Journal of Universal Computer Science, vol. 20, no. 10 (2014), 1469-1487
submitted: 30/10/13, accepted: 20/6/14, appeared: 1/10/14 © J.UCS

A Novel Vertical Fragmentation, Replication and Allocation Model in DDBSs

Hassan I. Abdalla
(King Saud University, Riyadh, Saudi Arabia
haabdalla@ksu.edu.sa)

Ali A. Amer
(King Saud University, Riyadh, Saudi Arabia
aliaaa2004@yahoo.com)

Hassan Mathkour
(King Saud University, Riyadh, Saudi Arabia
Mathkour@ksu.edu.sa)

Abstract: Modern database systems are commonly distributed, and data is kept at isolated locations (sites). The various sites are connected through communications links, which may be of low speed resulting in bottlenecks for data transfer between sites. Data replication is considered as one of the effective methods in dealing with such situations to achieve improved performance in distributed database systems (DDBSs). In this work, authors explore a new model for improving performance in distributed database environment by using a vertical fragmentation method along with a novel replication and allocation techniques. The solution procedure consists of a new vertical fragmentation model to fragment a relation and two phases of allocation of fragments to nodes. The paper discusses the tradeoffs between the different scenarios for finding an optimal way of deciding on attribute allocation to sites by evaluating performance based on the collected requirements. This model will significantly reduce communication cost and query response time in DDBSs.

Keywords: Distributed DBMS, Vertical Fragmentation, allocation, replication, frequency-matrix, heuristics, clustering
Categories: H.2, H.2.8, H.3

1 Introduction

Two efficient ways by which the performance of distributed applications could be enhanced are sites grouping and fragment allocation. Sites grouping is the process of placing different sites accessing the same data in one group. Fragment allocation, on other hand is a technique to distribute the fragmented attributes over sites to minimize the data transfer cost and the number of message exchanges during query processing.

DDBSs design proposed in [Ezeife and Barkerm 98] aims to develop applications performance by minimizing the amount of irrelevant data accessed by applications and by minimizing the amount of data transferred in processing queries [Karlapalem et al., 92]. In [Abuelyaman, 08], an attribute usage matrix (AUM) and Bond energy algorithm were used to produce vertical fragments. In [Navathe et al., 89], the work

presented in [Iacob, 10] was extended by developing an algorithm using a graphical technique. A partition evaluator was presented in [Chakravarthy et al., 94] to measure the goodness of a Vertical Fragmentation (*VF*). A mixed fragmentation technique is proposed in [Hababeh et al., 03] and an attribute usage frequency matrix (AUFM) is used with a cost model for VF in [Mei et al., 03]. In [Surmsuk and Thanawastien 07], a Stat Part static algorithm for VF is proposed. [Lee et al., 00] Presented a heuristic method for specifying file and workload allocation simultaneously on a LAN to minimize the response time for processing transactions. In [Bellatreche and Karlapalem, 98], the combined methods and class allocation problem was formulated and a model to calculate the total data transfer cost incurred was developed.

Several allocation aspects in different contexts were considered by many authors. For instance [Daudpota, 98] combined security considerations into the fragment allocation process; while [Wujuan and Veeravalli , 03] proposed an algorithm for allocation and replication that adapts to the changing patterns of online requests; Grouping partitioning into an automatic design framework was considered in [Agrawal et al., 14], and [Chin, 02] suggested incremental allocation and reallocation based on changes in the workload. In [Ma et al., 06], data allocation algorithms were presented to achieve the minimum overall communication cost. However, mathematical modeling technique and data allocation, and a genetic algorithm were developed in [Whitten et al., 97] to allocate operations to nodes.

However, based on the rule of thumb that says if fragment/attribute is queried more frequently than it is modified, then replication is advisable. In our technique duplication is made if there is more than one cluster having the same update cost value for the updated attribute. In addition, we can still access one of the copies even if some of the copies have failed. Actually, this benefit comes with the additional cost of keeping all copies identical. This cost, which could potentially be high, consists of total storage cost and communication cost [Rahimi and Haug, 10].

This paper presents a novel method for a synchronized vertical fragmentation and a two-phases of attribute allocation in a distributed environment. The proposed algorithm presents a new approach for clustering sites into clusters to which attributes would be allocated (and replicated when needed) at the first allocation phase according to the update cost values for clusters. The second phase of allocation process will be performed based on attribute priority (*AP)* value that will be computed for each site in the cluster at which the given attribute already has been allocated. The site with the highest *AP* value will be the candidate site to store the attribute. Thus, the proposed model is going to minimize the communication cost by distributing the attributes over different clusters. It will also increase the a availability and integrity of data by allocating multiple copies of the same attribute (except primary key attribute) over clusters. Moreover, it reduces the query response time between sites within the same cluster.

