A New Data Re-Allocation Model for Distributed Database Systems

Hassan I. Abdalla

Department of Computer Science, College of Computer & Information Sciences, King Saud University P.O Box 51178, Riyadh 11543, KSA.
habdalla@ksu.edu.sa

Abstract

An efficient functionality of any distributed database system "DDBS" is highly dependent on its proper design in terms of the adopted fragmentation and allocation methods. However, an optimal fragmentation and allocation design techniques could be very complex and requires good experiences and knowledge to reach. Fragmentation of a large, global databases are performed by dividing the database relations horizontally, vertically or as a combination of both. In order to enable distributed database systems to work efficiently, these fragments have to be allocated across the available sites in such a way that reduces communication cost i.e. to minimize the total volume of data transmitted during queries execution over sites. This paper presents a new data re-allocation model for replicated and non-replicated constrained DDBSs by bringing a change to data access pattern. This approach assumes that the distribution of fragments over network sites was initially performed according to a properly forecasted set of query frequency values that could be employed over sites. Our model takes sites constraints into account in the re-allocation phase. It proposes an efficient plan to re-allocate data fragments across sites based on communication and update cost values for each fragment individually. The re-allocation process will be performed by selecting the maximal update cost value for each fragment and making the re-allocation accordingly. Experimental results confirmed that the proposed technique will effectively contribute in solving fragments re-allocation problem in a dynamic distributed relational databases environment.

Keywords: *distributed database, fragment allocation, heuristic algorithm, redistribution algorithm*

1. Introduction

Fragmentation, replication and allocation problems are considered as an important design issues that have a great impact on *DDBS* performance and leads to optimal solutions particularly in a dynamic distributed environment.

Usually fragmentation is performed without having allocation in mind. While allocation, on the other hand, is always dependant on fragmentation and is performed on the assumption that fragmentation is obtained a priori.

Actually, four important issues should be taken into account in distributed database design. First, how global relation should be fragmented. Second, how many copies of a fragment should be replicated. Third, which technique will be used to allocate fragments to different sites of the network and what information for fragmentation and allocation need to be known.

Probability and heuristic methods are among the many different approaches that researchers have adopted to solve allocation problem. The probability method gives many possible scenarios to find an optimal solution but it is very costly. While the heuristic approach leads to near-optimal solution and rarely to an optimal solution. The results obtained by heuristic method determine the quality of this method. Our proposed model is considered as a heuristic approach rather than probability method although it somewhat combines the concepts of both.

An optimal allocation of fragments across the network sites is necessary for the better response time. Several solutions for solving the allocation problem are discussed in [21, 26, 15, 2, 1]. However, because of the complexity of this problem (*NP-complete*), a more heuristic algorithms were proposed in literature i.e. algorithms with a lower complexity that provide only an approximate solution.

One of the first works to study file allocation problem was considered in [8], where a general optimization model was developed to minimize the overall operating costs given time constraint, storage capacity and the number of copies for each file. *SAGA* approach was proposed in [1] to select the optimal places for data fragments. Data access pattern and transmission cost were used to help for specifying the optimal placement. However, [2] presented greedy approach where mathematical model for data allocation in *DDBSs* is proposed. It considered network communication, local processing and data storage costs to allocate a fragment site by site given the site capacity and fragment limit constraints.

A dynamic redistribution model was proposed in [17] that redistribute fragments with the objective of minimizing the transferred data. However, a more comprehensive approach to allocate fragments to minimize total data transfer cost was developed in [4]. On the other hand, [18] gave an integrated technique for fragmentation and allocation to obtain performance optimality. While a Lagrangian relaxation method is used for the same purpose in [7]. In [12] an integer programming formulation for the non-redundant version of the fragment allocation problem is presented. However, a high-performance computing method for data allocation in *DDBS* is used in [13]. The problem of distributing fragments of virtual *XML* repositories over the Web is considered in [5].

In most of the works described above, data allocation has been performed based on static allocation approaches that provide best solutions for environment where sites access probabilities to fragments never change. But, the static allocation solution would degrade the database performance in an ad-hoc environment, where these access probabilities change over time.

A framework for data redistribution on dynamic data allocation was presented in [6]. In [3], a dynamic data allocation algorithm for non-replicated database systems is proposed, but no modeling is done to analyze the algorithm. A dynamic data allocation and replication model for data redistribution was provided in [25 and 10]. In [16], an optimal algorithm for non-replicated database systems was developed. However, [23] proposed a threshold algorithm for non-replicated distributed databases where the fragments are continuously reallocated according to the changing data access patterns. In [14], a dynamic data allocation algorithm is proposed that reallocates data by changing data access patterns within a time constraint. A heuristic algorithm based on the ant colony optimization to minimize data transmission and allocation cost across network sites was presented in [11].

