Journal of Software Engineering and Applications, 2014, 7, 891-905
Published Online October 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.711080

A New Approach for Database Fragmentation and Allocation to Improve the Distributed Database Management System Performance

Rizik M. H. Al-Sayyed[1], Fawaz A. Al Zaghoul[1], Dima Suleiman[1], Mariam Itriq[1], Ismail Hababeh[2]

[1]King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
[2]Faculty of Computer Engineering and Information Technology, German Jordanian University, Amman, Jordan
Email: r.alsayyed@ju.edu.jo, fawaz@ju.edu.jo@ju.edu.jo, dimah_1999@yahoo.com, m.itriq@ju.edu.jo, Ismail.hababeh@gju.edu.jo

Received 10 August 2014; revised 5 September 2014; accepted 1 October 2014

Abstract

The efficiency and performance of Distributed Database Management Systems (DDBMS) is mainly measured by its proper design and by network communication cost between sites. Fragmentation and distribution of data are the major design issues of the DDBMS. In this paper, we propose new approach that integrates both fragmentation and data allocation in one strategy based on high performance clustering technique and transaction processing cost functions. This new approach achieves efficiently and effectively the objectives of data fragmentation, data allocation and network sites clustering. The approach splits the data relations into pair-wise disjoint fragments and determine whether each fragment has to be allocated or not in the network sites, where allocation benefit outweighs the cost depending on high performance clustering technique. To show the performance of the proposed approach, we performed experimental studies on real database application at different networks connectivity. The obtained results proved to achieve minimum total data transaction costs between different sites, reduced the amount of redundant data to be accessed between these sites and improved the overall DDBMS performance.

Keywords

Distributed Database Management System, Fragmentation, Allocation, Clustering, Network Sites

1. Introduction

Many researches have been done in the last few years in order to improve the performance of distributed database management systems (DDBMS). DDBMS is a collection of logically interrelated data that are physically allocated at different locations over a computer network [1]. Most of recent researches are concerned about keeping the performance of DDBMS high so that they focused on how to design the database such that the overall cost is kept minimal. Keeping the cost minimal is not an easy task as there are huge amount of transactions processing that increase the complexity of distributed databases. Several techniques have been proposed in order to improve database performance which can be achieved by improving at least one of the following database management issues: database fragmentation, data allocation and replication, and the network sites clustering.

The fragmentation process divides the database into portions each of which is called a fragment. Fragmentation can be horizontal, vertical or mixed. The main advantage of fragmentation is to improve the performance of distributed database design by increasing the efficiency since data is stored only where it is needed. Fragments can be allocated at different network sites in a process called data allocation.

The fragments allocation is an NP-complete problem so that the complexity is high. In order to reduce the complexity, some heuristic algorithms have been proposed to solve the problem. In the allocation process, each fragment is assigned to a network node and sometimes to more than one node to achieve the data availability, system reliability and performance.

The clustering technique is used for grouping distributed database network sites into logical clusters. In order to reduce the communication time for data allocation, there are many algorithms that are use to find the optimal solution for grouping distributed database network sites into a disjoint clusters and making a better data distribution among them. The clustering technique aims at eliminating the extra communication costs between the network sites and then enhancing the DDBMS performance.

Many existing algorithms of data fragmentation and allocation in DDBMS assume some restrictions on the number of network sites so that the results of such algorithms are impractical, and reflected on the efficiency and validity of their outcomes. Moreover, some constraints on network connectivity and transactions processing time will limit the applicability of the proposed solutions to a small number of DDBMS cases [1].

One of the drawbacks of fragmentation and allocation solutions is the high computational complexity of their associated algorithms. In fact, when distributing a database over a network with a big number of sites and then finding an efficient, reliable and optimal solution for fragmentation and allocation are considered difficult tasks.

This paper proposed a new technique that splits the database relations into disjoint fragments. In addition, it introduces a high speed clustering technique that groups the distributed network sites into a set of clusters according to their communication cost. Also, it proposes a new intelligent technique for data allocation and redistribution based on transactions processing cost functions. Moreover, it implements a user-friendly simulation tool that performs fragmentation, clustering, allocation and replication of a database, in addition to assisting database administrators in measuring DDBMS performance.

The rest of this paper is organized as follows. Section 2 summarizes related work. A method for partitioning the database is presented in Section 3. A description of network site clustering is covered in Section 4. In Section 5, fragments allocation to the clusters and then to their sites is discussed. Performance evaluation is presented in Section 6. Finally, the conclusions and the future work are presented in Section 7.

2. Related Work

Many studies have been published on attempts of improving the performance of DDBMS. These researches have mostly investigated fragmentation, allocation and sometimes clustering problems. In this section, we present the main contributions related to these problems, discuss them and highlight novelties of our proposed solutions with respect to fragmentation, allocation and clustering.

There are many types of fragmentation [2] vertical, horizontal and mixed. In vertical fragmentation each schema is divided into small fragments and all fragments must contain a common candidate key. A new algorithm; as proposed by [3]; has employed vertical fragmentation for object relational database system; this fragmentation method depends on user input at different sites.

In [4], the authors provide a solution for initial fragmentation problem of relational database for DDBMS. It partitioned the relational database properly at the initial stage when no data statistics and query execution frequencies are available.

In order to minimize communication time required for data allocation and query processing, many researches have been done to group large number of network sites into a small number of logical clusters which improve the performance of a DDBMS by increasing transactions response time [5].

The authors of [6] present a new formulation for the problem of fragmentation and allocating those fragments with minimum cost for both structured and unstructured data, by grouping sites which are nearer to each other into one cluster, hence they have low cost. Also, a dynamic clustering method is adopted for both structured and unstructured database to reduce the movement of data between sites.

In a DDBMS, the cost of communication is high so that many researches tried to minimize the cost and load sharing by making load balancing which can be achieved by making analysis of sharing resources, allocation of fragments and transaction in DDBMS [7].

The complexity of a distributed database algorithm depends on the allocation method used. Some enhancements have been done in reallocation algorithms. The authors of [8] proposed an algorithm that reallocates fragments based on calculating the cost for each fragment individually. The reallocation depends on finding the maximum update cost value for each fragment. This technique takes into account the network topology and set of queries frequency values employed over the network.

Data allocation of a DDBMS can be done by means of mobile agents and many algorithms implemented to make optimization and solve allocation problems [9].

The authors of [10] present a biogeography-based optimization technique for no-replicated static allocation of data fragments during database design that minimize total data transmission cost during the execution of a set of queries.

