I J C T A, 9(20), 2016, pp. 473-482

An Extended Efficient Approach to Dynamic Fragment Allocation in Distributed Database Systems

Raju Kumar* and Neena Gupta**

ABSTRACT

The distributed database better suits the real world organizations having different branches or sites at different locations. The performance of the distributed database is heavily dependent on the allocation of fragments at different sites. The main motive is to place the data fragments at different sites in such a way, so that total data transfer cost can be minimized while executing a set of queries. Moreover allocating data fragment dynamically is better than static fragment allocation. This paper proposed an extended approach to redundant and non-redundant dynamic fragment allocation in distributed database system. It additionally explains how to tackle the dynamic data fragment transfer, when more than one site simultaneously qualify for fragment allocation in redundant and non-redundant distributed database system. The comparison with other dynamic data fragment allocation approaches proved that the proposed extended approach is more capable and improves the overall system performance.

Keywords: Distributed database system,Redundant and non-redundant,Dynamic data allocation,Simultaneous fragment access.

1. INTRODUCTION

Distributed databases are in great demand among the organizations spread over multiple locations. Distributed database is defined by [1] as a single logical database that is partioned and distributed over different sites of a computer network. Each site of the network is independent to perform local applications. Each site must participate in the execution of at least one global application, which needs data accessing at several sites using a communication subsystem. The reliability and performance of the distributed database system is highly depend on fragment allocation [2]. For better performance fragments need to be allocated closer to the site where they are most frequently used.

Distributed databases can be of redundant or non-redundant nature. Therefore according to [1] and [3] fragment allocation can be done in redundant and non-redundant manner. In redundant approach same fragment may be allocated on several sites, whereas in non-redundant approach each fragment is allocated on single site. Several dynamic data allocation approaches already developed to improve the performance of distributed databases. In this paper an extended efficient approach of dynamic fragment allocation algorithm for redundant and non-redundant distributed database systems have been introduced which is an extension of the work carried out by [4]. The extended approach additionally explains how to deal the dynamic data transfer, when more than one site simultaneously qualify for data allocation in redundant and non-redundant distributed database system. The comparison with other dynamic data allocation approaches justify that the proposed extended approach is more capable and improves the system performance efficiently.

* Department of Computer Science, Gurukul Kangri Vishwavidyalaya, Haridwar, India, *Email: rajuk12@gmail.com*

** Department of Computer Science, Kanya Gurukul Mahavidyalaya, Dehradun (Second Campus of Gurukul Kangri Vishwavidyalaya, Haridwar), India, *Email: neena71@hotmail.com*

The rest of the paper is organized as follows: In section II the overview of the related work done so far is described. In section III the proposed extended algorithm for redundant and non-redundant fragment allocation is described. Section IV demonstrates algorithm working with relevant data. In section V the comparison of proposed extended algorithm with algorithms proposed by [4], [5], [6] and [7] is performed**.** Finally the contribution of the study is concluded in section VI.

2. RELATED WORK

Allocation of data or fragment is one of the key design issues of distributed database. Over the past few years several works have been published on the issue of dynamic fragment allocation to the nodes in distributed database systems.A threshold algorithm for non-redundant distributed databases is introduced by [8]. In this algorithm, the fragments are continuously reallocated according to the changing data access patterns. Thereafter an algorithm called Threshold and Time Constraint Algorithm (TTCA) was proposed by [9], which reallocates non-replicated data with respect to the changing data access patterns with time constraint in distributed database systems. This algorithm suffers from scaling problem. Thereafter [10] introduced Extended Threshold Algorithm (ETA) which not only eliminated the scaling problem of TTCA and also reduced space requirements. A new dynamic fragment allocation algorithm based on access threshold, time constraints of database accesses and volume of data transmission was proposed by [5]. Further [4] extended the work done by [5] and introduced distance parameter besides already existing parameters such as access threshold, volume of data transmission & time constraints of database accesses and proposed an extended algorithm for non-redundant allocation which efficiently handle the situation where multiple sites qualify for fragment relocation.

The redundant dynamic fragment allocation algorithm proposed by[7] considered multiple issues including lazy replication strategy, fragment's correlation, and non-uniform distances between network sites. Experimental results reflect that solution provided by this algorithm is efficient to the fragment allocation problem. Furthermore[6] and [11] proposed a dynamic fragment re-allocation model to find an optimum data re-allocation solution in distributed database systems. In this model fragment re-allocation is performed across sites based on update and communication cost values for each fragment individually. A systematic survey of total of 31 research papers for dynamic data allocation in redundant and non-redundant manner is presented by [12]. In this paper, an extended dynamic fragment allocation algorithm is proposed, which is an extension of the work carried out by [4] and [5]. The proposed extended fragment allocation approach provides solution for redundant and non-redundant fragment allocation and also improves the overall system performance.