Our proposed model will perform the fragmentation and allocation process on the fly following these steps:

1. Identifying the relation/fragment to be fragmented/sub fragmented.
2. Clustering network sites.

3. Constructing the Retrieval and Update Frequency Matrix (*RUFM*) using Attribute Retrieval Matrix (*ARM)*, Attribute Update Matrix (*AUM)* and Query Frequency Matrix (*QFM).*
4. For clusters, constructing the Sum of Retrieval and Update Frequency Matrix *(SRUM)* based on *RUFM.*
5. Calculate Pay of Retrieval and Update Frequency Matrix (*PRUM*), based on *SRUM*, and Communication Cost Matrix (CCM) of clusters.
6. Identifying the following allocation phases:
    a. Allocation phase-1: using PRUM, choose cluster C_k that has the maximum allocation value to be the candidate cluster to store attribute A_i. The replication decision for A_i over clusters is determined.in this phase.

    b. Allocation phase-2: For all sites within cluster C_k, allocate attribute A_i to site S_j that has the highest Attribute Priority (*AP*) value for A_i, while the site constraints are maintained.

2 The Proposed Vertical Fragmentation Model

2.1 Motivation Example

Figure 1, shows an example consisting of three scenarios for finding an optimal allocation for attribute (A_i). Attribute A_i has retrieval and update frequency values (*RF* and *UF*) obtained from queries executed across different sites.

In the first case (*Figure 1-A*), we have one site, two queries (Q_1 and *Q2)* with their frequencies (*QF1, QF2)* and one or more attributes. These queries originating at site S_1 performs retrieval/update operations on fragments/attributes placed at the same site. Thus, the retrieval/update operation will be done without incurring communication cost. In other words, the best location for attribute A_i will be the original owning site (S_1). While in the second case (*Figure 1-B*), we have two sits (S_1 *and* S_2) and two queries with their frequencies over the two sites. Query Q_1 originating at site S_1 with frequency QF_1 accessing attribute A_i *(retrieval or update)* and Q_2 originating at site S_2 with frequency QF_2 accessing A_i *(retrieval or update)*. For each query, transmission cost will be calculated as following to select the best location for attribute A_i.

The $TC_1 = QF_1 * C_{12} * ((ARM\ (Q_1,A_i) + AUM(Q_1,A_i))$ *and*

$TC_2 = QF_2 * C_{21} * ((ARM\ (Q_2,A_i) + AUM(Q_2,A_i))$.

If $TC_1 > TC_2$, then the best location for attribute A_i going to be S_1, otherwise will be site S_2.

In the third case (*Figure 1-C*), a little bit more complex case of two clusters (C_1 and *C)* are presented. Cluster C_1 consists of two sites, site S_1 issues query Q_1 with frequency (QF_1) and site S_2 issues query Q_2 with frequency QF_2. And cluster C_2 consists of site S_3 issuing two queries Q_2 and Q_3 with their frequency $(QF_2)$and (QF_3) *respectively*. In this case, a more complex computation is needed to find the best

allocation of attributes across multiple sites distributed over clusters with regard to the communication cost between clusters and sites. This computation problem will be considered in our proposed method as will be explained next.

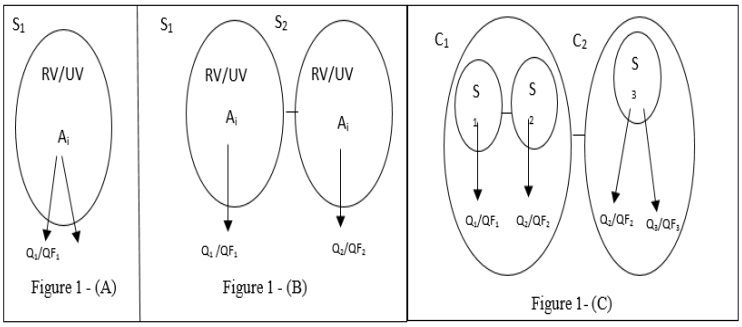

Figure 1(A, B, C): Different Clustering Scenarios to allocate attribute Ai.

2.2 The Proposed Model

The proposed model will be performed based on attributes $A[A_1,....,A_n]$ of relation R. Every attribute A_i has a retrieval frequency value (RF) and an update frequency value (UF). Since all queries accessing attributes are not expected to be for retrieval and update purpose. Therefore, the query purpose has to be clearly distinguished either $RF(A)$ or $UF(A)$. Both $RF(A)$ and $UF(A)$ are assumed to be obtained by queries accessing fragments at each site.

Actually, the RF and UF values assumed to be achieved by the executed queries in $DDBS$ that account for more than 75% of the processing in $DDBS$. Each query Q_h can be issued from any network site S_j with a certain frequency (QF_{hj}). The execution frequencies of k queries at m sites can be represented by $m \times k$ matrix (QF_{km}). The DDBSs network consists many sites $S(S_1, S_2, \ldots\ldots\ldots,S_m)$, where each site has capacity $C(C_1, C_2, \ldots\ldots\ldots,C_m)$, lower attribute limit (LAL) and upper attribute limit (UAL). LAL and UAL represents the minimum and maximum allowed attributes at each site respectively (see table 1).