3. Database Partitioning

Various methods already exist describe data fragmentation in distributed DDBMS. Naturally, there are benefits and drawbacks to all schemes. Some methods need to incorporate performance evaluation ways, may not minimize the transactions response time and cannot guarantee the ability to process a given portion of a given transaction in all sites [2] [3] and [4].

In our proposed approach, the database will be partitioned into pair-wise disjoint fragments by using a horizontal partitioning technique, in which the records of a relation split into disjoint fragments; this strategy guarantees the ability to processing all portions of a given transaction and distributes it precisely over the DDBMS sites.

Generating data fragments accomplished by performing the following processes respectively: defining transactions, creating segments and extracting disjoint fragments. **Figure 1** below describes the architecture of the fragmentation method that supports the use of knowledge extraction and helps to achieve the effective use of small data packets.

As shown in **Figure 1**, the data request is initiated from the DDBMS sites (Site1, Site2, and Site3) and defines transactions as queries (Query1, Query2, and Query3) and then these transactions (queries) are processed into disjoint fragments (Fragment1, Fragment2, and Fragment3). The database transaction could be associated with

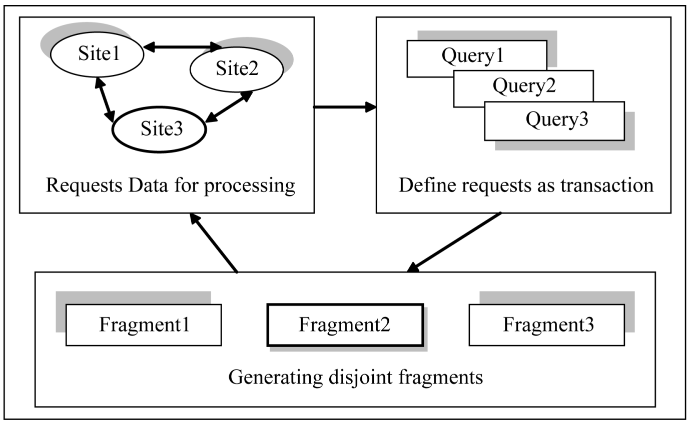

Site1
Site2
Site3
Requests Data for processing
Query1
Query2
Query3
Define requests as transaction
Fragment1
Fragment2
Fragment3
Generating disjoint fragments

Figure 1. Generating disjoint fragments.

more than one relation, in this case, the transaction should be divided into a number of sub-transactions equal to the number of relations used that transaction. The process of generating fragments is described as follows:

1) Database fragmentation starts with any two fragments having intersection records between them. If there is an intersection, then three disjoint fragments will be generated as follows:

- The common records in the two intersected fragments,
- The records in the first fragment but not in the second segment, and
- The records in the second fragment but not in the first segment.

2) The intersecting fragments are then removed from the fragments list. This process continues until removing all the intersecting fragments.

3) The new derived fragments and the non-overlapped ones that do not intersect with any other fragments from the new list of totally disjoint fragments.

We developed the fragmentation algorithm described in **Listing 1** to generate database disjoint fragments.

In the partitioning method, all transactions are processed, redundant transactions are eliminated, and the application speed and efficiency are improved by getting the minimum number of fragments to be accessed.

4. Network Sites Clustering

The benefit of generating database disjoint fragments can't be completed unless it enhances the performance of the distributed database system. As the number of database sites becomes too large, a problem of supporting high system performance with consistency and availability constraints becomes crucial. Different techniques could be developed for this purpose; one of them consists of clustering distributed networking sites. Clustering database sites is a technique in which the sites that have similar physical property (e.g., having comparable communication costs) are logically grouped together in order to increase the performance of the distributed database system. However, grouping sites into clusters is still an open problem and it is proven that the optimal solution to this problem is NP-Complete since it is transformed to a cheapest path problem. Therefore, near-optimal solution for grouping database sites into clusters helps to eliminate the extra communication costs between the sites during the process of data allocation and improves the system performance. Performing sites grouping after database fragmentation, will speed up the process of data allocation by distributing the fragments over clusters of sites rather than site by site. Thus, the communication costs are minimized and the distributed database system performance is improved.

A high speed clustering technique based on the least average communication cost between sites will be introduced. This is suitable for distributed databases where the communication costs between two sites are equal or near-equal, and similar computers on the network are used. The clustering parameters that will be used to control the input/output computations for generating clusters and determining the set of site(s) in each cluster are described and computed as follows:

- The Clustering Decision Value (*cdv*) that determines whether or not a site can be grouped in a specific cluster.

$$cdv\left(S_i, S_j\right) = \begin{cases} 1: \text{IF} \quad CC\left(S_i, S_j\right) <= CCR \wedge i \neq j \\ 0: \text{IF} \quad CC\left(S_i, S_j\right) > CCR \vee i = j \end{cases} \tag{1}$$

Accordingly, if $cdv(S_i, S_j)$ is equal to 1, then the sites S_i and S_j are assigned to one cluster, otherwise they are assigned to different clusters. Each site should be included in only one cluster, and the final distribution of the network sites will be represented by the cluster that satisfies the least average communication cost between the sites.

- The Communication Cost between the sites $CC(S_i, S_j)$. This is the cost of loading and processing database fragment(s) in ms/byte between any two sites in the distributed database system.
  $CC(S_i, S_j)$ = cost of creation the data packet + cost of transmitting the data packet between S_i, S_j
- The <u>C</u>ommunication <u>C</u>ost <u>R</u>ange *CCR* (ms/byte) that the site should match to be grouped in a specific cluster. This value is determined by the network administrators and depends on how much time is allowed for the sites to transmit or receive their data to be considered in the same cluster.

Based on the parameters, assumptions, and computations described above, we developed the following clustering algorithm illustrated in **Listing 2** to classify the networks clusters and their respective sites.