3. PROPOSED EXTENDED DYNAMIC FRAGMENT ALLOCATION ALGORITHM

The Threshold Time Volume and Distance Constraints Algorithm (TTVDCA) introduced by [4] allocates fragments on the basis of parameters such as access threshold, volume of data transmission, and time constraints of database accesses. It additionally includes distance parameter to handle the situation where multiple sites qualify for fragment reallocation. This algorithm is limited to non-redundant allocation of fragments. The new proposed extended efficient algorithm is the extension of the work carried out by [4], [5] and provides solution for redundant and non-redundant allocation of fragments and called it TTVDCA-RNR. The proposed TTVDCA-RNR algorithm has two phases- preparation phase and action phase. The notations used in this paper are defined in table 1.

3.1. Preparation Phase

In distributed database system suppose there are Y sites and X data fragments. In this phase, each site has at least one fragment allocated to it and possesses corresponding row of Site Distance Matrix containing

Table 1
Algorithm Notations

Notation	*Meaning*
X	Number of data fragments in distributed database system
Y	Number of sites in distributed database system
F_i	The i^{th} data fragment
S_j	The j^{th} site
α	Access threshold for fragment relocation
β	Time constraint for fragment relocation
A_p^q	Access log record for p^{th} access at site q
n_i^j	Total number of accesses from site S_j to the fragment F_i within time interval β up to current access time t
V_i^j	Volume of data transmitted between fragment F_i and site S_j within time interval β up to current access time t
D_i^j	Distance between site S_i and site S_j

information about distance of that site with respect to other sites. The fragment allocation matrix is stored at each site to prepare its access plan. The information about two parameters- Access Threshold for fragment reallocation (α) and Time constraint for fragment reallocation (β) are also stored at each site.

Step1: Initially allocate all the fragments of distributed database over different sites using any static allocation method in redundant / non-redundant manner.

Step2: Assign constant value for Access threshold (α) and Time constraint (β) at each site.

Step3: Store a row of site distance matrix at each site – showing corresponding site distance from all other sites.

Step4: Store Access_Log table at each site. This table has following schema:

Access_Log (AFID, ASID, ADateTime, DataVol)

Where AFID means ID of the fragment which is accessed, ASID means ID of the site which accesses the fragment, ADateTime means date and time of fragment access, and DataVol means volume of data transmitted to and from the accessed fragment. Each site stores an access log record for each access to the fragments allocated to that site. Each Access_Log record is denoted by $\mathbf{A_p^q}$, which means p^{th} access at site S_q, where p=1,2,3,.....□ ∞ and q = 1,2,3,....Y.

Step5: Access Access_Log table daily at a particular time t (say 24 hours i.e. 00:00:00 - hh:mm:ss) and delete all records older than time constraint β up to current access time t at each site.

Suppose time interval β up to current access time t is 7 days. This may be implemented by executing a query using SQL at each site like "Delete from Access_Log where DATEDIFF (SYSDATE, ADateTime) > 7".

Two assumptions are made. Firstly there exists some distributed concurrency control mechanism that preserves ACID properties of the transactions; and ensures that for each write operation on a fragment stored at a particular site, changes made to the fragment should be reflected to all sites wherever copy of that fragment is stored. Secondly all the sites of distributed database system are fully connected.

3.2. Action Phase

This phase has a set of activities. These activities are performed every time after each access to the fragment. Let at time t, an access is made for fragment F_i allocated at site S_q from site S_j, where i= 1,2,3,...X, j=1,2,3,.....Y, q= 1,2,3,...,Y, and q = j or q ≠ j. The local agent at site S_q performs the following

steps 1 to 8for each access to a fragment allocated at that site by some query or application invoked from the same or different site:

Step1: For each access to fragment F_i stored at site S_q, write a log record A_p^q in Access_Log table at site S_q.

Step2: If the ID of the accessing site in the log record A_p^q is the same as the ID of site S_q, that means local access is made ($S_q = S_j$), then do nothing.

Step3: If the ID of the accessing site in the log record A_p^q is different than ID of site S_q, that means remote access is made ($S_q \neq S_j$), then go to the next step.

Step4: Calculate the total number of accesses made to the fragment F_i from each accessing remote site(s).

Suppose n_i^sdenotes the total number of accesses made to the fragment F_i allocated at site S_q by each site s, where s = 1,2,3,...,Y. If ($n_i^s <= \alpha$) then do nothing, otherwise go to the next step.

Step5: Calculate the average volume of data transmitted between fragment F_i and all sites (including site S_q where fragment F_i is allocated) from where accesses to the fragment F_i are made ($V_i^s t$) and the average volume of data transmitted between fragment F_i and each remote site(s) S_j from where accesses to the fragment F_i are made ($V_i^j t$) where s = 1,2,3,....,Y and $S_j \neq S_q$, then go to next step.