However, to enhance performance, the resulted fragments/attributes need to be allocated dynamically across different sites. Based on the collected requirements which is represented by the *CMS, ARM, AUM and QFM (tables 4,6,7 and 8),* the vertical fragmentation will be performed. The *ARM, AUM and QFM* used to construct the Retrieval and Update Frequency Matrix *(RUFM), as* shown in *table 11.*

S	C	LAL	UAL
1	100 KB	1	3
2	70 KB	1	2
3	90 KB	1	4
4	120 KB	1	4
5	150 KB	1	3
6	100 KB	1	2

Table 1: Sites Constraints

RUFM contains clusters queries as rows and relation attributes as columns, *RUFM* in its turn is used to form the Sum of Retrieval and Update Frequency Matrix *(SRUM*) for clusters *(as shown in table 12)*. Finally, *SRUM* is multiplied by the clusters communication cost matrix (*table 10)* to produce Pay of Retrieval and Update Matrix *PRUM,(as shown in table13)*. The attributes allocation in the first phase is performed based on the *PRUM* table where each *PRUM* element gives the total cost for each cluster C_{cni} to access attribute A_i. The second phase of attribute allocation is done based on a technique called Attribute Priority *(AP)*. *AP* is a mathematical technique that is computed using Attributes Access Matrix ($ACC_{h,i,jIi}$), table 2, and sites communication matrix (*CMS), table4* . Every element of the $ACC_{h,i,jIi}$ matrix represents the number of times query *q* at site *j* accesses attribute A_i, *(*retrieval or update access*)* allocated at site *h* in cluster C_{cni} (table 2).

	A1	A2	A3	A4	A5	A6	A7	A8
S1	10	14	17	9	28	10	9	17
S2	2	10	16	10	7	9	6	6
S3	9	3	3	12	9	9	18	0
S4	30	8	6	22	18	24	25	9
S5	17	15	19	19	20	26	14	8
S6	28	23	20	14	44	12	13	37

Table 2: ACC matrix

3 The Proposed Allocation Model

Given that there is a set of *n* attributes $A = \{A_1, A_2, ..., A_n\}$exploited by a set of *k* queries $Q = \{Q_1, Q_2, ..., Q_k\}$ on a set of *m* sites $S=\{S_1, S_2, ..., S_m\}$ which are grouped into *h* clusters $C=\{C_1, C_2, ..., C_m\}$in a fully connected network, our allocation model concentrates on finding the optimal distribution of *A* on *S*. The allocation problem

may be expressed mathematically by a function from the set of fragments to the set of clusters $\pi: A \rightarrow C$ An optimal allocation is the one which minimizes the objective function. Therefore, The allocation phases must also take into account the imposed constraints such as site capacity.

3.1 Sites Clustering

The site clustering is a fast way to decide on fragment/attribute allocation. Based on the threshold value (threshold is a communication cost unit), the clustering will be performed using Site Communication Cost Matrix *CMS* (table 3) to reduce storage overheads.

In this paper, we adopt clustering algorithm based on communication cost to determine on sites assignment to clusters according to our proposed model. If the communication cost between two or more sites is less than, or equal to the threshold value (*thv*), then they will be grouped together in one cluster, otherwise they won't (table 9 shows the site grouping to form clusters). Moreover, communication cost matrix between clusters is computed (table 10). Adopting this procedure will further minimize the communication cost between clusters and sites. Threshold value is determined by the network system as used in [CERI et al., 84].

Sites	S1	S2	S3	S4	S5	S6
S1	0	10	8	2	4	6
S2	10	0	7	3	5	4
S3	8	7	0	3	2	5
S4	2	3	3	0	11	5
S5	4	5	2	11	0	5
S6	6	4	5	5	5	0

Table 3: Communication Cost Matrix

3.2 Allocation Phases

Phase1: Using *SRUM* matrix and *CCM* matrix *(table 10)*, the pay of retrieval and update matrix (*PRUM*) will be formed. And using *PRUM table*, the cluster with the maximum value, *Max (PRUM(i))* for attribute A_i will be selected to be the candidate cluster to store the attribute, given that there is no site constraints violation. However, based on the rule of thumb, if fragment/attribute is queried more frequently than being modified, then replication is advisable (in this work, we adopted the attribute replication scenario). But, this replication could come with additional processing and communication cost which could be potentially high.

Phase2: the second phase will be more straight-forward and will use the formal description of the allocation problem. As mentioned earlier. The *DDBSs* network consists of many sites $S(S_1, S_2, \ldots\ldots\ldots, S_m)$, where each site has capacity $C(C_1, C_2, \ldots\ldots\ldots\ldots, C_m)$ and attribute limits (*LAL* and *UAL)*. Each attribute is required by at least one of the cluster sites.

However, this allocation phase is performed using the attribute priority (*AP)* technique which is constructed in two steps. **The first step,** is building the Access Cost matrix (*ACC*) by multiplying *ARUM* matrix by *QF* matrix (*ARUM = ARM + AUM*)). Every $ACC_{h,i,j,cni}$ represents the cost of site S_h queries accessing attribute A_i allocated at site S_j in cluster C_{cni}. **The second step**, is multiplying *CMS* matrix by *ACC* matrix. The attribute priority will be calculated to find the optimal place for attributes across sites within the original owning clusters given that the sites constraints are preserved. Any site S_j having the highest access cost of the attribute A_i *(AP)*, will be a prime candidate to store attribute A_i.