Listing 1. Fragmentation algorithm.

```
Input: K: Number of the last fragment
Rmax: Number of database relations
Nmax: Number of fragments in each relation
Step 1: Set 0 to K
Step 2: Set 1 to R
Step 3: Do steps (4 - 21) until R > Rmax
Step 4: Set 1 to I
Step 5: Do steps (6 - 20) until I > Nmax
Step 6: Set 1 to J
Step 7: Do steps (8-18) until J > Nmax
Step 8: If $I \neq J$ and  $\exists S_i, S_j \in S_R$ go to step (9)
Else Add 1 to J, go to step (18)
Step 9: If $S_i \cap S_j \neq \emptyset$ do steps (10)-(17)
        Else, Add 1 to J and go to step (19)
Step 10: Add 1 to K
Step 11: Create new fragment $F_k = S_i \cap S_j$ and add it to F
Step 12: Create new fragment $F_{k+1} = S_i - F_k$ and add it to F
Step 13: Create new fragment $F_{k+2} = S_j - F_k$ and add it to F
Step 14: Delete $S_i$
Step 15: Delete $S_j$
Step 16: Set Nmax + 1 to J
Step 17: End IF
Step 18: End IF
Step 19: Loop
Step 20: Add 1 to I
Step 21: Loop
Step 22: Set 1 to I
Step 23 Do steps (24 - 35) until I > Nmax
Step 24: Set 1 to J
Step 25: Do steps (26 - 33) until J > Nmax
Step 26: If $I \neq J$ and  $\exists S_i, S_j \in S_R$ go to step (27)
         Else Add 1 to J, go to step (33)
Step 27: If $S_i \cap S_j = \emptyset$ do steps (28)-(33)
Step 28: Add 1 to K
Step 29: Create new fragment $F_k = R_j - UF$
Step 30: End IF
Step 31: If $F_k \neq \emptyset$ Add $F_k$ to the set of F
Step 32: End IF
Step 33: Loop
Step 34: Add 1 to I
Step 35: Loop
Step 36: Set 1 to I
Step 37: Do steps (38 - 53) until I > F
Step 38: Set 1 to J
Step 39: Do steps (40 - 51) until J > F
Step 40: If $I \neq J$ and  $\exists F_i, F_j \in F_R$ go to step (41)
         Else, Add 1 to J and go to step (50)
Step 41: If $F_i \cap F_j \neq \emptyset$ do steps (42)-(49)
         Else, Add 1 to J and go to step (49)
Step 42: Add 1 to K
Step 43: Create new fragment $F_k = F_i \cap F_j$ and add it to F
Step 44: Create new fragment $F_{k+1} = F_i - F_k$ and add it to F
Step 45: Create new fragment $F_{k+2} = F_j - F_k$ and add it to F
Step 46: Delete $F_i$
Step 47: Delete $F_j$
Step 48: Set F + 1 to J
Step 49: End IF
Step 50: End IF
Step 51: Loop
Step 52: Add 1 to I
Step 53: Loop
Step 54: Add 1 to R
Step 55: Loop
```

Listing 2. Clustering algorithm.

```
Input: CC($S_i$, $S_j$): Matrix of communication cost between sites
CR: Clustering Range
NS: Number of sites in the distributed database system network
Output: CSM: Clusters Set Matrix
Step 1: Set 1 to i
Step 2: Do steps (3 - 12) until i > NS
Step 3: Set 1 to j
          Set 0 to k
          Set 0 to Sum
          Set 0 to Average
          Set 0 to clusters matrix CM
Step 4: Do steps (5 - 10) until j > NS
Step 5: If $i \neq$ j AND CC($S_i$, $S_j$) <= CR, go to step (6)
          Else, go to step (7)
Step 6: Set 1 to the CM($S_i$, $S_j$) and CM($S_j$, $S_i$) in the clusters matrix
          Add CC($S_i$, $S_j$) to sum
          Add 1 to k
          Go to step 8
Step 7: Set 0 to the CM($S_i$, $S_j$) and CM($S_j$, $S_i$) in the clusters matrix
Step 8: End IF
Step 9: Add 1 to j
Step 10: Loop
Step 11: Average = Sum/k
           Average(i) = Average
           Add 1 to i
Step 12: Loop
Step 13: Set 1 to m
Step 14: Do steps (15 - 36) until m > NS
Step 15: Set 1 to q
           Set 0 to Minaverage
           Set 0 to Minrow
Step 16: Do steps (17 - 20) until q > NS or Minaverage > 0
Step 17: If Average(q) > 0 Then
              Minaverage = Average(q)
           Else
              Go to Step 18
Step 18: End If
Step 19: Add 1 to q
Step 20: Loop
Step 21: If Minaverage = 0 Then
              Set site number to a new cluster
           Else
              Go to Step 22
Step 22: End If
Step 23: Set 1 to p
Step 24: Do steps (25 - 28) until p > NS
Step 25: If Average(p) > 0 AND Average(p) < Minaverage Then
              Minaverage = Average(p)
              Minrow = p
Step 26: End IF
Step 27: Add 1 to p
Step 28: Loop
Step 29: Set 1 to a
Step 30: Do steps (31 - 34) until a > NS
Step 31: If CM($S_{minrow}$, $S_a$) = 1 Then
              Set 1 to CSM($S_{minrow}$, $S_a$)
              CM($S_{minrow}$, $S_a$) = 0
Step 32: End IF
Step 33: Add 1 to a
Step 34: Loop
Step 35: Add 1 to m
Step 36: Loop
Step 37: Stop
```

Clustering Example

To demonstrate the applicability of clustering, a simulation of the clustering algorithm on the given communication cost between 12 sites will take place at clustering range CR equals to 5 (CR could be any communication cost as multiples of 0.125 ms/byte). The cluster set matrix resulted as shown in **Table 1**.

From the cluster set matrix, it is clear that only S1 can be grouped in the first cluster because the other sites can't match the cluster range along with S1. The sites S2, S5, S6, S7, and S8 are also grouped in the second cluster because their communication costs match the cluster range between them. Since the cluster range match the communication costs between S3, S9, S10 and the other sites are far from them, then these three sites can be only grouped in the third cluster. In the same way, the fourth cluster is constructed only from sites S4, S11, and S12. **Table 2** displays the generated clusters and their respective sites.

The communication costs within and between clusters have to be taken into consideration in the computation of the fragment allocation. The optimal way to find the communication cost between clusters and between sites within each cluster is to find the shortest path between the clusters/sites and compute the communication cost between them. However, this way is considered to be NP complete, therefore, the cluster average communication cost with symmetric communication cost between clusters is suitable for the computations of fragments allocation in many distributed database environments where similar computers with similar communication costs between sites are used.

Table 1. Cluster set matrix.

Site #	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
S1	1	0	0	0	0	0	0	0	0	0	0	0
S2	0	1	0	0	1	1	1	1	0	0	0	0
S3	0	0	1	0	0	0	0	0	1	1	0	0
S4	0	0	0	1	0	0	0	0	0	0	1	1
S5	0	1	0	0	1	1	1	1	0	0	0	0
S6	0	1	0	0	1	1	1	1	0	0	0	0
S7	0	1	0	0	1	1	1	1	0	0	0	0
S8	0	1	0	0	1	1	1	1	0	0	0	0
S9	0	0	1	0	0	0	0	0	1	1	0	0
S10	0	0	1	0	0	0	0	0	1	1	0	0
S11	0	0	0	1	0	0	0	0	0	0	1	1
S12	0	0	0	1	0	0	0	0	0	0	1	1

Table 2. The generated clusters and their sites.