In this paper, a dynamic data reallocation model for replicated and non-replicated *DDBS* is proposed according to a given fragments Update Matrix *(UM)*, and Distance Cost Matrix *(DM)*. The costs of re-allocating fragment to the network site is evaluated and then the migration decision is made by selecting the site S_j that has the highest query update cost for fragment F_i to be chosen as the candidate site to store fragment F_i to minimize the communication cost.

This model uses Fragment Priority (*FP)* technique to avoid fragments duplication (that could take place at the initial allocation phase) and sites constraints violation whenever they occur.

2. Problem Statement And Information Requirements

2.1 Problem Description and Notations

The proposed model assumes that there is a fully connected network consisting of many sites $S = \{S_1, S_2, ..., S_m\}$, each site has capacity (C), fragment limit (FL) and a local database system. A link between two sites S_i and S_j has a positive integer (CC_{ij}) associated with them representing the cost of a unit data transfer from site S_i to site S_j. Each site has a set of queries $Q = \{Q_1, Q_2, ..., Q_k\}$ which are considered as the most frequently executed queries accounting for more than 75% of the site processing in *DDBS*. Each query Q_k can be executed from any site with a certain frequency, the execution frequencies of k queries at m sites can be represented by $m \times k$ matrix (QF_{ij}). Let S contain a set of fragments $F = \{F_1, F_2, ..., F_n\}$ representing a relation partitions during the fragmentation phase of *DDBS* design. To make the allocation more efficient we need to determine not only the number of copies for each fragment at every site, but also finding the proper allocation for each of them across sites according to the gathered query information.

There are two optimality definitions according to [24]: first; *minimal cost* (cost of storing each fragment F_i at site S_k, + cost of querying F_i at site S_k, + cost of updating F_i at sites where it is stored + the cost of data communication); second; the *performance* (minimizing response time and maximizing system throughput).

We have adopted the first (*minimal cost*) option in our proposed model for optimality measurement. Thus, the optimality problem according to our re-allocation model can be defined as the minimization of the communication cost in the process of re-allocating fragment F_i to site S_j.

The used notations in this paper are described next in table 1;

Table 1. Model Notations

M	Number of DDBS network sites
S_j	The jth site
N	Number of data fragments in DDBS
F_i	The ith data fragment
K	Number of DDBS queries
RN	Relations number
Q	Set of queries
Q_k	The kth query
$Q_j\#$	Queries number over site j
RF_{ij}	Access frequency value for retrieval query i at site j
UF_{ij}	Access frequency value for updating query i at site j
QF_{ki}	Access frequency of the kth query at site j
C_j	Capacity of site j
CC_{ij}	Communication cost between site i and site j
FL_i	Maximum number of fragments for site j
TC	The total cost

2.2 Information Requirements

To derive cost formulas such as cost functions, some information need to be analyzed in advance [20]; this could include: (1) database information, (2) query behavior information, (3) site information, and finally (4) network information. They will be described next.

2.2.1 Database Information: Usually, the relation consists of many fragments. Each of which has a size $Z(F_i)$) that need to be predefined as it plays a key role in computing the communication cost.

$$Z(F_i) = Card(F_i) * L(F_i), \quad i = 1, \ldots, n.$$

Where L is the length of fragment tuple and $card(F_i)$ is the fragment cardinality (i.e. number of fragment records), $length(F_i)$ can be calculated using the following formula:

$$Length(F_i) = \sum L_A, \ 1 <= A <= h.$$

Where h is the number of attributes for that fragment and A indicates to A^{th} attribute. So, $Z(F_i) = n * \sum L_j$, where n is the number of fragment records. To simplify our model we will assume the fragments sizes are given as shown in *table 2*.

Table 2. Fragments Sizes

Fragment	F1	F2	F3	F4	F5	F6
Size	25	45	32	60	36	57

We also need define the X_{ij} matrix that shows the assignment of fragments to sites. This matrix will be determined by the Retrieval Matrix (*RM*) and will be used at the initial allocation phase.

$$X_{ij} = \begin{cases} 1, & \text{if } F_i \text{ is assigned to } S \\ 0, & \text{otherwise} \end{cases}$$

2.2.2 Query Behavior Information: Since any query is either retrieving or updating data, then we define two access matrices; Retrieval Matrix *RM* (*table 3*) and Update Matrix *UM* (table 4), describing the retrieval and update behaviors respectively for all queries. The elements R_{kj} or U_{kj} in *RM* or *UM* respectively, gives the access frequency of fragment F_j by query k.

$$RM(R_{kj}) = \begin{cases} 1, & \text{if } Q_k \text{ retrieve } F_j \\ 0, & \text{otherwise} \end{cases}$$

$$UM(U_{kj}) = \begin{cases} 1, & \text{if } Q_k \text{ update } F_j \\ 0, & \text{otherwise} \end{cases}$$