The proposed model definitions, notations and cost functions will be presented in table 4 as follows:

n	number of attributes
m	number of sites
k	number of queries
CN	number of clusters
i	attribute index, I = {1,..., n}
j	site index, j = {1,..., m}
h	query index, h = {1,..., k}
cni	cluster index cni ={1,…,cn}
C_i	storage capacity at site i
FL_i	fragment limit for site i
RF_{ij}	achieved retrieval frequency for A_i by S_j
UF_{ij}	achieved update frequency for A_i by S_j
LAL	lowest limit of attributes allowed at each site
UAL	upper limit of attributes allowed at each site
V	attribute size
QFij	= 1, if attribute i is required by a query at site j and 0, otherwise
ACC_{ij}	access frequency of the i^{th} query at site j
CCM_{ij}	the communication cost unit between cluster C_i and cluster C_j
CMS_{ij}	the communication cost unit between site S_i and site S_j
CRF	the total of retrieval frequency for all sites queries within the owning cluster
CUF	the total of update frequency for all sites queries within the owning cluster
U_{ij}	=1 if A_i required for update by cluster C_j, and 0, otherwise
R_{ij}	=1 if A_i required for retrieval by cluster C_j, and 0, otherwise

Table 4: Used Notations

3.3 The Definitions Cost Functions

The proposed model is targeting to minimize the total cost function (*TC*) which is the clusters communication cost consists of the sum of retrieval and update costs.

Minimize (TC) = $RC + UC$ (1)

Retrieval Cost (RC) function:

$$RC = \sum_{i=}^{m} (1 - AR_{ij}) * CRF_{ij} * R_{ij,} \qquad 1 <= i <= n \qquad (2)$$

Update Cost (UC) function:

$$UC = \sum_{i=}^{m} (1-RA_{ij}) * CUF_{ij} * U_{ij} * CCM_{ij,} \qquad 1 <= i <= n \qquad (3)$$

Cost is incurred when retrieving and updating attributes by site queries. To calculate attribute allocation for a certain relation, we take the *RUFM* of the relation and the following cost functions as input:

$$SRUM_{cni,i} = \sum_{cni}^{cn} \sum_{j}^{m,} \sum_{j}^{n} ((RF_{h,i,j,cni} * QF_{i,j,cni}) + (UF_{h,i,,,cni} * QF_{i,j,cni})) \qquad (4)$$

$$PRUM_{cni,i} = \sum_{i}^{m} \sum_{i\mathbf{m}}^{n} (SRUMcn_{i,i} * CC_{ji}) \qquad (5)$$

$$AV_i = Max(\sum_{i}^{n} PRUM_{cni,i}) \qquad (6)$$

$$ARUM_{ij} = \sum_{i}^{n} ARM_{ij} + \sum_{i}^{n} AUM_{ij} \qquad 1 <= h <= h \qquad (7)$$

$$ACC_{h,i,jcni} = \sum_{i}^{n} QF_{i,j} * ARUM_{ij} \qquad 1 <= h <= h \qquad (8)$$

$$AP(S_j, A_i) = Max(\sum_{h}^{k} (ACC_{h,i,jcni} * DMSij),\ 1 <= j <= m,\ 1 <= i <= n \qquad (9)$$

Equation (4) is used to build the *SRUM* matrix, every element of this matrix represents the sum of all queries cost for specific cluster to access attribute A_i allocated at site S_j. Equation (5) is used to calculate *PRUM* matrix based on the *SRUM* and *CCM matrices*. Every $PRUM_{i,j}$ represents the total cost incurred by all sites queries of a specific cluster (it is different from *attributes' original cluster*) to access to attribute A_i allocated at particular S_j *at the original owning cluster*. Equation (6) gives (AV_i) value that represents the actual cost to access attribute A_i. It will be used in the first allocation phase to select the prime candidate cluster to store attribute A_i. Equation (7) is used to compute Attribute Retrieval and Update Matrix (*ARUM*) that will be used to construct Attribute Access Matrix (*ACC*). Equation (8) is used to

calculate the access matrix (*ACC*) based on *ARUM* and *QF* matrices, and finally equation (9) represents the mathematic expression of attribute priority which is constructed using equations 4 and 5 (as shown in table14) to perform the second allocation phase.

The following constraints have to be maintained throughout the implementation of allocation process.

$$\sum_{i}^{n} X_{ij}=1 \ , \ \forall j=1,\ldots., m \qquad (10)$$

$$\sum_{i}^{n} V_i * Q_{ij} <= C_i \ \forall j=1,\ldots., m \qquad (11)$$

$$LA_i <= \sum_{i}^{n} X_{ij} <= UA_i \ \forall j=1,\ldots., m \qquad (12)$$

$$X_{ij} \ , \ Q_{ij} \ , \ L_j \in (0,1) \qquad (13)$$

Constraint (10) states that an attribute will be allocated to only one site within a cluster. Constraint (11) ensures that no site will receive more than its capacity. Constraint (12) guarantees that the number of assigned attributes will be between *LAL* and *UAL* and finally constraint (13) is the binary constraint on the decision variable.

4 Heuristics for Vertical Fragmentation

In this work, we propose a novel cost model that performs a relation fragmentation and allocation on the fly. This model towards to find the optimal distribution for attributes that minimizes the communication cost and response time given that it collects the attributes retrieval and update information at the site where the queries are executed.

Based on the obtained communication cost values between sites, the sites will be grouped into clusters and based on the gathered information about the retrieval and update query frequencies the allocation process is done in two phases. In the first phase the model evaluates and calculates the cost of fragmenting the intended relation/fragment and allocating the resulted attributes to the network clusters and the decision on attribute replication across cluster will be made at this stage. In the second phase, the allocation decision is made for attribute A_i that has been already allocated to the cluster C_k by selecting a site S_j that incurs the highest access cost for that allocated attribute A_i.