Cluster #	Cluster Sites											
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
C1	✓	-	-	-	-	-	-	-	-	-	-	-
C2	-	✓	-	-	✓	✓	✓	✓	-	-	-	-
C3	-	-	✓	-	-	-	-	-	✓	✓	-	-
C4	-	-	-	✓	-	-	-	-	-	-	✓	✓

5. Fragments Allocation

Data allocation and replication technique places and distributes the database fragments on the clusters and their respective sites. Initially fragments are allocated to the clusters that have transactions using that fragments. Allocating fragments to a small number of clusters instead of large number of sites will reduce the number of communications and therefore enhance the system performance. **Figure 2** illustrates the structure of data allocation and replication technique.

A heuristic fragment allocation and replication technique will be introduced to perform the processes of fragments allocation in the distributed database system. Initially, all fragments are subject for allocating to all clusters having transactions using these fragments at their sites. If the fragment shows positive allocation decision value (*i.e.* allocation benefit greater than or equal to zero) for a specific cluster, then the fragment is subject for allocating at each site in this cluster, otherwise the fragment is not allocated (cancelled) from this cluster. This step is repeated for each cluster in the distributed database system.

As a result of the previous step, the fragment that shows positive allocation decision value at any cluster is a candidate for allocating at all sites of this cluster. If the fragment shows positive allocation decision value at a site of cluster that already shows positive allocation decision value, then the fragment is allocated at this site, otherwise, the fragment is not allocated to this site. This step is repeated for each site at this cluster.

To ensure data availability in the distributed database system, each fragment should be allocated to at least one cluster and one site. In case a fragment shows negative allocation decision value at all clusters, the fragment is allocated to the cluster that holds the least average communication cost and then to the site that has the least communication cost in this cluster.

5.1. Allocation Cost Functions

The allocation cost functions identifies the allocation status which is computed as a logical value for the comparison between the cost of remote access the fragment to the cluster/site and the cost of allocating the fragment to the cluster/site. If the cost of remote access the fragment to the cluster/site is greater than or equals to the cost of allocating the fragment to the cluster/site, then the allocation status is positive and the fragment is allocated to the cluster/site. On the other hand, if the cost of remote access is less than the cost of allocating, then the allocation status is negative and the fragment is cancelled from the cluster/site.

5.1.1. Cost of Allocating Fragments

The cost of allocating the fragment F_i issued by the transaction T_k to the cluster C_j, identified as $CA(T_k, F_i, C_j)$, is defined in terms of the following costs:

- Cost of Local Retrievals issued by the transaction T_k to the fragment F_i at cluster C_j. It is computed as the multiplication of the average cost of local retrievals for all sites (m) at cluster C_j times the average number of frequency of local retrievals issued by the transaction T_k to the fragment F_i for all sites at cluster C_j.

$$CLR(T_k,F_i,C_j)=\left(\frac{\sum_{q=1}^{m} CLR(T_k,F_i,C_j,S_q)}{m}\right)*\left(\frac{\sum_{q=1}^{m} FREQLR(T_k,F_i,C_j,S_q)}{m}\right) \quad (2)$$

- Cost of Local Updates issued by the transaction T_k to the fragment F_i at cluster C_j. This cost is computed as the result of the product of the average cost of local updates for all sites (m) at cluster C_j times the average frequency of local updates.

$$CLU(T_k,F_i,C_j)=\left(\frac{\sum_{q=1}^{m} CLU(T_k,F_i,C_j,S_q)}{m}\right)*\left(\frac{\sum_{q=1}^{m} FREQLU(T_k,F_i,C_j,S_q)}{m}\right) \quad (3)$$

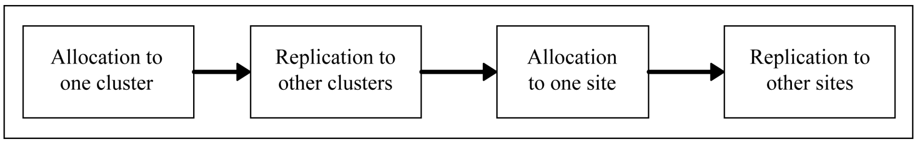

Allocation to one cluster
Replication to other clusters
Allocation to one site
Replication to other sites

Figure 2. Data allocation and replication technique.

- Cost of Space occupied by the fragment F_i in the cluster C_j. The space is computed as a result of multiplication of the average space cost for all sites (m) at cluster C_j occupied by the fragment F_i times the fragment size.

$$SCP\left(T_k, F_i, C_j\right) = \left(\frac{\sum_{q=1}^{m} SCP\left(T_k, F_i, C_j, S_q\right)}{m}\right) * F_{size}\left(T_k, F_i\right) \tag{4}$$

- Cost of Remote Updates issued by the transaction T_k to the fragment F_i sent from all clusters (n) in the distributed database system except the current cluster C_j. This remote cost is computed as the result of the product of cost of local updates (Equation (3)) times the average frequency of remote update.

$$CRU\left(T_k, F_i, C_j\right) = \left(\sum_{q=1, q\neq j}^{m} CLU\left(T_k, F_i, C_q\right)\right) * \left(\sum_{q=1, q\neq j}^{m}\left(\sum_{\alpha=1}^{m} FREQRU\left(T_k, F_i, C_j, S_q\right)/m\right)\right) \tag{5}$$

- Update ratio which determines the percentage of update transactions.

$$U_{ratio} = \left(\frac{\sum_{q=1}^{m} FREQLU\left(T_k, F_i, C_j, S_q\right)}{\sum_{q=1}^{m} FREQLR\left(T_k, F_i, C_j, S_q\right) + \sum_{q=1}^{m} FREQLU\left(T_k, F_i, C_j, S_q\right)}\right) \tag{6}$$

- Cost of Remote Communications issued for the transaction T_k to the fragment F_i sent from all clusters (n) in the distributed database system except the current cluster C_j. This cost is computed as the result of multiplication of the average cost of remote communication times the average frequency of local update times update ratio.

$$\begin{aligned} CRC\left(T_k, F_i, C_j\right) = & \sum_{q=1, q\neq j}^{n}\left(\sum_{\alpha=1,\beta=1,\alpha\neq\beta}^{m} CRC\left(T_k, F_i, C_j, \left(S_\alpha, S_\beta\right)\right)/m\right) \\ & * \left(\frac{\sum_{q=1}^{m} FREQLU\left(T_k, F_i, C_j, S_q\right)}{m}\right) * U_{ratio} \end{aligned} \tag{7}$$