Table 3. RM

Query / Fragment	F_1	F_2	F_3	F_4	F_5	F_6
Q_1	2	3	0	1	0	0
Q_2	2	0	0	0	1	1
Q_3	6	0	3	0	2	0
Q_4	3	0	2	0	0	0
Q_5	0	1	0	2	0	1

Table 4. UM

Query / Fragment	F_1	F_2	F_3	F_4	F_5	F_6
Q_1	0	0	2	0	1	0
Q_2	0	1	1	0	0	2
Q_3	3	0	0	2	0	1
Q_4	0	1	0	0	1	1
Q_5	0	0	2	0	1	0

For example in Table 3, *query* Q_1 retrieves *fragments* F_1 twice, F_2 three times and F_4 only once. It also, updates *fragments* F_3 twice and F_5 once in Table 4. However, for any query employed at any site, it should have frequency value in that site. Thus, we have to define Frequency Matrix for Queries *(QF)* that shows execution frequency of queries at each site (table 5 depicts this).

Table 5. Query Frequency

Query/Site	S_1	S_2	S_3	S_4
Q_1	0	2	3	1
Q_2	0	3	0	0
Q_3	2	0	1	0
Q_4	0	0	4	0
Q_5	1	1	0	2

2.2.3 Site Information: Site information is represented in our model by the site constraints, the system needs to comply with the site capacity *(C)* and fragment limit *(FL)* constraints that represent the maximum size and number of fragments that a site can handle. The constraints used to manage the allocation process are presented next;

$$\sum_{i=1}^{m} X_{ij} > = 1, 1 < = i < = n \qquad (1) \qquad // \text{ initial allocation phase}$$

$$\sum_{i=1}^{n} Q_{ij} * \text{size}(F_i) < = C_i, 1 < = j < = m \qquad (2)$$

$$\sum_{i=1}^{m} Q_{ij} < = FL_i, 1 < = i < = n \qquad (3)$$

$$\sum_{i=1}^{m} Q_{ij} = 1, 1 < = i < = n \qquad (4) \qquad // \text{ re-allocation phase}$$

Equation (1), indicates that each fragment *F* must be allocated to at least one site at the *initial allocation phase*. Equation (2), states that no site will receive more than its capacity. Equation (3), states that each site will not handle more than a given number of fragments (*FL*) and finally equation (4), states that a fragment will be allocated to only one site at the post allocation *(re-allocation) phase*. Table 6 depicts our model constraint limits.

Table 6. Network Sites with Constraints

Site	Capacity (C)	Fragment Limit (FL)
S_1	100	6
S_2	80	5
S_3	60	4
S_4	90	4

2.2.4 Network Information: In our proposed approach it is assumed that the *DDBS* network sites are fully connected network. And each link between sites S_i and S_j has a communication cost value (CC_{ij}) representing the cost of communication between them as shown in Figure 1.

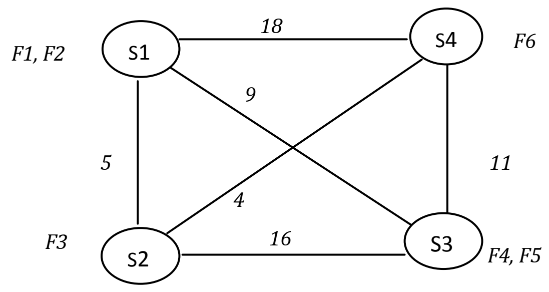

Figure 1. Network Sites

In order to simplify our model, the communication cost matrix is given as symmetric matrix, i.e. the communication cost between sites S_i and S_j is the same as that between sites S_j and S_i and the communication cost within the same site is zero as shown in *table 7*.

Table 7. Communication Cost Matrix

Sites	S1	S2	S3	S4
S1	0	5	9	18
S2	5	0	16	4
S3	9	16	0	11
S4	18	4	11	0

After applying the *minimum* algorithm of [20] on the communication cost matrix, we obtain the Distance Cost Matrix (*DM)* as shown in (table 8).

Table 8. Distance Cost Matrix (DM)

Sites	S1	S2	S3	S4
S1	0	5	9	9
S2	5	0	14	4
S3	9	14	0	11
S4	9	4	11	0

3. The Proposed Model and Cost Functions

In this section, our proposed model along with cost functions that will be used to calculate the cost of fragments re-allocation will be presented.

3.1 The Proposed Model

The proposed model assumes that there is a priori gathered information about the update frequency involved in the fragments allocation across sites. To properly accomplish the allocation process, each fragment must be assigned to one or more sites in the *DDBS* at *the initial allocation phase*. However, since the allocation problem involves finding optimal distribution of fragments across sites. Then for a given set of fragments with different sizes, and a number of network sites where each site has a number of executed queries, the optimal allocation model will involve minimizing the total cost of update in the entire network without violating the sites constraints.

