An Enhanced Distributed Database Design Over the Cloud Environment

Ahmed E. Abdel Raouf(✉), Nagwa L. Badr, and M.F. 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. The design of a distributed database is one of the major research issues within the distributed database system area. The main challenges facing distributed database systems (DDBS) design are fragmentation, allocation and replication. In this paper, we present an enhanced distributed database design over the cloud environment. It minimizes the execution time needed for the transactions and for noticing an error when an invalid query or data cannot found occurs. It also allows users to access the distributed database from anywhere. Moreover, it allows fragmentation, allocation and replication decisions to be taken statically at the initial stage of designing the distributed database. Experimental results show that the proposed system design, results in a significant reduction of the execution time needed for the transactions to reach the data in an appropriate site and the time taken to notice an error when an invalid query or data not found occurs.

Keywords: Distributed database system (DDBS) · Distributed database management system (DDBMS) · Fragmentation · Replication · Allocation · Cloud computing

1 Introduction

A database that consists of very large amounts of data used by different applications at different physical locations needs efficient support. Large distributed enterprise databases, telecom databases and scientific databases are examples of application areas [5]. The delay of accessing remote databases is the main problem of many of these applications. As a result, it is necessary to use a distributed database that employs fragmentation, allocation and replication [5].

Distributed database systems (DDBS) typically consist of a number of distinct database fragments and its replicas which allocate and replicate at different geographic sites. The sites of DDBSs are managed by distributed database management systems (DDBMS) and can also communicate through a network. Each site has autonomous processing capability and also participates in the execution of at least one global database application, which requires accessing data residing at several different sites [1].

The major research issue in a distributed database system area is the design of the distributed database system. The main challenges facing the DDBS design are the following questions: Which type of fragmentation will be used to fragment the database relations and where to allocate, replicate the fragments and its replicas to the sites of the distributed database system to enhance system performance and increase availability.

The main contribution in this extended research is adding new layers with modules to our previous DDBS design in [1]. The new modules aim to minimize the execution time needed for the transactions to reach the data in an appropriate site and also minimize the required time to notice an error when an invalid query or data not found occurs. The proposed system design allows the distributed database system to be accessed from anywhere without owning any technology infrastructure. Moreover, it allows static fragmentation, allocation and replication decisions to be taken at the initial stage of designing the distributed database, without the help of empirical data about query executions. Experimental results show that the proposed system design, results in a significant reduction of the execution time needed for the transactions to reach the data in an appropriate site and the time taken to notice an error when an invalid query or data not found occurs.

The rest of this paper is organized as follows. Section 2 reviews the related work. The enhanced distributed database design over the cloud environment and its components proposed in Sect. 3. Section 4 presents the experimental results. Finally the conclusion and future work are presented in Sect. 5.

2 Related Work

A dynamic DDBS over the cloud environment was proposed by our previous work in [1]. Moreover, an enhanced allocation and replication technique was proposed, which allocates the fragments and its replicas to the sites that already requires it without taking into account the location of the sites. The enhanced allocation and replication technique aims at maximizing the number of local accesses compared to access from the remote sites, enhance the system performance and increase the availability.

An optimized scheme for vertical fragmentation, allocation and replication of a distributed database called (VFAR) was proposed by our previous work in [2]. The proposed VFAR 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. Moreover, it allocates and replicates the resulted fragments and its replicas to the sites that require it.

The authors of [4] propose a new technique for horizontal fragmentation of the distributed database relations. This technique is used for partitioning the relations at the initial stage as well as in later stages of DDBS. The authors of [6] provide some algorithms to ensure generality of the technique developed in [4] by addressing some important scalability issues. However, the allocation strategy of this technique [4] doesn't meet the goal of data allocation. The goal of 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.

A decentralized approach for dynamic fragmentation, allocation and replication in DDBS was proposed in [5]. It performs fragmentation, allocation and replication based on recent access history. However, the main drawback of this approach is that it doesn't consider the load of the sites after the process of allocation and replication, site constraints as well as the optimal number of replicas of each fragment to enhance the system's performance and increase availability.

Chord was designed to create a network that is decentralized, reliable and scalable [3]. It uses the consistent-hashing method to distribute hash keys to nodes. The nodes in chord are organized in a ring topology, in which each node in chord contains a finger table of its neighbors and their possible assignments of keys.