- According to the computations in the previous cost Equations (2)-(7), the Cost of Allocating a fragment to a cluster is computed as the sum of costs of local retrievals, local updates, space, remote update, and remote communication.

$$\begin{aligned} CA\left(T_k, F_i, C_j\right) = & CLR\left(T_k, F_i, C_j\right) + CLU\left(T_k, F_i, C_j\right) + SCP\left(T_k, F_i, C_j\right) \\ & + CRU\left(T_k, F_i, C_j\right) + CRC\left(T_k, F_i, C_j\right) \end{aligned} \tag{8}$$

5.1.2. Cost of Not Allocating Fragments

The cost of not allocating the fragment F_i issued by the transaction T_k to the cluster C_j, identified as $CN(T_k, F_i, C_j)$, is defined in terms of the following costs:

- Cost of local retrievals issued by the transaction T_k to the fragment F_i at cluster C_j as computed in Equation (2).
- Retrieval ratio which determines the percentage of retrieval transactions.

$$R_{\text{ratio}} = \left(\frac{\sum_{q=1}^{m} FREQLR\left(T_k, F_i, C_j, S_q\right)}{\sum_{q=1}^{m} FREQLR\left(T_k, F_i, C_j, S_q\right) + \sum_{q=1}^{m} FREQLU\left(T_k, F_i, C_j, S_q\right)}\right) \tag{9}$$

- Cost of Remote Retrievals issued by the transaction T_k to the fragment F_i sent from all clusters (n) in the distributed database system except the current cluster C_j. This cost is computed as the product of the cost of communications between all clusters times the frequency of remote retrieval times the retrieval ratio

$$\begin{aligned} CRR\left(T_k, F_i, C_j\right) = & \sum_{q=1}^{n}\left(\sum_{\alpha=1,\beta=1,\alpha\neq\beta}^{m} CCC\left(T_k, F_i, C_j, \left(S_\alpha, S_\beta\right)\right)/m\right) \\ & * \left(\sum_{q=1, q\neq l}^{n}\left(\sum_{\alpha=1}^{m} FREQRR\left(T_k, F_i, C_j, S_q\right)/m\right)\right) * U_{\text{ratio}} \end{aligned} \tag{10}$$

- According to the computations in the previous cost equations (2,9,10), the Cost of Not allocating a fragment to a cluster is computed as follows:

$$CN\left(T_k, F_i, C_j\right) = CLR\left(T_k, F_i, C_j\right) + CRR\left(T_k, F_i, C_j\right) \tag{11}$$

5.1.3. Cost of Allocation Decision Value

The Allocation Decision Value ADV determines the allocation status of the fragment at the cluster and its respective sites. ADV is computed as a logical value resulted from the comparison between the cost of not allocating the fragment to the cluster $CN(T_k, F_i, C_j)$ and the cost of allocating the fragment to the cluster $CA(T_k, F_i, C_j)$.

$$ADV\left(T_k, F_i, C_j\right) = \begin{cases} 1; & CN\left(T_k, F_i, C_j\right) >= CA\left(T_k, F_i, C_j\right) \\ 0; & CN\left(T_k, F_i, C_j\right) > CA\left(T_k, F_i, C_j\right) \end{cases} \tag{12}$$

Accordingly, if $CN(T_k, F_i, C_j)$ is greater than or equal to $CA(T_k, F_i, C_j)$, then the allocation status $ADV(T_k, F_i, C_j)$ shows positive allocation benefit and the fragment is allocated to the cluster. On the other hand, if $CN(T_k, F_i, C_j)$ is less than $CA(T_k, F_i, C_j)$, then the allocation status $ADV(T_k, F_i, C_j)$ shows negative allocation benefit and the fragment is cancelled (not allocated) from the cluster.

Based on the fragment allocation procedures and the generated equations described above, we developed the fragment allocation algorithm described in **Listing 3** to allocate database fragments in the network clusters and their respective sites.

Once the fragments are allocated and replicated at the clusters of distributed database systems, then studying the benefit of allocating the fragments at all sites of each cluster becomes crucial. To get better system performance, fragments should be distributed over the sites of clusters where a clear allocation benefit is accomplished. The same allocation and replication algorithm will be applied in this case, taking to consideration the costs for/between sites in each cluster.

6. Performance Evaluation

We believe that the performance of allocation and network load are the major performance issues; hence, we will focused our performance evaluation on these two issues as they will maximize the overall system throughput at each network site in the distributed database environment, and minimize the cost of data that has been transferred and processed.

6.1. Fragment Allocation Performance Evaluation

The technique of fragment allocation and replication at clusters is evaluated according to the performance generated by reducing the size of fragments that allocated finally at the clusters. The closest methods in the literature to the proposed technique of fragment allocation and replication are those proposed in [7]-[10]. The main differences between these two methods are described as follows.

Compared to the allocation techniques in the literature, our allocation and replication technique considers different communication costs, namely, the cost of the real network communication between cluster sites, the update and retrieval costs (*i.e.*, it mostly represents the cost of writing operation that takes place during the execution time). Moreover, with our clustering technique, database sites are grouped according to a clustering range and not only to a specific communication cost. Our clusters can communicate with each other instead of preferring to have all fragments in their sites. This communication is cheaper than allocating fragments in all sites. In addition to the gain in term of communication cost, the independency of our clusters makes our DDBMS more reliable and more functional.

Figure 3 illustrates the effectiveness of the proposed fragment allocation and replication technique in terms of average computation time over the other two fragment allocation methods.

This figure shows that the proposed fragment allocation and replication technique results in the least average computation time required for fragment allocation. This is because the clustering technique produces groups (clusters) of small number of sites, and then the fragments will be reduced to the related fragments at each cluster. From **Figure 3**, it can be inferred that the average computation time for fragment allocation increases as the