In our proposed model, it is assumed that if the same query is issued at multiple sites, it will be treated as a different query and could have different frequency values. However, to accomplish optimal dynamic re-allocation, we utilize the query update frequency values of all distributed fragments to make use of them every time a re-allocation process is required. To produce the initial phase of fragment distribution (that might contain fragments duplication), the proposed technique will assume that fragments have already been distributed across network sites based on a given retrieval matrix (*RM*) and query frequency matrix *(QF)* described in section III.B.2. The proposed approach assumes that, retrieval requests will be handled with no communication cost by allocating each fragment to the requesting site. For example, fragment *F3* in *RM* (table 3), could be requested by queries *Q3* and *Q4* only. And since queries *Q3* and *Q4* are issued at sites *S1* and *S3*, then fragment *F3* will be allocated to both sites.

A new technique to re-allocate the resulting fragments from the initial phase in a non-redundant fashion is presented in this work. This re-allocation technique is performed using update matrix only along with its queries frequency, because the update request usually incurs more cost than retrieval requests. Therefore, tables 4 and 5, which are *UM* and *QF* matrices will be used as input to construct a new matrix called Fragment Update Frequency Matrix, *FUFM* (table 10) as base for re-allocation. The *FUFM* matrix can be defined as the values of query updates at a particular site for the affected fragments.

The FUFM matrix will be used to produce the Cumulative Update Frequency Table, *CUFT* (table 11). And multiplying *CUFT* with *DM*, we obtain the fragment pay matrix (*FUPM)* presented in table 12. Finally, using *FUPM* matrix, the site with maximal update cost value for particular fragment will be chosen as the candidate site to store that fragment. However, if there is any site constraint violations for the candidate site, then the fragment will be migrated to the site with the next highest update cost value. Moreover, if there is more than one site having the same update cost value for a certain fragment, a Fragments Priority *(FP)* procedure will be run at those sites to allocate that fragment to the site with the highest *FP* value and subsequently this action guarantees that the current fragment copy will be the only copy in the entire network. The phases of our proposed system are depicted in Figure 2.

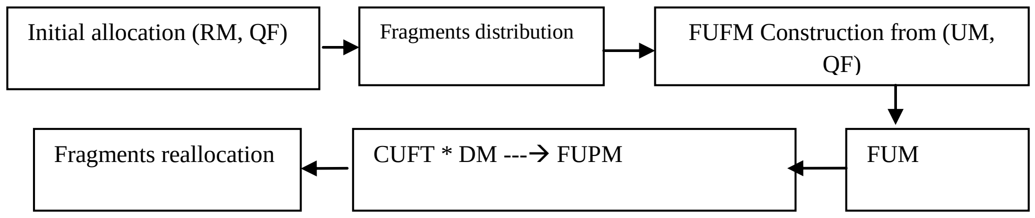

Figure 2. Reallocation Phases

3.2 Fragments Priority (FP)

The fragment priority (*FP*) procedure gives the number of fragment accesses, i.e. how many times the queries of every site (*QS*) reach the intended fragment. This is used if there are requests for the same fragment F_i from more than one site having the same update cost value for that fragment. Thus, to avoid fragment duplication over sites, *FP* will be calculated for each site ^{k}and will be used to make the decision on allocating fragment *Fi* to the site that have the hig$_{h=1}$hest *FP* value.

$$FP\ (S_j,F_i) = \sum (QF_{hi} * QS_{hi}) * DM_{j,h}\ , \quad 1 <= j <= m \qquad (1)$$

$$QS_{hi} = \sum_{i=1}^{n} RM_{hi} + \sum_{i=1}^{n} UM_{hi}, \qquad 1 <= h <= h \qquad (2)$$

Equation (1) is used to calculate *FP* to avoid the fragment duplication. However, equation (2) produces the cumulative of accesses made by queries.

3.3 Cost Functions

Cost is incurred by queries in the process of accessing and updating fragments at a particular site. As mentioned earlier, to perform the calculation of *FUPM*, we take *FUFM* matrix as input and use the following cost functions:

$$CUFT = \sum_{i=1}^{n} \sum_{i=1}^{m} (QF_i * UF(F_i)) \qquad (3)$$

$$DM_{ij} = Min\ (CC_{ij}), \qquad 1 <= i, j <= m \qquad (4)$$

$$FUPM = \sum_{i=1\ \ i<>i'}^{m} CUFT\ (F_i) * DM_{jj'}\ , 1 <= j, j' <= m \qquad (5)$$

$$\sum_{i=1}^{m} Max\ (FUPM_{ij}), \qquad 1 <= i <= n \qquad (6)$$

Equation (3) is used to produce *CUFT* table, Equation (4) applies the "*minimum"* algorithm on communication cost matrix, Equation (5) is used to build *FUPM* and finally Equation (6) represents the re-allocation decision. In general, the redistribution process could be considered as optimal when reaching an allocation scheme that results in a minimal total update costs for queries. Since this problem is practically incomputable, we decided here to use a heuristic method instead.

