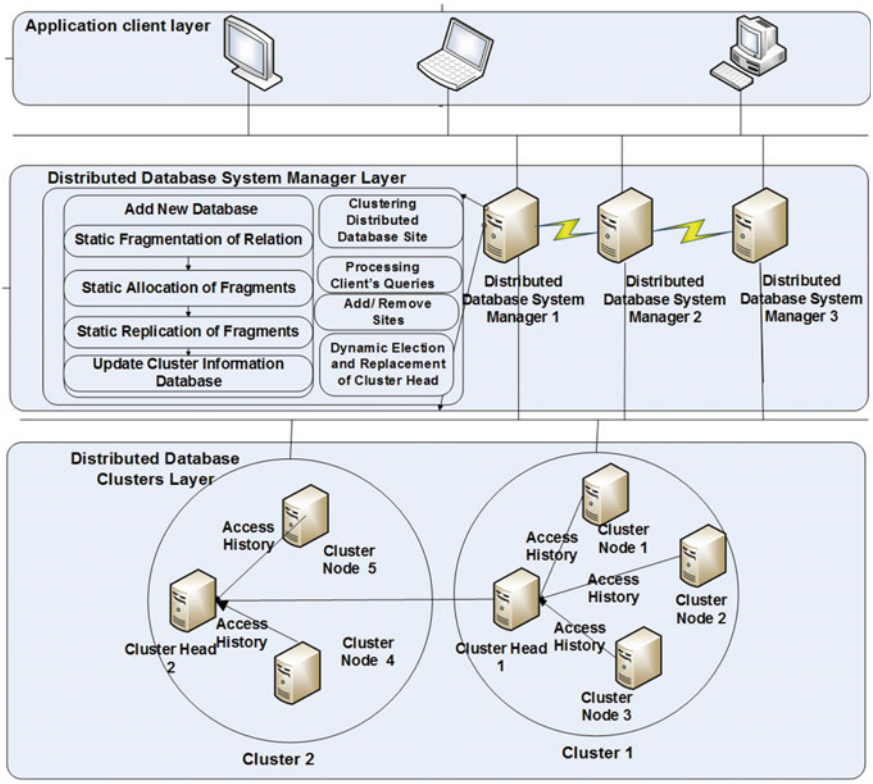

Fig. 1 A dynamic distributed database system over a cloud environment architecture

4.1 Application Client Layer

This layer consists of a set of interfaces which allow the users of distributed database systems to access the databases from anywhere. In order to join the distributed database system, first the client calls the distributed database system manager to get all the information about the structure and the location of the sites of the distributed system. Second, the client joins the distributed system by sending a query to the distributed database system manager.

4.2 Distributed Database System Manager Layer

This layer is composed of five modules: Add a new database, clustering distributed database sites, client queries processing, add/remove sites and dynamic election and replacement of the cluster head.

i. **Add a 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 the 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 and in which site or cluster.

ii. **Clustering Distributed Database Sites Module**

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

iii. **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 re-directs the user to that site to fetch the needed data.

iv. **Add/Remove Sites Module**

The distributed database system manager uses this module to add new sites to the distributed database system or remove existing sites from the distributed system. By using this module the distributed database system manager can expand or reduce the distributed database system without any affect to the client and other the sites of distributed database system.

v. **Dynamic Election and Replacement of the Cluster Head Module**

This module is used by the distributed database system manager to elect and replace the cluster head. A new cluster head is elected in two cases. In the first case the recent cluster head is removed. In the second case a new site is added to the cluster and its capacity is higher than the cluster head.

4.3 Distributed Database Clusters Layer

The distributed database cluster layers consist of more than one cluster. Each cluster has one cluster head and more than one cluster node.

i. **Cluster Node**

Each cluster node contains two local databases. The first local database is used to store the fragments and replicas of the distributed system. The second local database is used to store the information about the user's access to the stored fragments and replicas. At each access firstly, the cluster node checks whether it is a local access or remote.

Secondly, it allows the user to access a local database of the node to run the query and fetch the required data.

Thirdly, the local database of user's access is updated to save the information about the user access. Forth, cluster node runs query composition to answer the query after it fetches all needed fragments.

Finally, it sends the site access record to the distributed database system manager and also sends the results of the query to user. Each cluster node sends the access history to the cluster head to take any suitable decisions such as: Create or delete replicas, re-fragmentation or re-allocation of the fragments.

ii. **Cluster Head**

The cluster head is a cluster node and has special and additional operations to manage the other cluster nodes. The cluster head performs three operations: Manage cluster nodes data, load balancing of the cluster, dynamic fragmentation, replication and allocation.