A cluster based peer to peer architecture for a distributed database named Flexipeer was proposed in [3]. They try to implement the concept of chord in peer-to-peer based data management. The proposed architecture in [3] contains a local cluster administrator (LCA) which manages the sites of each cluster. It also contains a global cluster administrator (GCA) which manages the whole architecture. This work uses the fragmentation technique of previous work done [4]. In addition, a clustering approach for partitioning database sites and allocating fragments across the sites of each cluster was proposed.

However, this approach [3] wastes a lot of time between nodes, LCA, LCA Validator, Resource Checker and GCA until it reaches the required data. Moreover, the allocation technique used in this work doesn't meet the goal of data allocation by allocating fragments to the sites that need it. In addition, it doesn't address the replication phase of database design.

The authors of [7] present a synchronized horizontal fragmentation, allocation and replication model. It performs horizontal fragmentation of distributed database relations based on the attribute retrieval and update frequency.

The works in [7] and [8] perform the allocation process based on the cost of moving the data fragments from one site to the other and the fragment access pattern. A new algorithm called the Region Based Fragment Allocation (RFA) was proposed in [9]. It considers the frequency of fragments accessed by region as well as individual nodes to move the fragment from the source node to target node. It decreases the fragments migration using its knowledge of the network topology in comparison to optimal [10] and threshold [11] algorithms. It also reduces the amount of topological data required in decision making in comparison to the BGBR [12] algorithm.

A model that takes site constraints into account in the process of re-allocation was presented in [13]. However, when information queries continuously change in a faster way and when the number of fragments and sites largely increase, this model will be more complicated.

A Near Neighborhood Allocation (NNA) algorithm which moves data to a neighborhood node that is placed in the path to the node with the maximum access counter was proposed in [14]. It could be of greater use in larger networks to decrease the delay of response times.

The authors of [15] propose an algorithm that dynamically re-allocates the fragments to sites of the distributed database system at runtime. It takes into account the time constraints, volume threshold and the volume of data transmitted in successive time intervals in accordance with the changing access patterns to dynamically re-allocate the fragments to the sites. However, the number of time intervals, their duration and the volume threshold are the most important factors that determine the frequency of fragment re-allocations.

The authors of [16] perform the re-allocation process given the time and sites constraints as well as changing the data access patterns of the DDBS. It adopts the shortest path algorithm once the data movement decision is taken to reduce data transmission costs compared to the previous methods. The drawback of this algorithm is the storage required compared to some previous algorithms.

The limitations of existing literature can be summarized in these points. Firstly, the earlier approaches that suggest a distributed database design wastes a lot of time between nodes to reach outsourced data in an appropriate site and to notice an error when an invalid query or data not found occurs.

Secondly, the proposed approaches do not consider the load of the sites after the process of allocation and replication, site constraints as well as the optimal number of replicas of each fragment to enhance the system's performance and increase availability.

Finally, the allocation strategy that's used at the initial stage of distributed database design 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 addition, it doesn't address the replication phase of the distributed database at the initial stage of designing distributed databases.

3 The Enhanced Distributed Database Design Over Cloud Environment Architectures

To overcome the limitations of existing literature highlighted by the above survey, this research proposes an enhanced DDBS design over the cloud environment. In our previous work in [1] a DDBS design over the cloud was introduced. To extend this work, the main contribution in this extended research is adding new layers with modules to our previous DDBS design in [1]. The new modules aim to minimize the execution time needed for the transactions to reach the data in an appropriate site and also minimize the required time to notice an error when an invalid query or data not found occurs. The proposed design allows distributed database systems to be accessed from anywhere without the need to own any technology infrastructure. The proposed system can be accessed through: A web browser, mobile application or desktop application while the database is stored on servers at remote sites. In addition, it allows fragmentation, allocation and replication decisions to be taken statically at the initial stage of designing the distributed database, without the help of empirical data regarding query executions.

The proposed design is shown in Fig. 1 and consists of two layers: Distributed database system manager and distributed database cluster layers.