4. The Proposed Algorithm

```
ALGORITHM: Re-Allocation
Input:
For each relation {1,… , K}
{QN = { Q1……, Qn} // a set of queries
F = { F1……., Fn } // DDBS fragments
S = { S1…….., Sm} // a set of network sites
RM matrix, UM matrix, QF matrix and communication cost matrix
Output: fragment reallocation scheme

// Initial Allocation Phase

Begin
  Start initial allocation using RM and QF matrices
  Build Distance Cost matrix (DM) = min (CCM)
Calculate_DM(CCM);
Calculate_CUFT (QFM, FUFM);
Calculate_FUPM (DM, CUTF);
```

```
End
// Step 1
// Distance matrix formation
Min(CCM);{
// Output is the distance matrix and the shortest path values
For k=1 to m do
 For i=1  to m do
  For j=1  to m do
    $DM_{ij}$ = minimum ($cc_{ij}$,$cc_{ik}$ + $cc_{kj}$)
    keep (SPA[i],SPA[j], $value_{ij}$);
  End for j
 End for i
End for k

  // Step 2
  // The sum of retrieval and update cost calculation
 Calculate_CUFT (QFM,FUFM);
// Output is CUFT table;
Begin
For j=1 to m     // sites
For h=1 to k     // site queries
 For i=1 to n    // fragments

    $CUTF_{jhi} = QF_{jh} * UM_{hi}$

  End i;
 End h;
End j;

End ;

// Step 3
// Computation of the pay of update costs for sites to access fragments
Calculate_FUPM(CCM, SRUM);
output is FUPM;
Begin
For j=1 to m //sites
For h=1 to m //sites
For i=1 to n // fragment

$FUPM_{ji}= CUFT_{ji} * DM_{j,h}$

End j;
End h;
End i;

End;
// Re- Allocation Phase

Start the post-allocation or re-allocation phase using (FUPM) matrix
// Step 4

// Allocation of the  fragment $F_i$ to the site incurring the highest update cost
Begin
For j=1 to m // sites
  For i=1 to n // fragments

    Allocate ($F_i$ to $S_j$) = Max ($FUPM_{ji}$)

If ($S_j$.constraints-violated= true)
     allocate Fi to the $S_{j+1}$ has the second highest update cost value
end if

If $S_j.F_i$ (FUPM_value) = $S_{j+!}.F_i$ (FUPM_value)
  FP ($S_j$,$S_{j+1}$,$F_i$,FUPM_value);
End if

  End for i
End for cni

End;

// Step 5
   // If more than one site require the same fragment then use fragment priority procedure
Allocate-FP($S_j$,$S_{j+1}$,$F_i$,FUPM_value);
//Output is an allocation of the intended fragment to its new site

Begin
// construct the access matrix
For j=1 to m do// sites
 For h=1 to k do            // queries
  For i=1 to n do           // attributes

   $ACC_{ji} =(RM_{i,h,j}+UM_{i,h,j}) * QF_{ji})$

End for
End for
End for

// To calculate fragment priority (FP) and to accurately perform the re-allocation phase,
   choose value (v) such that Payv($F_i$) = $Max_s^m$ and  find the  site with maximum update cost
   for fragment ($S_j$).  Allocate $F_i$ to $S_j$ (v)
   // allocate the fragment i to the  corresponding site $S_j$ (v)
  If many sites require one fragment $F_i$
          Run FP procedure at all sites that require $F_i$
          Allocate fragment to $s_j$ that has maximum FP
ENDFOR;
//

For j=1 to m do
 For i=1 to n do
  For h=1 to m do

   $FP(F_i, S_j)= Max (ACC_{hji} * DM_{jh} )$,                //given that j' <> j

  End for j
 End for i
End for h

End;
```

5. Experimental Results

5.1 Initial Allocation

To test our technique, we implemented it on an existing *DDBS*. Given the requirements described in section II.B and based on *RM* and *QF* matrices, we obtained the initial allocation table (*table 9*).

Table 9. Initial Allocation

S/F	F_1	F_2	F_3	F_4	F_5	F_6
S_1	0	1	1	1	1	1
S_2	1	1	0	1	1	1
S_3	1	1	1	1	1	0
S_4	1	1	0	1	0	1

$$\text{Where} \begin{cases} 1, & \text{means } F_i \text{ is assigned to Site } S_j \\ 0, \text{otherwise} \end{cases}$$

The basic problem at this stage is the fragments duplication (e.g. *F1* appears in three sites). However, this problem will be resolved in post allocation (re-allocation phase).

5.2 Re-Allocation Process

As mentioned earlier in section II.C.1, the *FUFM* (*table 10*) is constructed using *UM* and *QF* matrices.