1. Managing Cluster Node Data

Each cluster node sends the access history to the cluster head to take the decision to either create or delete replica, re-fragmentation or re-allocation of the fragments. The cluster head 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.

2. Load Balancing of the Cluster

Each cluster head contains load balancing algorithm that contains two services: Site capacity identifier and performance prediction services. The site capacity identifier service contains the capacity of each site in each cluster.

The capacity of the site contains two objects: The number of users that can access site at any time and the number of fragments and replicas that the site can hold.

The performance prediction service will be used to predict the load of the site after adding the fragments or replicas to it. The fragment or replica will be assigned to the site with less workload.

3. Dynamic Fragmentation, Replication and Allocation

The cluster head will take fragmentation, replication and allocation 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.

5 Enhanced Allocation and Replication Technique

The authors of [21] proposed a new technique of horizontal fragmentation. This technique helps in taking the fragmentation decision at the initial stage of designing the distributed database. It uses knowledge gathered during the requirement analysis phase by using the enhanced CRUD (Create, Read, Update, and Delete) matrix without the help of empirical data about query executions. This technique performs data allocation according to the maximum attribute locality precedence (ALP) value and the location 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 costs and time.

In our previous work in [6], we 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 this case we used 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 operations than other sites. For example, if we have two sites: Site 1 and Site 2. Site 1 performs CUD operations. Site 2 performs R operations. The replica will be sent to Site 2 although Site 1 has the maximum ALP single value. If the replica is sent to Site 1, 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.