This model is considered as a heuristic model that determines the cluster with the highest update cost (and access cost for sites within a cluster) to find the optimal fragmentation and allocation scenarios given that the queries are already assigned to sites prior to the fragmentation phase. In other words, our proposed cost model finds the relationship between the total update costs of clusters (Pay of Access Cost) and the individual access costs of sites to find the optimal attribute allocation. The model starts from the following formula:

$$\text{Access_cost}(A_i) = \sum_{j}^{m} (\text{Access}(A_i), \quad \text{i.e.}$$

$$= \sum_{j}^{m} \sum_{h}^{k} \sum_{i}^{n} ((\sum \text{ARM}(A_i) * QF_{hj} + \sum_{i}^{n} \text{AUM}(A_i) * QF_{hi})) * CMS_{lj} \quad l <> j$$

$$= \sum_{j}^{m} (\sum_{h}^{k} \sum_{i}^{n} (\text{ARM}(A_i) + \text{AUM}(A_i)) * QF_{hj}) * CMS_{lj}$$

$$= \sum_{j}^{m} [(\sum_{h}^{k} \sum_{i}^{n} \text{ARUM}(A_i) * QFh_j) * CMS_{lj}]$$

$$= \sum_{cni}^{cn} (\sum_{j}^{m} \text{ACC-site}(A_i) * CMS_{lj})$$

$$= \sum_{cni}^{cn} \text{Total_ Access_ Cluster}(A_i)$$

$$= \text{TAC}(A_i) \qquad 1 <= cni <= cn$$

The above formulas give one and only one heuristic solution towards an optimal allocation of allocate attribute A_i to the network clusters/sites.

5 Results and Performance Evaluation

To test our model we have implemented it on the employee relation (table 5) of the DDBS. For simplicity we have considered six sites of the distributed system for allocation.

Name	Birth-date	Job-Id	Salary	Location	Dept-Id
Michael	4/2/1980	MGR	2000	S1	2
Shelley	4/2/1980	MAN	1200	S2	3
Nancy	12/1/1978	MAN	1750	S2	2
Den	22/9/1989	MAN	2100	S3	1
Jone	22/9/1977	MAN	1400	S1	3
Matthew	2/2/1982	MAN	1200	S2	4
Adam	4/2/1980	MGR	1500	S1	2
Kevin	12/12/1979	MAN	1200	S3	1
Zamel	22/9/1977	MAN	1500	S2	5

Table 5: Employee table

5.1 Requirements

The Attribute Update Matrix (AUM), Attribute Retrieval Matrix (ARM), and Query Frequency Matrix (QFM) for all sites queries (extracted by the DDBS designers) are shown in tables 6, 7 and 8.

	A1	A2	A3	A4	A5	A6	A7	A8
Q1	2	1	1	0	3	1	0	2
Q2	0	0	1	3	3	1	2	1
Q3	2	0	1	0	1	2	3	0
Q4	3	2	0	0	4	0	0	5
Q5	0	4	2	2	0	0	2	0
Q6	1	1	0	3	0	3	1	0

Table 6: ARM

	A1	A2	A3	A4	A5	A6	A7	A8
Q1	0	2	0	1	5	1	2	0
Q2	0	3	5	0	0	2	0	1
Q3	1	1	0	4	2	1	3	0
Q4	3	0	3	1	2	0	1	4
Q5	2	0	2	2	1	3	0	2
Q6	4	0	0	0	1	2	1	0

Table 7: UAM

Site/Query	Q1	Q2	Q3	Q4	Q5	Q6
S1	2	2	0	1	0	0
S2	0	2	0	0	1	0
S3	0	0	3	0	0	0
S4	0	0	3	1	0	3
S5	1	3	0	0	0	3
S6	3	0	0	3	2	0

Table 8: QFM

5.2 Clustering Sites

Having the threshold value (*thv* = *6*), and using the above communication cost matrix (*CMS*) presented in table 3, the network sites will be grouped (according to the clustering algorithm) as displayed in table 9.

Cluster / Site	S 1	S 2	S 3	S 4	S 5	S 6
C1	1	1	1	0	0	0
C2	0	0	0	1	1	0
C3	0	0	0	0	0	1

Table 9: Sites Grouping (Clusters)

The communication cost between clusters (*CCM*) is presented in table 10.

Cluster / Site	C1	C2	C3
C1	6	3.8	5
C2	3.8	5.5	5
C3	5	5	4

Table 10: Communication Cost Between Clusters

5.3 Constructing Tables

The *AUM, ARM and QFM* matrices will be used as input to produce the retrieval and update frequency matrix (*RUFM*) shown in table 11.