Listing 3. Allocation algorithm.

```
Input: Tmax: number of transactions issued in the database
Fmax: number of the disjoint fragments used for allocation
Cmax: number of clusters in the distributed database system
Step 1: Set 1 to k
Step 2: Do steps (3 - 32) until k > Tmax
Step 3: Set 1 to i
Step 4: Do steps (5 - 30) until i > Fmax
Step 5: Set False to allocation flag
Step 6: Set 1 to j
Step 7: Do steps (8 - 26) until j > Cmax
Step 8: initialize Cost of Remote Update (0) to 0
        initialize Cost of Remote Communication (0) to 0
        initialize Cost of Remote Retrieval (0) to 0
Step 9: Set 1 to y
Step 10:Do steps (11 - 17) until y > Cmax
Step 11: If y ≠ j Then
         Do steps (12-15)
           Else
         Go to step (15)
Step 12: Cost of Remote Update(y) = Cost of Remote Update(y -1)
         + (Cost of Local Update * Average Frequency of Remote Update)
Step 13: Cost of Remote Communication(y) = Cost of Remote Communication(y -1)
         + (Average Cost of Remote Communication * Average Frequency of Local Update * $U_{ratio}$)
Step 14: Cost of Remote Retrieval(y) = Cost of Remote Retrieval(y -1)
         + (Cost of Communication between Clusters * Frequency of Remote Retrieval * $R_{ratio}$)
Step 15: End If
Step 16: Add 1 to y
Step 17: Loop
Step 18: Cost of Allocation = Cost of Local Retrieval + Cost of Local Update + Cost of Storage
                             + Cost of Remote Update + Cost of Remote Communication
Step 19: Cost of Not allocating = Cost of Local Retrieval + Cost of Remote Retrieval
Step 20: Allocation Decision Value = (Cost of Not allocating >= Cost of Allocation)
         # Comments: Allocation Decision Value is a Boolean value #
Step 21: If Allocation Decision Value = True Then
         Go to step (22)
           Else
         Go to step (23)
Step 22: Allocate the fragment to the current cluster
         Set True to allocation flag
         Go to step (24)
Step 23: Cancel the fragment from the current cluster
Step 24: End If
Step 25: Add 1 to j
Step 26: Loop
Step 27: If allocation flag = False Then
         Allocate the fragment to the cluster who has the least communication cost
Step 28: End If
Step 29: Add 1 to i
Step 30: Loop
Step 31: Add 1 to k
Step 32: Loop
```

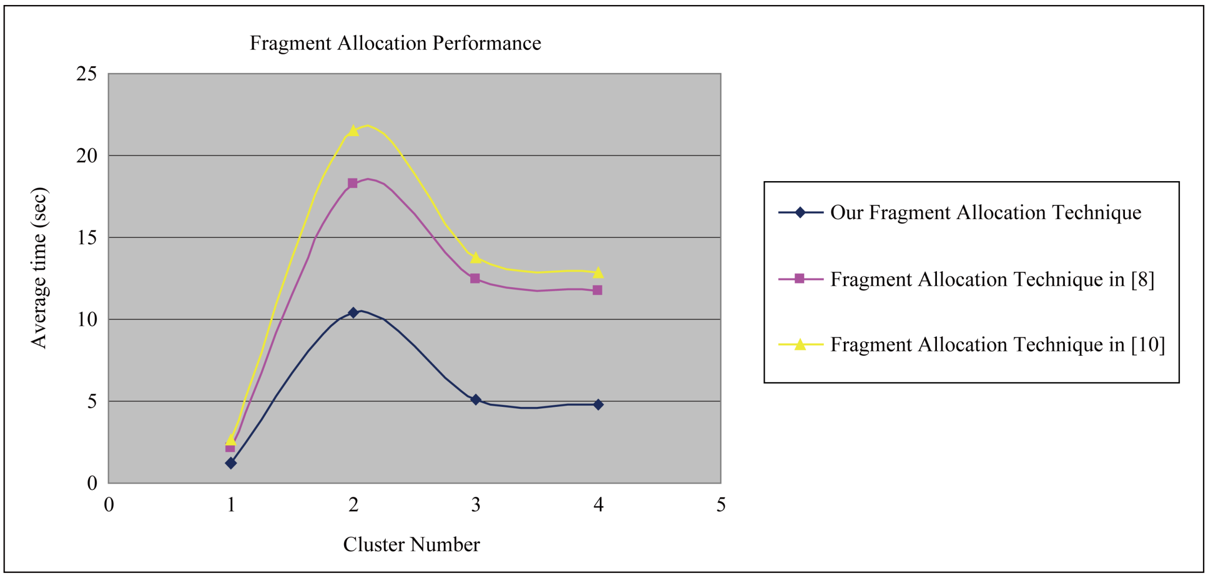


Figure 3. Fragments allocation after clustering.

number of sites in the cluster increases and vice versa. Besides, this figure depicts the fact that our fragment allocation and replication technique outperform the other two methods.

6.2. Network Performance Evaluation

The DDBMS network workload involves the queries from a number of database users who access the same database simultaneously. The amount of data needed per second and the time in which the queries should be processed depend on performance of the database applications running under DDBMS network. The DDBMS network system performance evaluation will be simulated by using the OPNET IT Guru Academic Edition 9.1 software [11].

Figure 4 shows the distribution of disjoint fragments over 4 clusters: C1 (node_12), C2 (node_5), C3 (node_7) and C4 (node_17). The first cluster (C1), combines node_18 and represents Site 1. The second cluster (C2) combines node_0, node_1, node_2, node_3 and node_4 and represents Sites 2, 5, 6, 7 and 8 respectively. The third cluster (C3) combines node_8, node_9 and node_10 and represents Sites 3, 9 and 10 respectively. The fourth cluster (C4) combines node_13, node_14 and node 15 and represents Sites 4, 11 and 12 respectively.

Figure 4 depicts the network topology for the proposed DDBMS should be evaluated. The figure depicts a simulation of building DDBMS and consists of 12 sites which are represented by nodes (0, 1, 2, 3, 4, 9, 10, 11, 16, 17, 18, and 20).

6.2.1. Server Load

The server load determines how much the speed of the server in terms of bits per seconds. The following figures depict the network server loads in the proposed DDBMS. The loads (bits/sec) on the DDBMS servers (C1, C2, C3, C4) are compared and illustrated in **Figure 5**.

This figure shows that at its peak, the load on server 2 (node 6) is well below 1300 bits/second, on server 3 (node 13) the load is well below 750 bits/second. The load on server 4 (node 14) is also well below 750 bits/second, and the load on server 1 (node 22) is well below 500 bits/second. The results of comparison states that the network is almost balanced throughout processing sites clustering.

6.2.2. Network Delay

The network delay is the delay caused by the transactions traffic on the server of the distributed database system. The maximum time required for the network system to reach the steady state is defined as network delay, it's measured in millisecond. **Figure 6** shows the network delay caused by each server (C1, C2, C3, C4).