3.1 Distributed Database System Manager (DDBSM)

The distributed database system manager (DDBSM) is used to manage the distributed database systems. It has special operations to firstly, partition the distributed database

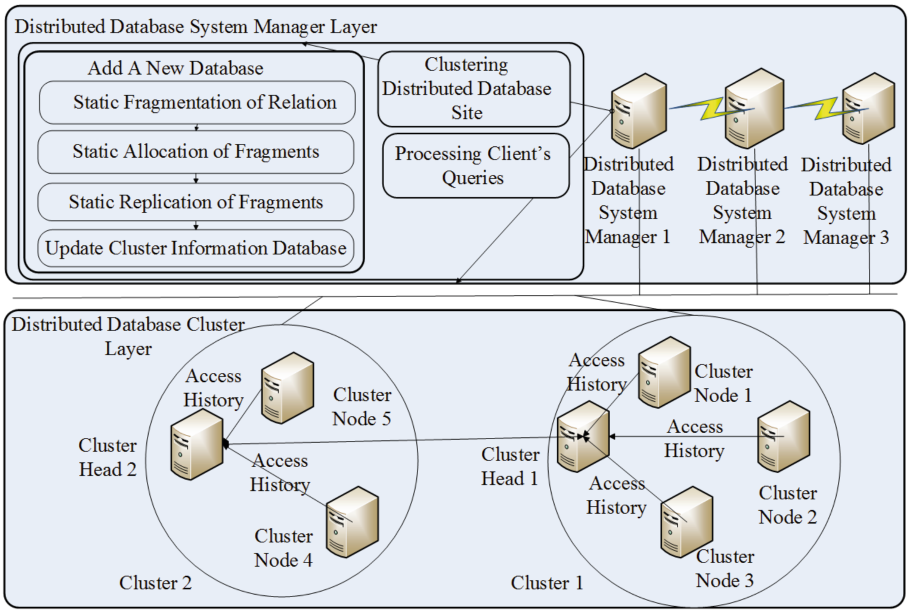
Distributed Database System Manager Layer
Add A New Database
Static Fragmentation of Relation
Static Allocation of Fragments
Static Replication of Fragments
Update Cluster Information Database
Clustering Distributed Database Site
Processing Client's Queries
Distributed Database System Manager 1
Distributed Database System Manager 2
Distributed Database System Manager 3
Distributed Database Cluster Layer
Access History
Cluster Node 5
Cluster Head 2
Access History
Cluster Node 4
Cluster 2
Access History
Cluster Node 1
Access History
Cluster Head 1
Cluster Node 2
Access History
Cluster Node 3
Cluster 1

Fig. 1. The enhanced distributed database design

relation into fragments and also allocates and replicates the resulted fragments to the sites. Secondly, it clusters the distributed database sites into disjoint clusters and finally, processing the client queries. This layer consist of three modules: Adding a new database, clustering distributed database sites and processing client queries.

Add New Database. This module is used to add the database tables to DDBS. Firstly, the new tables are fragmented, allocated and replicated to the sites of the distributed database using the static fragmentation, allocation and replication technique proposed by our previous work in [1], which can be used at the initial stage of the DDBS design using the enhanced CRUD (Create, Read, Update and Delete) matrix, without the help of empirical data about query executions and also without taking into account the location of the sites.

Secondly, the cluster information database is updated to save the locations of each fragment and replica in the distributed database system. The Cluster Information Database is a database that holds complete information about each fragment or replica in which site or cluster.

Clustering Distributed Database Sites. This module is used to cluster the distributed database sites into disjoint clusters. In addition, it also allocates sites to each cluster. After that it updates the cluster information database with the recent location of each site in DDBS.

Client Queries Processing. This module is used for processing the client queries. When the client sends a query to the DDBSM, firstly the distributed database systems manager checks the syntax of the query, if the query doesn't satisfy the syntax, then the DDBSM sends a message to the client saying the query is invalid.

However, if the query satisfies the syntax, then the DDBSM uses the cluster information database to determine the location of the sites that hold the required data based on the type of query. If the locations of the data are found, then the DDBSM sends the locations of the site that holds the data to the clients.

After that, the client uses the information about the communication costs between the sites of the DDBS to determine the closest site that holds the data. Finally, the query is sent to the selected site and the results are sent from the site to the client. However, if the locations of the data aren't found in the cluster information database then, the DDBSM sends a message to the client that the data was not found. The flow diagram of the query processing is shown in Fig. 2.