Table 10. FUFM Matrix

S	Q	QF	F1	F2	F3	F4	F5	F6
S1	Q3	2	3	0	0	2	0	1
	Q5	1	0	0	2	0	1	0
S2	Q1	2	0	0	2	0	1	0
	Q2	3	0	1	1	0	0	2
	Q5	1	0	0	2	0	1	0
S3	Q1	3	0	0	2	0	1	0
	Q3	1	3	0	0	2	0	1
	Q4	4	0	1	0	0	1	1
S4	Q1	1	0	0	2	0	1	0
	Q5	2	0	0	2	0	1	0

Using *FUFM* matrix, we obtain *CUFT* table which is produced by multiplying every *QF* value to its corresponding *UM* value for every query at each site, then groping the results based on sites, as presented in table (11).

Table 11. CUFT Matrix

S/F	F1	F2	F3	F4	F5	F6
S1	6	0	2	4	1	2
S2	0	3	9	0	3	6
S3	3	4	6	2	7	5
S4	0	0	6	0	3	0

Then, to produce *FUPM*, the *DM* matrix is multiplied by *CUFT* matrix as shown in *table 12*.

Table 12. FUPM

S/F	F1	F2	F3	F4	F5	F6
S1	27	51	153	18	105	75
S2	72	56	118	48	115	80
S3	54	42	210	36	84	102
S4	87	56	120	58	98	97

Finally, based on *FUPM* table, the fragment re-allocation process will be performed. Every fragment will be re-allocated to the site that incurs the highest cost value for updating it. From Table 12, it can be noticed that sites *S2* and S4 are having the same update cost value for *F2*. But fragment *F2* is allocated to site *S2* because the fragment priority (*FP*) of site *S2* is higher than that of site *S4*. Table 13 gives the final re-allocation results.

Table 13. Fragment Allocation

F	F1	F2	F3	F4	F5	F6
S	S4	S2	S3	S4	S2	S3

6. Performance Evaluation

Whenever an attribute A_i is allocated to the site S_k having the highest update cost value for it, the communication cost will be dramatically minimized.

$$\text{Maximum } (S_j)\ [\sum_j \sum_k (QF_{jk} * UM_{kl})]$$

i.e. allocate attribute A_i to site S_j in which the number of query update references to A_i is the highest.

Proof of this theory is as follows; Suppose that the communication cost is given as summation of all update accesses performed by site queries to attributes such that:

$$Comm\_Cost = QF_{11} * UM_{11} + QF_{21} * UM_{12} + \ldots\ldots. + QF_{jk} * UM_{kl}$$

$$= \sum_{i}^{n} \sum_{k}^{q} \sum_{j}^{m} (QF_{jk} * UM_{kl})$$

Provided that the targeted allocation site is S_h and h $\in[1\ldots.m]$ given that h < > j, so :

$$Comm\_Cost = \sum_{i}^{n} \sum_{k}^{q} [\sum_{j}^{m} QF_{jk} * UM_{kl}]$$

$$= \sum_{i}^{n} \sum_{k}^{q} [\sum_{j}^{m} QF_{jk} * UM_{kl}]$$

$$= \sum_{j}^{n} \sum_{k}^{q} [\sum_{i}^{m} QF_{jk} * UM_{kl} - QF_{hk} * UM_{kl}]$$

$$= \sum_{i}^{n} \sum_{k}^{q} \sum_{j}^{m} QF_{jk} * UM_{kl} - \sum \sum QF_{hk} * UM_{kl}$$

And assuming that $\sum_{j}^{n} \sum_{k}^{q} \sum_{i}^{m} QF_{jk} * UM_{kl}$ is a constant, then maximizing $\sum_{i}^{n} \sum_{k}^{q} QF_{hk} * UM_{kl}$ will minimize

the communication cost. However, if site S_h and S_j are the same where $h,j \in [1\ldots..m]$ then, the re-allocation will

be based on selecting max $\sum_{i}^{n} \sum_{k}^{q} QF_{hk} * UM_{kl}$

As mentioned earlier, the two ways to solve the allocation problem are through probabilities and heuristic methods. Our proposed approach exploits the concepts of both and thus attempts to enhance the *DDBS* performance in a dynamic environment such that the decision of fragments re-allocation will be taken based on the changes of query information to reflect the dynamic nature of the *DDBS* and minimize the communication cost and query response time. It introduces a simplified model for fragments re-allocation process. Moreover, minimizing the communication cost matrix to get distance cost matrix (*DM*) forms a key factor in the *DDBS* performance enhancement. However, in our model the re-allocation process is based on the update cost only because the update operations incur more cost than retrieval, especially when the fragments replicated across several sites at the initial allocation.