The distributed database system reaches the steady state after 0.065 milliseconds and less network delay time

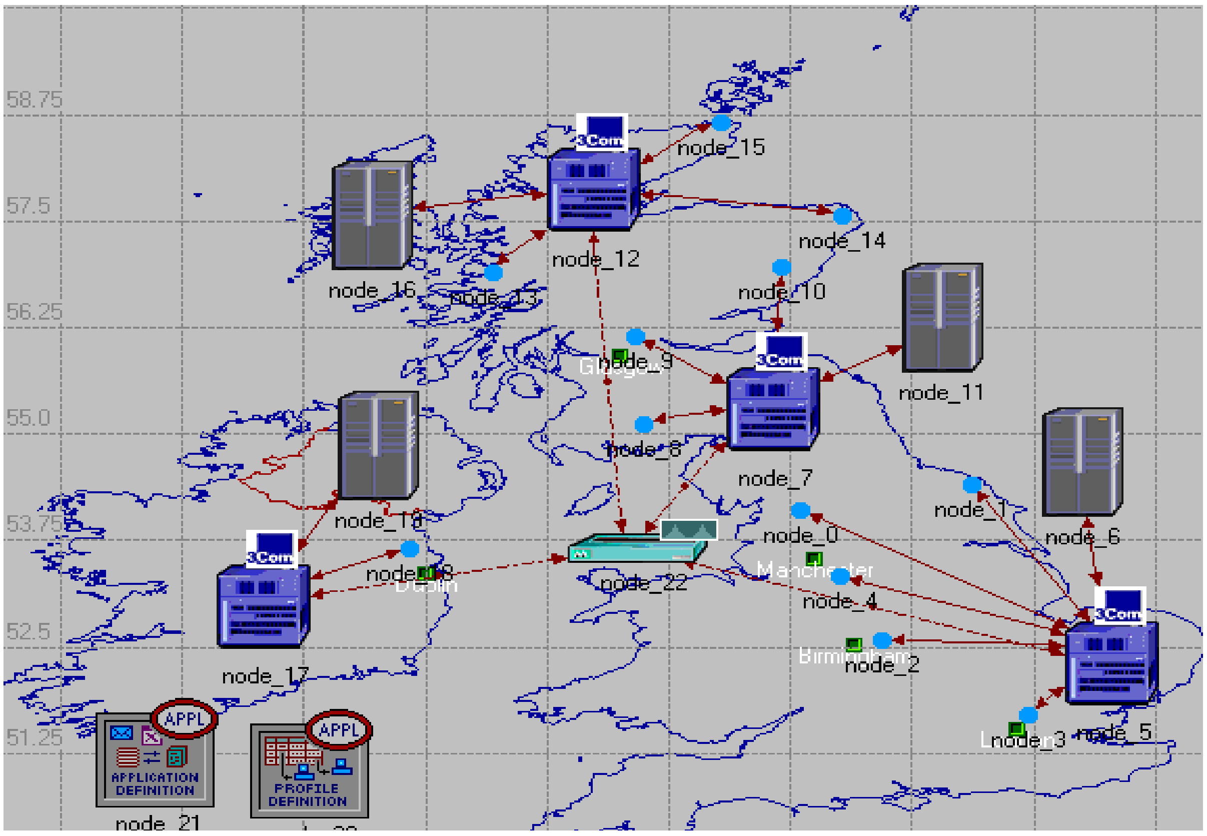


Figure 4. DDBMS network topology.

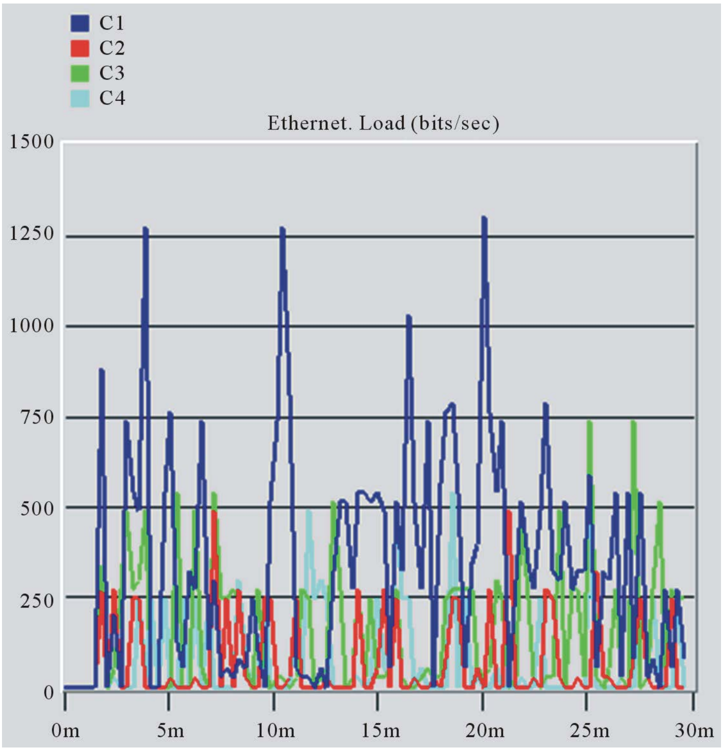


Figure 5. DDBMS network servers load.

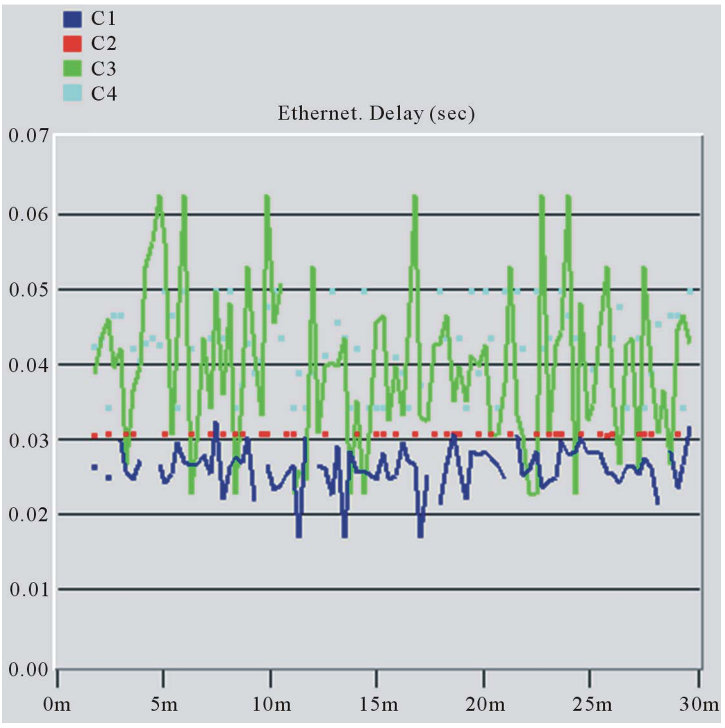


Figure 6. DDBMS network delay.

is consumed in case of distributing the sites over 4 servers compares to the delay time consumed in case of having all sites connected to 1 or 2 servers.