	S	Q	QF	A1	A2	A3	A4	A5	A6	A7	A8
C1	S1	Q1	2	0	2	0	1	5	1	2	0
				2	1	1	0	3	1	0	2
		Q2	2	0	3	5	0	0	2	0	1
				0	0	1	3	3	1	2	1
		Q4	1	3	0	3	1	2	0	1	4
				3	2	0	0	4	0	0	5
	S2	Q2	2	0	3	5	0	0	2	0	1
				0	0	1	3	3	1	2	1
		Q5	1	2	0	2	2	1	3	0	2
				0	4	2	2	0	0	2	0
	S3	Q3	3	1	1	0	4	2	1	3	0
				2	0	1	0	1	2	3	0
C2	S4	Q3	3	1	1	0	4	2	1	3	0
				2	0	1	0	1	2	3	0
		Q4	1	3	0	3	1	2	0	1	4
				3	2	0	0	4	0	0	5
		Q6	3	**4**	**0**	**0**	**0**	**1**	**2**	**1**	**0**
				1	1	0	3	0	3	1	0
	S5	Q1	1	0	2	0	1	5	1	2	0
				2	1	1	0	3	1	0	2
		Q2	3	0	3	5	0	0	2	0	1
				0	0	1	3	3	1	2	1
		Q6	3	**4**	**0**	**0**	**0**	**1**	**2**	**1**	**0**
				1	1	0	3	0	3	1	0
C3	S6	Q1	3	0	2	0	1	5	1	2	0
				2	1	1	0	3	1	0	2
		Q4	3	3	0	3	1	2	0	1	4
				3	2	0	0	4	0	0	5
		Q5	2	2	0	2	2	1	3	0	2
				0	4	2	2	0	0	2	0

Table 11: Retrieval and Update Frequency Matrix (RUFM)

5.3.1 First Phase Allocation

Based on the *RUFM* matrix, the summation of all queries cost (update & retrieval) for each cluster to access every attribute A_i is calculated (*SRUM*), as shown in table 12.

Cluster #	A1	A2	A3	A4	A5	A6	A7	A8
C1	21	41	47	31	46	28	35	23
C2	47	23	25	40	38	47	39	17
C3	28	23	20	14	44	12	13	37

Table 12: Sum of Quey Cost Matrix (SRUM)

Multiplying *SRUM* matrix with clusters communication cost matrix (CCM), the pay of attributes update across clusters will be produced as shown in table 13.

Cluster #	A1	A2	A3	A4	A5	A6	A7	A8
C1	444.6	448.4	477	408	640.4	406.6	423.2	387.6
C2	478.3	397.3	416.1	407.8	603.8	424.9	412.5	365.9
C3	452	412	440	411	596	423	422	348

Table 13: PRUM

Table 14 shows the final allocation of attributes to clusters in phase 1.

Attribute	A1	A2	A3	A4	A5	A6	A7	A8
Cluster	2	1	1	3	1	2	1	1

Table 14: Phase-1 final allocation

5.3.2 The Second Allocation Phase

The second allocation phase is performed using Attribute Priority (*AP*) technique as shown in table 15.

	A1	A2	A3	A4	A5	A6	A7	A8
S1	388	338	392	400	522	386	388	332
S2	450	352	384	391	673	413	413	385
S3	358	351	404	316	587	327	282	406
S4	380	347	400	363	544	420	309	325
S5	538	315	320	422	583	427	442	382
S6	348	254	306	359	431	391	363	211

Table 15: Attributes Priority (AP) Matrix

As mentioned earlier, *AP* technique matrix is constructed by multiplying *ACC (table2)* matrix with *CMS matrix (table 4)*. Then any site S_j with highest *AP* for attribute A_i, will be the prime candidate to store attribute A_i.

Based on *AP* matrix, the final allocation phase will be produced. For example, according to the *AP* matrix, attribute A_7 should be allocated to site S_2. But site S_2 has an attribute limit constraint violation which is supposed to be in the *range* between 1 and 2 inclusive (i.e. 1 < = S_2.attribute limit <= 2). Thus, instead of assigning attribute A_7 to site S_2 in the same cluster we select the next site that has the least number of attributes to be the new candidate to store attribute A_7 (table 16).

Site	S1	S2	S3	S4	S5	S6
Attribute	A7	A2, A5, A7	A3, A8	A1	A6	A4

Table 16: phase2 - Final Allocation

5.4 Discussion and performance evaluation

Our proposed model enhances the DDBS performance in a dynamic environment where the decision on vertical fragmentation and allocation is taken based on the changes of query information to minimize the communication cost and query response time. It introduces a simplified cost model for vertical fragmentation and allocation.

Actually, clustering of sites enhances the DDBS performance by minimizing the communication cost, eliminating unnecessary redundant attribute allocation across clusters, reducing the query response time between sites within the same cluster and finally by providing high data availability by maintaining multiple copies of the same attribute in different clusters when needed. As explained earlier, the first allocation phase is based on the update costs because the update operations incur more cost than retrieval operation, especially when attributes are replicated across clusters.

Theory: Communication cost will be minimized when attribute A_i is allocated to the cluster with the high update cost.

The proof of this theory is as follows: Suppose that the communication cost is given as the summation of all update accesses performed by clusters to attributes such that:

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

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

And suppose that the targeted allocation cluster is C_h and $h \in [1 \ldots m]$ given that $h <> j$, so :

$$\text{Comm. Cost} = \sum_{i}^{n} \sum_{k}^{q} (\sum_{i}^{m} QF_{jk} * UM_{kl})$$

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

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

Provided that ∑ ∑ ∑ QFjk * UMkl is a constant, then maximizing ∑ ∑ QFhk * UMkl will minimize the communication cost. However, if h equal j and consequently the cluster Ch is the same as cluster Cj given that h &, j ∈ [1…..m] the re-allocation will be based on selecting max(∑ ∑ QFhk * UMkl).