Thus, our proposed model concentrates on finding an optimal fragments re-allocation method to improve *DDBS* performance by minimizing the communication cost and the query response time. Eventually, providing a better data availability by maintaining one copy of a fragment over *DDBS* at post allocation phase.

Advantages and drawbacks of our model are summarized in the following points;

1. The re-allocation process is a dynamic method based on extracted information (retrieval and update frequency values) from existing *DDBS*. So, any changes in the site queries and their frequencies will have an effect on the re-allocation process.

2. It guarantees that no fragments duplication (duplication only allowed at the initial allocation phase) at the post allocation phase, i.e. whenever there is more than one site having the same update cost value, in this case the fragment priority procedure will be used to specify the optimal site for allocation. At post allocation phase only the primary key duplication across sites is allowed.
3. There is less computation complexity. For instance, if k is the number of queries, m is the number of sites and n is the number of fragments the complexity of re-allocating fragments will be O $(n + m^2 * n)$.

The only drawback for this method is that when the queries information continuously changes in faster way or when the number of sites and fragments largely increase, the method will be more complicated.

Table 14 shows the fragments allocations to sites at the initial allocation phase according to our example discussed in section (V.A). It gives the fragments distribution across sites for existing *DDBS* before applying our method, there are several queries requesting attributes to produce 19 allocations. In contrast *table 15* shows the re-allocation process after using our approach at post allocation which gives only 6 allocations after applying our method will have a great impact on the *DDBS* performance improvement which is clearly shown in the differences between the initial and final distribution for fragments (*table 16*).

Table 14. Initial Allocation

S/F	F_1	F_2	F_3	F_4	F_5	F_6
S_1	0	1	1	1	1	1
S_2	1	1	0	1	1	1
S_3	1	1	1	1	1	0
S_4	1	1	0	1	0	1

Table 15. Fragment Allocation

F	F1	F2	F3	F4	F5	F6
S	S4	S2	S3	S4	S2	S3

Table 16 and *figure 3* show the number of fragments distributed to sites at the initial and post allocation phases.

Table 16. Method Performance Evaluation

Site number	Fragments number (initial)	Initial distribution	Fragments number (post)	Post distribution	Optimization
S_1	5	19	-	6	-
S_2	5		2		60%
S_3	5		2		60%
S_4	4		2		50%

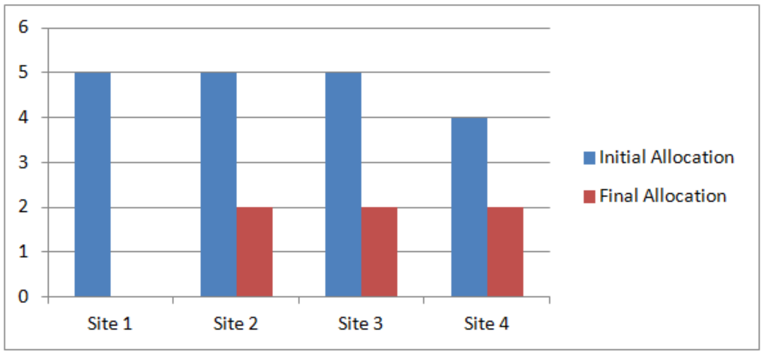

Figure 3. Allocation Process

7. Conclusions

This work presents a dynamic heuristic re-allocation model to find an optimal solution for fragments re-allocation in *DDBS* environment. The idea behind this approach is to re-allocate data fragments to their optimal location based on the maximum cost of updating a fragment at the intended site for allocation. This "maximum update cost" technique provides an optimal solution and takes into consideration the site constraints and network topology.

The proposed model improves the overall *DDBS* performance and results in significant reduction in the frequency of fragment migrations from one site to the other and consequently reducing the data transmission overhead. If there is more than one site having the same update cost value for a certain fragment, then that fragment will be allocated to the optimal site according to our proposed fragment priority (*FP*) procedure.

This work guarantees the avoidance of fragment duplication via the usage of the *FP* procedure which results in low computational complexity. The beauty of this approach relies in its simplicity and efficiency.

Acknowledgment

The author would like to thank and appreciate the help received from the Research Centre at the College of Computer and Information Sciences in King Saud University for providing the necessary facilities to accomplish this work.