6 Experimental Results

We have implemented our technique on an HP Compaq computer with Coretwo Duo 2.33 processors and 2 GB RAM using the SQL Server as DBMS. We have implemented the modified technique on the MCRUD matrix of the bank account table as 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 fragmentations. 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 Site 1 and fragment2 has been allocated to Site 2 because they have the maximum

```
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

Table 1 MCRUD matrix

	Site 1			Site 2			Site 3		
	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 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 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 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 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

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, Site 1 performs two read operations and Site 3 performs three read operations. As a result, fragment 3 will be allocated to Site 1 and its replica will be sent to Site 3.

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

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 Site 3, replica 2 of fragment 2 is allocated to Site 1 and replica 3 of fragment 3 is allocated to Site 3. The final results of the allocation and the replication processes are shown in Table 7.

7 Clustering Distributed Database Sites

An efficient way to minimize the communication time required for query processing and data allocation is grouping distributed database sites into disjoint clusters [17]. As a result, clustering distributed database sites is an important issue in distributed database systems.

The previous works done in this area are divided into two parts. In the first part are clusters distributed database sites based on communication costs between the network sites. In the second part are cluster distributed database sites based on the region fields of the database.

7.1 Clustering Distributed Database Sites Based on Region Fields

The simplest way to cluster the distributed database sites into disjoint clusters is by using the region field of the database [8]. The sites in the same region belong to the same cluster. However, clustering distributed database sites based on region has two disadvantages.

The first disadvantage, this methodology of clustering sites will not work on all the sites that belong to the same region or at most two regions. In this case we will have one cluster that contains all the sites in the distributed system.

The second disadvantage is that this methodology will not work in some cases belonging to different regions. In this case, each cluster will contain one site.

7.2 Clustering Distributed Database Sites Based on Communication Costs Between Network Sites

The second way to cluster distributed database sites is to use the communication cost between database sites [15, 17]. This work focuses on grouping distributed database sites into disjoint clusters according to the least average communication cost between network sites. The clustering process highly depends on the communication cost range of the CCR value.

The CCR parameter represents the communication cost value (ms/byte) that is allowed for the maximum difference between the sites to be grouped in the same cluster. If the communication cost between two sites is less than the CCR then the two sites are grouped on one cluster. The CCR value depends on how much time is allowed for the sites of the same cluster to receive or transmit their data.

The first advantage of methodology is that it can be implemented in different DDBS environments even if the network sites are enormous.

The second advantage is performing the clustering algorithm after fragmentation will speed up the process of data allocation by eliminate extra communication costs between sites [17].

Based on the advantages and disadvantages of the clustering techniques, we implemented the clustering technique mentioned in [17] that clusters the distributed database sites based on the communication costs.

The implemented method categorizes the distributed database sites according to Clustering Decision Value (CDV). The value of CDV is based on two values. The first value is communication cost range CCR value. The second value is the communication cost between the distributed database sites (CC). The communication cost between two sites (S_i, S_j) is defined as the following linear function:

$CC(S_i, S_j)$ = the cost of creating the data packet + the cost of transmitting the data packet from site S_i to site S_j

The cluster decision value (CDV) is logical value. It determines whether a pair of sites can be grouped on one cluster or not. The value of CDV is determined by Eq. 1.

$$CDV(Si, Sj) = \begin{cases} 1 \; if \; CC(Si, Sj) \leq CCR \\ 0 \; if \; CC(Si, Sj) > CCR \end{cases} \quad (1)$$

In this proposed research we run the implemented algorithm on distributed database system that consists of 10 sites. The communication cost between the sites is shown on Table 8. We assumed that the communication cost range value (CCR) equal to 5. After running the implemented algorithm, the resulted clusters and the sites assigned to it are shown in Table 9. More details about the implemented algorithm in [17]. However, we will use our enhanced technique to allocate and replicate the fragments to the sites of each cluster.

Table 8 Communication cost between sites

Site#	Site 0	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7	Site 8	Site 9
Site 0	0	6	11	10	7	8	9	9	12	12
Site 1	6	0	10	10	2	3	4	2	7	8
Site 2	11	10	0	6	6	7	8	7	2	3
Site 3	10	10	6	0	8	7	6	7	9	6
Site 4	7	2	6	8	0	1	2	2	7	6
Site *5*	8	3	7	7	1	0	I	2	8	8
Site 6	9	4	8	6	2	1	0	3	6	6
Site 7	9	2	7	7	2	2	3	0	8	7
Site 8	12	7	2	9	7	8	6	8	0	1
Site 9	12	8	3	6	6	8	6	7	I	0

Table 9 Resulted clusters and the assigned sites to it

Cluster #	Site #				
Cluster 0	Site 0				
Cluster 1	Site 3				
Cluster 2	Site 1	Site 4	Site 5	Site 6	Site 7
Cluster 3	Site 2		Site 8	Site 9	

8 Future Research Directions

As proposed future work, we plan to implement the remaining parts of the proposed architecture to efficiently allow FAR decisions to be taken automatically at run time. We plan to take the site constraints, load of the sites and the volume of data transmitted in a specific time interval from or to fragments into account to take the dynamic FAR decisions.

9 Conclusions

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 proposed research, we present an enhanced dynamic DDBS over a cloud environment. The proposed system allows FAR decisions to be taken based on access history and the load of the site. Moreover, this research presents enhanced allocation and replication techniques, which allocate 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 accesses compared to access from the remote sites, enhances the systems performance and