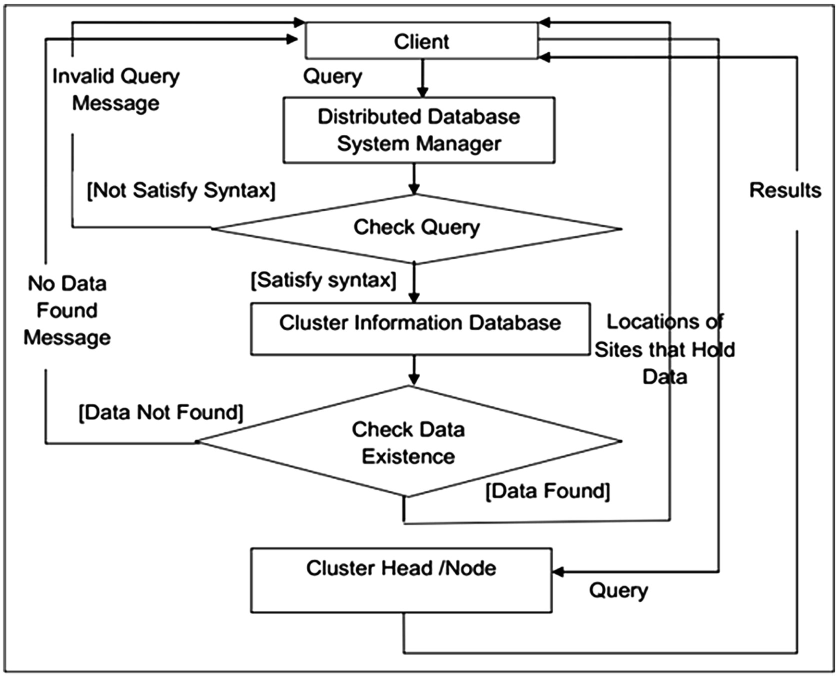

Fig. 2. Flow diagram of query processing

3.2 Distributed Database Clusters Layers

The distributed database clusters layer consists of more than one cluster. Each cluster consists of one cluster head and more than one cluster node. The cluster head is the same as a cluster node; however it has especial and additional operations to manage the other cluster nodes.

At each access to the cluster node, firstly, it checks whether it is a local access or remote. Secondly, it allows the user to access the database of the node to run the query and fetch the required data. Finally, the node access record is updated to save the information about the user's access.

4 The Experimental Results

The performance of the proposed enhanced distributed database design is studied in a simulated environment. The simulated environment consists of three HP Compaq computers with Core-two Duo 2.10 processors and 2 GB RAM using SQL Server as DBMS. This simulation is intended to evaluate the performance of the proposed enhanced distributed database design in execution of transactions reaching the data in an appropriate site and the time taken to notice an error when an invalid query or the data not found occurs.

The execution time and error message indication time is measured in milliseconds. We evaluate the results of the proposed enhanced distributed database design against Chord and FlexiPeer architectures in [3].

The evaluation results are shown in Fig. 3. It describes the evaluation results of the average times taken to execute a transaction in the proposed enhanced distributed database design, Chord and FlexiPeer [3] when the client is at the same site of the data or when the client is at a different site to that of the data. It also describes the evaluation results of the average time taken to indicate errors in the proposed enhanced distributed database design, Chord and FlexiPeer.

The main contribution of the proposed enhanced distributed database design is that it solves all mentioned drawbacks of the clustered approach proposed in [3] to minimize communication costs and enhance the DDBSs performance. The proposed