In general, the dynamic re-allocation scheme performs better when information change is less frequent and vice versa. Briefly, we can summarize the merits in the following;

(1) It is a novel approach that presents a simple cost model to make a dynamic vertical fragmentation and allocation on the fly.
(2) The allocation process is performed in two phases, *phase1* minimizes the communication cost between clusters and *phase 2* reduces the query response time between sites within the same cluster.
(3) It allows for attributes replication across clusters when necessary, i.e. when more than one cluster have the same update cost, in this case replication is allowed. Of course, primary replication is an exception. However, attribute replication is not allowed between sites within the same cluster.
(4) Since our method is dynamic and based on the extracted information (retrieval and update frequency info) from the existing DDBS, any changes in sites queries and their frequencies will affect the re-allocation process when repeated.

The drawback of this method relies in the computation complexity and allocation process time such that when there are a large number of sites and fragments as mentioned in merit (5), the allocation process will be affected. However, the complexity of our approach is still low compared to the affinities-based methods presented in some previous algorithms.

Table 17 and table 18, show an initial random allocation of the attributes to sites and clusters respectively for an existing DDBS before applying our model, the initial query requirements for attributes generates 21 allocations. Table 19 shows the re-allocation process after adopting our approach which produces only 8 allocations which is considered as a great improvement in performance compared to the initial allocation.

S/F	A 1	A 2	A 3	A 4	A 5	A 6	A 7	A 8
S_1	1	0	0	1	1	0	0	0
S_2	1	1	0	0	0	1	1	0
S_3	0	1	1	1	0	0	1	0
S_4	1	1	0	0	1	1	0	0
S_5	1	0	1	0	1	0	0	0
S_6	0	0	1	1	0	0	0	1

Table 17: initial allocation to sites

C/F	A1	A2	A3	A4	A5	A6	A7	A8
C_1	1	1	1	1	1	1	1	0
C_2	1	1	1	0	1	0	0	0
C_3	0	0	1	1	0	0	1	1

Table 18: initial allocation to clusters

Cluster number	Site number	Attributes number (Before)	Initial distribution	Attributes number (After)	Final distribution	Optimization
C1	S_1	3	11	-	5	60%
	S_2	4		3		
	S_3	4		2		
C2	S_4	4	7	1	2	72%
	S_5	3		1		
C3	S_6	3	3	1	1	66%

Table 19: Method Performance Evaluation

Table 20 and figure 2 show the attributes allocation before and after using our method

Cluster #	Initial Allocation	Final Allocation
Cluster 1	11	5
Cluster 1	7	2
Cluster 2	3	1

Table 20: attributes allocation

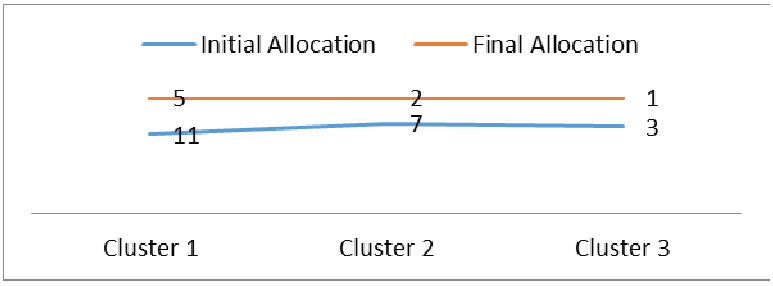

Figure 2: Attributes Allocation

Table 21 and figure 3 show the performance optimization for the DDBS before and after applying our algorithm.

Cluster #	Optimization before	Optimization after
Cluster 1	40%	60%
Cluster 1	28%	72%
Cluster 2	34%	66%

Table 21: performance optimization

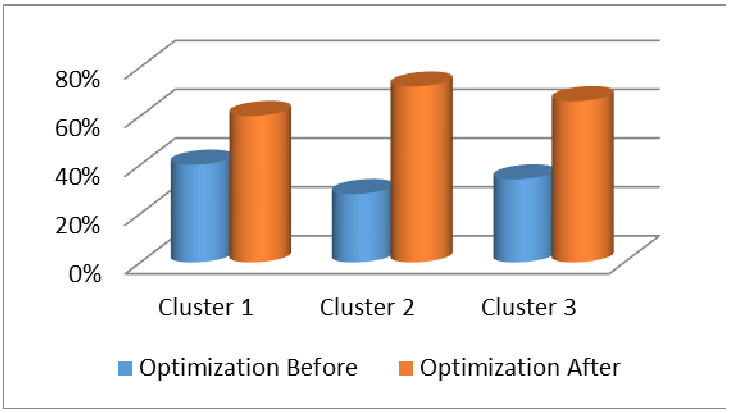

Figure 3: Performance Improvement

6 Conclusion

This paper addressed performance enhancement problem in DDBSs by adopting a vertical fragmentation and allocation model using a technique that efficiently divides DDBS relations into fragments to find an optimal data allocation method. In this work, a site clustering technique that further minimizes communication cost has been used with a two phase allocation approach. The first phase forms the clusters and allocates fragments to each cluster satisfying the selection criteria. In addition, the replication decision is taken in this phase. In the second phase, a technique called attribute priority (AP) is used, it gives the total cost for (each site within the cluster) accessing an attribute in another cluster. The proposed model has taken into consideration the tradeoffs between different scenarios for finding an optimal way of deciding on attribute allocation to sites.