References

[1] Abdalla H, "An Efficient Approach for Data Placement in Distributed Systems", Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering, **(2011)**.
[2] Abdalla H, "Using a Greedy-Based Approach for Solving Data Allocation Problem in a Distributed Environment", to appear in the Proceedings of the 2008 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'08), **(2008)**.
[3] Brunstrom A, Leutenegger ST and Simha R, "Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database With Changing Workloads", in Proceedings of the 1995 International Conference on Information and Knowledge Management, Baltimore,MD, USA, **(1995)**, pp. 395-402.
[4] Apers P, "Data Allocation in Distributed Databases", ACM Trans. Database Systems, vol. 13, no. 3, **(1988)** September, pp. 263-304.
[5] Abiteboul S, Bonifati A, Cobena G, Manolescu I and Milo T, "Dynamic XML Documents with Distribution and Replication", Proc. 2003 ACM SIGMOD Int'l Conf. Management of Data, **(2003)**, pp. 527-538.
[6] Wilson B and Navathe SB, "An Analytical Framework for the Redesign of Distributed Databases",, in Proceedings of the 6th Advanced Database Symposium, Tokyo, Japan, **(1986)**, pp. 77-83.
[7] Chiu G and Raghavendra C, "A Model for Optimal Database Allocation in Distributed Computing Systems", Proc. IEEE INFOCOM 1990, vol. 3, **(1990)** June, pp. 827-833.

[8] Chu WW, “Optimal File Allocation in Multiple Computer Systems”, IEEE Transaction on Computers, vol. C-18, no. 10, **(1969)**.
[9] Cormen TH, Leisorson CE, Rivest RL and Stein C, “Introduction to Algorithms 2nd Edition”, McGraw Hill **(2001)**.
[10] Lin WJ and Veeravalli B, “A Dynamic Object Allocation and Replication Algorithm for Distributed System with Centralized Control”, International Journal of Computer and Application, vol. 28, no. 1, **(2006)**, pp. 26-34.
[11] Karimi Adl R and Rouhani Rankoohi SMT, “A new ant colony optimization based algorithm for data allocation problem in distributed databases”, Knowl Inf Syst 2009, vol. 20, **(2009)** January 23, pp. 349–373.
[12] Menon S, “Allocating Fragments in Distributed Databases”, IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 7, **(2005)** July.
[13] Hababeh IO, Ramachandran M and Bowring N, “A high-performance computing method for data allocation in distributed database systems”, Springer J Supercomput, vol. 39, **(2007)**, pp. 3-18.
[14] Singh A and Kahlon KS, "Non-replicated Dynamic Data Allocation in Distributed Database Systems", IJCSNS International Journal of Computer Science and Network Security, vol. 9, no. 9, **(2009)** September, pp. 176-180.
[15] Sleit A, AlMobaideen W, Al-Areqi S and Yahya A, "A Dynamic Object Fragmentation and Replication Algorithm In Distributed Database Systems", American Journal of Applied Sciences, vol. 4, no. 8, **(2007)**, pp. 613-618.
[16] John LS, “A Generic Algorithm for Fragment Allocation in Distributed Database System”, ACM **(1994)**.
[17] Tâmbulea L and Horvat M, "Dynamic Distribution Model in Distributed Database", Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844, Vol. III, **(2008)**, Suppl. issue: Proceedings of ICCCC 2008, pp. 512-515.
[18] Tamhankar A and Ram S, “Database Fragmentation and Allocation: An Integrated Methodology and Case Study,” IEEE Trans. Systems,Man and Cybernetics—Part A, vol. 28, no. 3, **(1998)** May.
[19] Ozsu MT and Valduriez P, “Principles of Distributed Database Systems”, 2nd ed.Prentice-Hall International Editions, **(1999)**.
[20] Toadere T, “Graphs. Theory, Algorithms and Applications”, Editura Albastra, Cluj-Napoca, ROMANIA, **(2002)**.
[21] Upadhyaya S and Lata S, "Task allocation in Distributed computing VS distributed database systems: A Comparative study", IJCSNS International Journal of Computer Science and Network Security, vol. 8, no. 3, **(2008)** March.
[22] Ulus T and Uysal M, “Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems”, Pakistan Journal of Information and Technology, vol. 2, no. 3, **(2003)**, pp. 231-239.
[23] Rivera-Vega PI, Varadarajan R and Navathe SB, “Scheduling Data Redistribution in Distributed Databases”, in IEEE Proceedings of the Sixth International Conference on Data Engineering, **(1990)**, pp. 166-173.
[24] Dowdy LW and Foster DV, “Comparative models of the file assignment problem,” ACM Computing Survey, vol. 14, no. 2, **(1982)**, pp. 287-313.
[25] Wolfson O, Jajodia S and Huang Y, “An Adaptive Data Replication Algorithm”, ACM Trans. Database Systems, vol. 22, no. 2, **(1997)**, pp. 255-314.
[26] Wolfson O and Jajodia S, "An Algorithm for Dynamic Data Distribution", Proceedings of the 2nd Workshop on the Management of Replicated Data (WMRD-II), Monterey, CA, **(1992)** November.

Authors

Hassan Ismail Abdalla Holds a PhD in computer Science from UK, an Oracle Certified Professional (OCP). Currently working as an assistant professor of Computer Sciences at King Suad University. Worked as an assistant professor of Computer Sciences at Prince Sultan University and as an Oracle Database Administrator (DBA) for Hackney College and Anglo Continental Consultants Ltd. in London.