increases the availability. This research also presents different clustering techniques that used to cluster distributed database sites into disjoint clusters. It also presents the clustering technique advantages, disadvantages, and which one we will use in our system.

References

1. Abdalla, H.I.: An efficient approach for data placement in distributed systems. In: 2011 5th FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE). IEEE (2011)
2. Abdalla, H.I.: A new data re-allocation model for distributed database systems. Int. J. Database Theory Appl. **5**(2), 45–60 (2012)
3. Abdalla, H.I.: A synchronized design technique for efficient data distribution. Comput. Hum. Behav. **30**, 427–435 (2014)
4. Abdalla, H.I., Amer, A.A.: Dynamic horizontal fragmentation, replication and allocation model in DDBSs. In: 2012 International Conference on Information Technology and e-Services (ICITeS). IEEE (2012)
5. Abdalla, H.I., Amer, A.A., Mathkour, H.: Performance optimality enhancement algorithm in DDBS (POEA). Comput. Hum. Behav. **30**, 419–426 (2014)
6. Abdel Raouf, A.E., Badr, N.L., Tolba, M.F.: Dynamic distributed database over cloud environment. In: International Conference of Advanced Machine Learning Technologies and Applications. Springer, Berlin Heidelberg (2014)
7. Abdel Raouf, A.E., Badr, N.L., Tolba, M.F.: An optimized scheme for vertical fragmentation, allocation and replication of a distributed database. In: 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS). IEEE (2015)
8. Amalarethinam, D.G., Balakrishnan, C.: A Study on Performance Evaluation of Peer-to-Peer Distributed Databases. IOSR J. Eng. **2**(5), 1168–1176 (2012)
9. 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**(3), 1–8 (2013)
10. Basseda, R., Rahgozar, M., Lucas, C.: Fuzzy neighborhood allocation (FNA): a fuzzy approach to improve near neighborhood allocation in DDB. In: Advances in Computer Science and Engineering. Springer, Berlin (2009)
11. Bayati, A., Ghodsnia, P., Rahgozar, M., Basseda, R.: A novel way of determining the optimal location of a fragment in a DDBS: BGBR. In: ICSNC'06 International Conference on Systems and Networks Communications, 2006. IEEE (2006)
12. Brunstrom, A., Leutenegger, S.T., Simha, R.: Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads. In: Proceedings of the Fourth International Conference on Information and Knowledge Management. ACM (1995)
13. Chang, R.S., Chang, H.P.: A dynamic data replication strategy using access-weights in data grids. J. Supercomput. **45**(3), 277–295 (2008)
14. Corcoran, A.L., Hale, J.: A genetic algorithm for fragment allocation in a distributed database system. In: Proceedings of the 1994 ACM Symposium on Applied Computing. ACM (1994)
15. Suganya, A., Kalaiselvi, R.: Efficient fragmentation and allocation in distributed databases. Int. J. Eng. Res. Technol. **2**(1), 1–7 (2013)
16. Gope, D.C.: Dynamic data allocation methods in distributed database system. Am. Acad. Sch. Res. J. **4**(6), 1–8 (2012)
17. Hababeh, I.: Improving network systems performance by clustering distributed database sites. J. Supercomput. **59**(1), 249–267 (2012)

18. Hauglid, J.O., Ryeng, N.H., Nørvåg, K.: DYFRAM: dynamic fragmentation and replica management in distributed database systems. Distrib. Parallel Databases **28**(2–3), 157–185 (2010)
19. Kamali, S., Ghodsnia, P., Daudjee, K.: Dynamic data allocation with replication in distributed systems. In: 2011 IEEE 30th International Performance Computing and Communications Conference (IPCCC). IEEE (2011)
20. Khan, S.I., Hoque, A.L.: Scalability and performance analysis of CRUD matrix based fragmentation technique for distributed database. In: 2012 15th International Conference on Computer and Information Technology (ICCIT). IEEE (2012)
21. Khan, S.I., Hoque, A.S.M.L.: A new technique for database fragmentation in distributed systems. Int. J. Comput. Appl. **5**(9), 20–24 (2010)
22. Menon, S.: Allocating fragments in distributed databases. IEEE Trans. Parallel Distrib. Syst. **16**(7), 577–585 (2005)
23. Mukherjee, N.: Synthesis of non-replicated dynamic fragment allocation algorithm in distributed database systems. ACEEE Int. J. Inf. Technol. **10**, 154–159 (2011)
24. Ranganathan, K., Foster, I.: Identifying dynamic replication strategies for a high-performance data grid. In: Grid Computing—GRID 2001. Springer, Berlin (2001)
25. Singh, A., Kahlon, K.S.: Non-replicated dynamic data allocation in distributed database systems. IJCSNS Int. J. Comput. Sci. Netw. Secur. **9**(9), 176–180 (2009)
26. Tang, M., Lee, B.S., Yeo, C.K., Tang, X.: Dynamic replication algorithms for the multi-tier data grid. Futur. Gener. Comput. Syst. **21**(5), 775–790 (2005)
27. Basseda, R., Tasharofi, S., Rahgozar, M.: Near neighborhood allocation (NNA): a novel dynamic data allocation algorithm in DDB. In: 11 International CSI Computer Conference (CSICC'2006). School of Computer Science, IPM (2006)
28. Tosun, U., Dokeroglu, T., Cosar, A.: Heuristic algorithms for fragment allocation in a distributed database system. In: Computer and Information Sciences III. Springer, London (2013)
29. Ulus, T., Uysal, M.: Heuristic approach to dynamic data allocation in distributed database systems. Pak. J. Inf. Technol. **2**(3), 231–239 (2003)
30. Varghese, P.P., Gulyani, T.: Region based fragment allocation in non-replicated distributed database system. Int. J. Adv. Comput. Theory Eng. **1**(1), 62–70(2012)
31. 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**(1), 227–250 (2012)
32. Zhang, J., Lee, B.S., Tang, X., Yeo, C.K.: A model to predict the optimal performance of the hierarchical data grid. Futur. Gener. Comput. Syst. **26**(1), 1–11 (2010)
33. Zhenglong, L.: Study of dynamic data redistribution algorithm based on distributed database system. Procedia Engineering **15**, 5611–5615 (2011)

Additional Reading

34. Hababeh, I.O., Ramachandran, M., Bowring, N.: A high-performance computing method for data allocation in distributed database systems. J. Supercomput. **39**(1), 3–18 (2007)
35. Hababeh, I., Khalil, I., Khreishah, A.: Designing high performance web-based computing services to promote telemedicine database management system. IEEE Trans. Serv. Comput. **8** (1), 47–64 (2015)
36. Kumar, R., Gupta, N.: An extended approach to Non-Replicated dynamic fragment allocation in distributed database systems. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). IEEE (2014)
37. Mukherjee, N.: Non-replicated dynamic fragment allocation in distributed database systems. In: Advances in Computer Science and Information Technology. Springer, Berlin (2011)