Acknowledgement

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

References

[Abuelyaman, 08] Abuelyaman, E. S., "An optimized scheme for vertical partitioning of a distributed database," Int. Journal of Computer Science & Network Security, Vol. 8, (2008).

[Agrawal et al., 14] Agrawal, S., Narasayya, V., & Yang, B., "Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design,"Proc. 2004 ACM SIGMOD Int'l Conf. Management of Data,'2004', 359-370.

[Amir and Abdalla, 12] Amir, A., Abdalla, H.: An Integrated Design Scheme for Performance Optimization in Distributed Environments", International Conference on Education & E-Learning Innovations ICEELI' 2012, 1-3, Sousse, Tunisia, July 2012.

[Bellatreche and Karlapalem, 98] Bellatreche, L., Karlapalem, K., & Li, Q. (1998),"Complex Methods and Class Allocation in Distributed OODBs", Proceedings of the 5th International Conference on Object Oriented Information Systems. Paris, Sept,(1998), 239 – 256.

[Chakravarthy et al., 94] Chakravarthy, S., Muthuraj, J., Varadarajan, R., & Navathe, S. B., "An objective function for vertically partitioning relations in distributed databases and its analysis," Distributed and Parallel Databases, Springer, Vol. 2, No. 2,(1994), 183–207.

[CERI et al., 84] Ceri, S., Negri, M., and Pelagatti,G.: Distributed Database Principles and System. McGraw-Hill, New York, (1984).

[Chin, 02] Chin, A. G, "Incremental Data Allocation and Reallocation in Distributed Database Systems," Data Warehousing and Web Eng.,S. Becker, ed., chapter 7, IRM Press, 137-160, 02.

[Daudpota, 98] Daudpota, N. H., "Five steps to construct a model of data allocation for distributed database systems", Journal of Intelligent Information Systems: Integrating Artificial Intelligence and DatabaseTechnologies, vol.11, no.2, (1998), 153-68.

[Ezeife and Barkerm 98] Ezeife, C. I., & Barker, K., "Distributed Object Based Design: Vertical Fragmentation of Classes".International Journal of Distributed and Parallel Databases, Vol. 6, No. 4, Kluwer Academic Publishers,(1998), 327-360.

[Hababeh et al., 03] Hababeh, I. O., Bowring, N. I. C. H. O. L. A. S., & Ramachandran, M. U. T. H. U., " AN INTEGRATED STRATEGY FOR DATA FRAGMENTATION AND ALLOCATION IN A DISTRIBUTED DATABASE DESIGN", ICITNS 2003 International Conference on Information Technology and Natural Sciences, (2003).

[Iacob, 10] Iacob, N, "DATA REPLICATION IN DISTRIBUTED ENVIRONMENTS", Annals of the „Constantin Brâncuşi" University of Târgu Jiu, Economy Series, Issue 4,(2010).

[Karlapalem et al., 92] Karlapalem, K., Navathe, S. B., & Morsi, M. M., Morsi "Issues in Distribution Design of Object Oriented Databases". Distribut ed Object Management. Morgan Kaufmann Publishers, (1992).

[Lee et al., 00]Lee, H., Park, Y. K., Jang, G., & Huh, S. Y., "Designing a distributed database on a local area network: Amethodology and decision support system", Information and Software Technology. (2000), 171-184.

[Ma et al., 06] Ma, H., Schewe, K. D., & Kirchberg, M., "A heuristic approach to vertical fragmentation incorporating query information," in Proc. 7th International Baltic Conference on Databases and Information Systems (DB&IS), (2006), 69–76.

[Mei et al., 03] Mei, A., Mancini, L. V., & Jajodia, S., "Secure Dynamic Fragment and Replica Allocation in Large-Scale Distributed File Systems," IEEE Trans. Parallel and Distributed Systems, vol. 14, no. 9, (2003) 885-896.

[Navathe et al., 89] Navathe, Shamkant B.,and Mingyoung Ra ,"Vertical partitioning for database design: A graphical Algorithm," ACM SIGMOD Record, Vol. 14, No. 4, 440-450, 89.

[Rahimi and Haug, 10] Rahimi, S. K., & Haug, F. S., DISTRIBUTED DATABASE MANAGEMENT SYSTEMS. A JOHN WILEY & SONS, INC., PUBLICATION, IEEE Computer Society, (2010).

[Surmsuk and Thanawastien 07] Surmsuk, P., & Thanawastien, S., "The integrated strategic information system planning Methodology," IEEE Computer Society Press, (2007), 467-475,

[Wujuan and Veeravalli, 03] Wujuan, L., & Veeravalli, B., "An Adaptive Object Allocation and Replication Algorithm in Distributed Databases," Proc. 23rd Int'l Conf. Distributed Computing Systems Workshops (ICDCSW '03),(2003),132-137.

[Whitten et al., 97] Whitten, J. L., Barlow, V. M., & Bentley, L., "Systems Analysis and Design Methods", McGraw-Hill Professional, (1997).