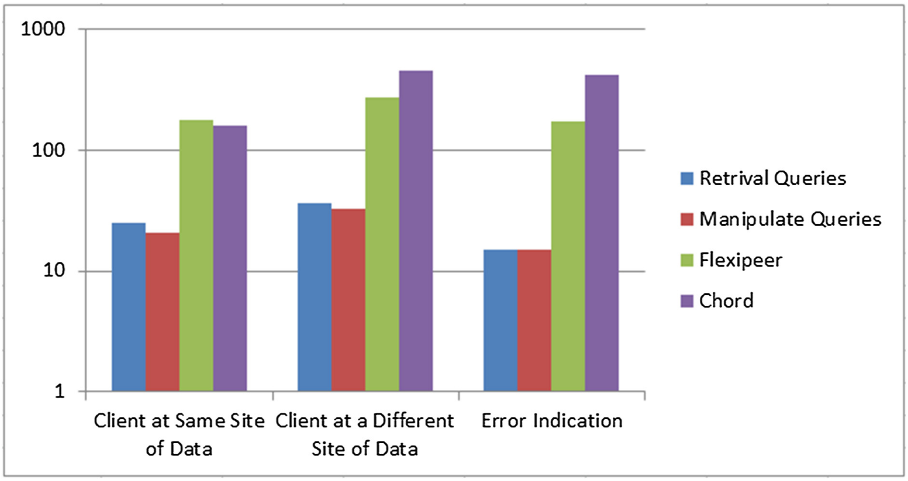

Fig. 3. Evaluation results

enhanced distributed database design firstly, uses our previous enhanced fragmentation, allocation and replication technique proposed in [1], which takes the fragmentation, allocation and replication decisions at the initial stage of designing the distributed database. It is knowledge based on the requirement analysis phase by using the enhanced CRUD (Create, Read, Update and Delete) matrix without help of empirical data regarding query executions and without taking into account the location of the sites. It also aims to maximize the number of local accesses compared to remote accesses in order to enhance the systems performance and increases the availability. Secondly, it results in a significant reduction of the execution times needed for the transactions to reach the data in an appropriate site and the time taken to notice an error when an invalid query or data not found occurs.

5 Conclusion and Future Work

Improving the performance, increasing the availability of data and access facility are the main motivation for distributed databases. The efficient distribution of fragments and its replicas to the sites of the distributed database system play the critical role of the performance and the cost of a distributed database system. In this paper, we present an enhanced distributed database design over a cloud environment. The proposed system design allows distributed database systems to be accessed from anywhere. In addition, it allows fragmentation, allocation and replication decisions to be taken statically at the initial stage of designing the distributed database, without the help of empirical data about query executions. Experimental results show that the enhanced distributed database design, results in a significant reduction of the execution time needed for the transactions to reach the data in an appropriate site and the time taken to notice an error when an invalid query or data not found occurs. As proposed future work, firstly, we plan to use an enhanced clustering technique to cluster distributed database sites into disjoint clusters. Secondly, we plan to implement the remaining parts of the proposed architecture to efficiently allow fragmentation, allocation and replication decisions to be taken automatically at run time.

References

1. Abdel Raouf, A.E., Badr, N.L., Tolba, M.F.: Dynamic distributed database over cloud environment. In: Hassanien, A.E., Tolba, M.F., Taher Azar, A. (eds.) AMLTA 2014. CCIS, vol. 488, pp. 67–76. Springer, Heidelberg (2014). doi:10.1007/978-3-319-13461-1_8
2. 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), pp. 506–513. IEEE, Cairo (2015)
3. Amalarethinam, D., Balakrishnan, C.: A study on performance evaluation of peer-to-peer distributed databases. IOSR J. Eng. **2**, 1168–1176 (2012)
4. Khan, S., Hoque, A.: A new technique for database fragmentation in distributed systems. Int. J. Comput. Appl. **5**, 20–24 (2010)

5. 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)
6. 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)
7. Abdalla, H.I.: A synchronized design technique for efficient data distribution. Comput. Hum. Behav. **30**, 427–435 (2014)
8. Abdalla, H.I.: An efficient approach for data placement in distributed systems. In: 2011 Fifth FTRA International Conference Multimedia and Ubiquitous Engineering, pp. 297–301. IEEE, Loutraki (2011)
9. 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)
10. Corcoran, L.: A genetic algorithm for fragment allocation a distributed database system. In: Proceedings 1994 ACM Symposium on Applied Computing, SAC 1994, pp. 247–250. ACM, USA (1994)
11. Ulus, T., Uysal, M.: Heuristic approach to dynamic data allocation in distributed database systems. Inf. Technol. J. **2**, 231–239 (2003)
12. Bayati, A., Ghodsnia, P.: A novel way of determining the optimal location of a fragment in a DDBS: BGBR. In: Systems Networks, pp. 64–69. IEEE Computer Society, Washington (2006)
13. Abdalla, H.: A new data re-allocation model for distributed database systems. Int. J. Database Theory **5**, 45–60 (2012)
14. Gope, D.: Dynamic data allocation methods in distributed database system. Am. Acad. Sch. Res. J. **4**, 1–8 (2012)
15. Mukherjee, N.: Synthesis of non-replicated dynamic fragment allocation algorithm in distributed database systems. Int. J. Inf. Technol. **1**, 36–41 (2011)
16. Abdallaha, H.I., Amer, A.A., Mathkour, H.: Performance optimality enhancement algorithm in DDBS (POEA). Comput. Hum. Behav. **30**, 419–426 (2014)