7. Conclusion and Future Work

In this work we proposed a new approach to improve DDBMS performance. This approach is integrating three enhanced techniques namely; database fragmentation, network sites clustering and fragments allocation. These techniques are developed to avoid drawbacks of database fragmentation and data allocation like data redundancy and complexity of data redistribution problem. In addition, a novelty of our approach is to satisfy a certain level of data availability and consistency. The results obtained in our evaluation showed that our approach significantly improved performance requirement satisfaction in distributed systems. This conclusion requires more investigation and experiments. Therefore, as future work we plan to investigate our approach on larger scale networks involving great number of database sites. Also, we will apply different types of clustering and use strong database sites grouping criteria. Finally, we plan to introduce search based technique to perform more intelligent data redistribution.

References

[1] Özsu, M.T. and Valduriez, P. (2011) Principles of Distributed Database Systems. Springer, Berlin.

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

[3] Gupta, S. and Panda, S. (2012) Vertical Fragmentation, Allocation and Re-Fragmentation in Distributed Object Relational Database Systems-(Update Queries Included). *International Journal of Engineering Research and Development*, **4**, 45-52.

[4] Khan, S.I. and Hoque, A.S.M.L. (2010) A New Technique for Database Fragmentation in Distributed Systems. *International Journal of Computer Applications*, **5**, 20-24. http://dx.doi.org/10.5120/940-1318

[5] Hababeh, I. (2012) Improving Network Systems Performance by Clustering Distributed Database Sites. *The Journal of Supercomputing*, **59**, 249-267. http://dx.doi.org/10.1007/s11227-010-0436-9

[6] Suganya, A. and Kalaiselvi, R. (2013) Efficient Fragmentation and Allocation in Distributed Database. *International*

Journal of Engineering Research & Technology (*IJERT*), **2**, 1-7.

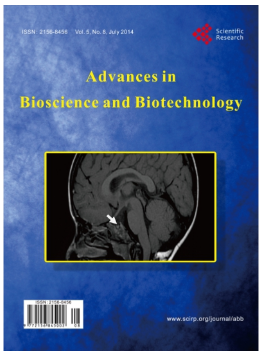
[7] Jagannatha, S., Geetha, D.E., Kumar, T.S. and Kanth, K.R. (2013) Load Balancing in Distributed Database System Using Resource Allocation Approach. *International Journal of Advanced Research in Computer and Communication Engineering*, **2**, 2529-2535.

[8] Abdalla, H.I. (2012) A New Data Re-Allocation Model for Distributed Database Systems. *International Journal of Database Theory and Application*, **5**, 45-60.

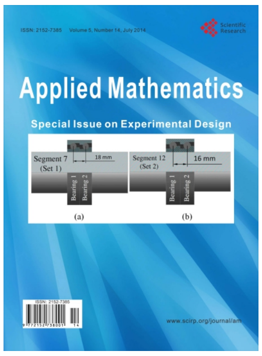
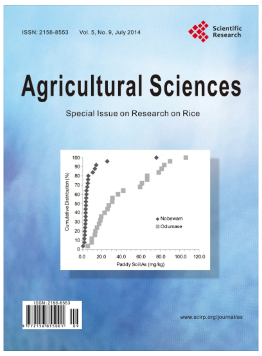
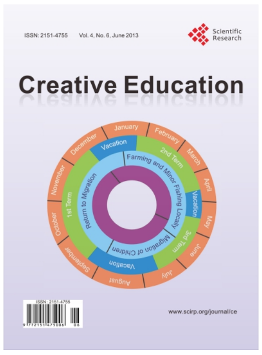
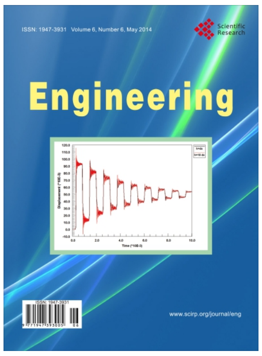
[9] Grebla, H., Moldovan, G., Darabant, S.A. and Câmpan, A. (2004) Data Allocation in Distributed Database Systems Performed by Mobile Intelligent Agents. *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics-ICTAMI*, Greece, 164-173.

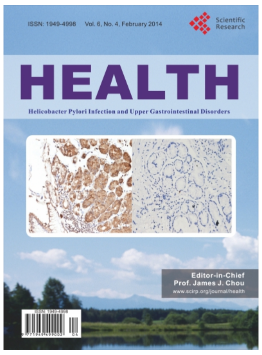
[10] Singh, A., Kahlon, K.S. and Virk, R.S. (2014) Nonreplicated Static Data Allocation in Distributed Databases Using Biogeography-Based Optimization. *Chinese Journal of Engineering*, 1-9.

[11] OPNET Technologies, Inc. (2014) OPNET IT Guru Academic Edition 9.1.
http://www.opnet.com/university_program/itguru_academic_edition

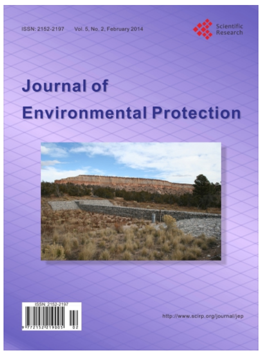
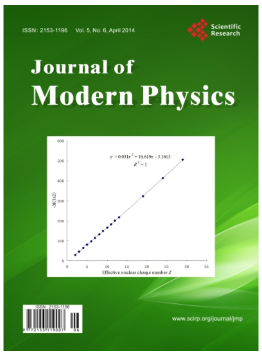

Advances in Bioscience and Biotechnology

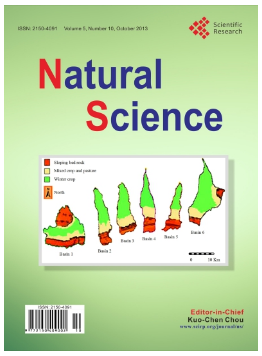
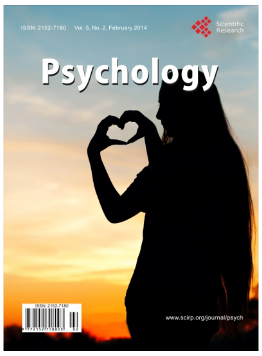
American Journal of Analytical Chemistry

American Journal of Plant Sciences

Applied Mathematics

Agricultural Sciences

Creative Education

Engineering

Food and Nutrition Sciences

HEALTH

Journal of Computer and Communications

Journal of Cancer Therapy

Journal of Environmental Protection

Journal of Modern Physics

Modern Economy

Natural Science

Psychology