The average volume of transmitted data can be calculated using equation – (1). Consider $A_p^q V_i^s$ denotes the volume of data transmitted between the fragment F_i allocated at site S_q and the site s in the access_log A_p^q, where s = 1,2,3,...,Y. Furthermore let $V_i^s t$ denotes the average volume of data transmitted between the fragment F_i allocated at site S_q and the all accessing site s, then:

$$V_i^s t = (\sum A_p^q V_i^s) / \sum n_i^s \quad (1)$$

Step6: If each accessing remote site(s) S_j does not qualify the condition ($V_i^j t > V_i^s t$), then do nothing, otherwise go to next step.

Step7: If there is only one remote accessing site S_j qualify the condition ($V_i^j t > V_i^s t$), then the reallocate fragment F_i to site S_j and removed it from the current site S_q and update the fragment allocation matrix at each site accordingly, otherwise go to next step.

Step8: If more than one remote accessing sites S_j simultaneously qualify the condition ($V_i^j t > V_i^s t$), then find out the distance between the site S_q where fragment F_i is allocated and the sites which qualified the condition. Select the site which is at maximum distance from the site S_q and reallocate the fragment F_i to that site. Remove the fragment F_i from site S_q and update the fragment allocation matrix at each site accordingly.

Delay in operation is reduced by reallocating fragment F_i to site which is at maximum distance from current site S_q which results overall faster access.The main functionalities of the proposed algorithm can be implemented using the following procedures, written in English like language.

```
/* Deleting data beyond time constraint (β) */

Create or replace procedure Proc_AccessLog (vTimeCons in Number)

Begin

    Delete from Access_Log where DATEDIFF( SYSDATE, ADateTime) >vTimeCons;

End;
```

This procedure can be executed once at a particular time on daily basis at each site.

/* *Checking threshold constraint* (α) *and data volume* */

```
Create or replace procedure Proc_AlphaDataVol (vAlpha in Number)
    vCnt    Number(6);
    vAvgVolOne      Number(25);
    vAvgVolAll      Number(25);
Begin
    Select Count(*)+1 into vCnt
    From Access_Log
    Where ASID = Sj and AFID= Fi ;

If vCnt > vAlpha then
    Select Avg(DataVol) into vAvgVolOne
    From Access_Log
    Where ASID = Sj and AFID= Fi ;
Select Avg(DataVol) into vAvgVolAll
    From Access_Log
    Where AFID= Fi ;
If vAvgVolOne > vAvgVolAll then
        Create table Fi at site Sj
        As select * from Fi at site Sq ;
        Drop table Fi at site Sq ;
        Update Fragment_Allocation_Matrix at each site
        Set Fi = 1 where site = Sj ;
    End if ;
End if ;
End;
```

/* *Checking Simultaneous Multiple Site Qualify* */

If multiple site qualify for fragment reallocation simultaneously, i.e.

If ($vCntS_j$> vAlpha) and ($vCntS_k$> vAlpha) and ($vAvgVolOneS_j$> vAvgVolAll) and ($vAvgVolOneS_k$> vAvgVolAll) where j=1,2,3…..,Y and k =1,2,3,….,Y and j ‘“ k, then

```
    Create or replace procedure proc_SimulAccess ( )
        vMaxDistance Number(6):= 0;
        vDistance Number(6) ;
        vSite Number(6);
    Begin
```

```
        For c IN j.. k
        Loop
            Select S_c into vDistance at site S_q /*Checking site which is at max distance from site S_q*/
            From Site_Distance_Matrix ;
        if vMaxDistance < vDistance then
                vMaxDistance = vDistance ;
                vSite = S_c ;
        end if ;
    end loop ;
Create table F_i at site vSite
As select * from F_i at site S_q ;
Drop table F_i at site S_q ;
Update Fragment_Allocation_Matrix at each site
            Set F_i = 1 where site = vSite ;
End;
```

4. PROPOSED EXTENDED DYNAMIC FRAGMENT ALLOCATION ALGORITHM DEMONSTRATION WITH DATA

In order to explain how the new proposed extended fragment allocation algorithm can handle the redundant fragment reallocation in normal case as well as in special case when more than one site qualify for fragment reallocation; some hypothetical data are taken into consideration. Let there are four sites in fully connected distributed database system placed at some distance from one another as per following Site Distance Matrix:

Table 2
Site Distance Matrix

Site	S_1	S_2	S_3	S_4
S_1	0	300	500	650
S_2	300	0	700	400
S_3	500	700	0	600
S_4	650	400	600	0

Each site stores only its respective row of Site Distance Matrix. The distance shown is in km. Suppose there are total four fragments (F_1, F_2,...,F_4) of global relations in DDS. These fragments are allocated in replicated manner on 4 sites as per following scheme:

F_1: S_4 and S_1

F_2: S_3 and S_4

F_3: S_2 and S_3

F_4: S_1 and S_2

Table 3
Fragment Allocation Matrix

Site → Fragment ↓	S_1	S_2	S_3	S_4
F_1	1	0	0	1
F_2	0	0	1	1
F_3	0	1	1	0
F_4	1	1	0	0

$$Where \begin{cases} 1, fragment\ F_i is\ allocated\ to\ S_j \\ 0, otherwise \end{cases}$$

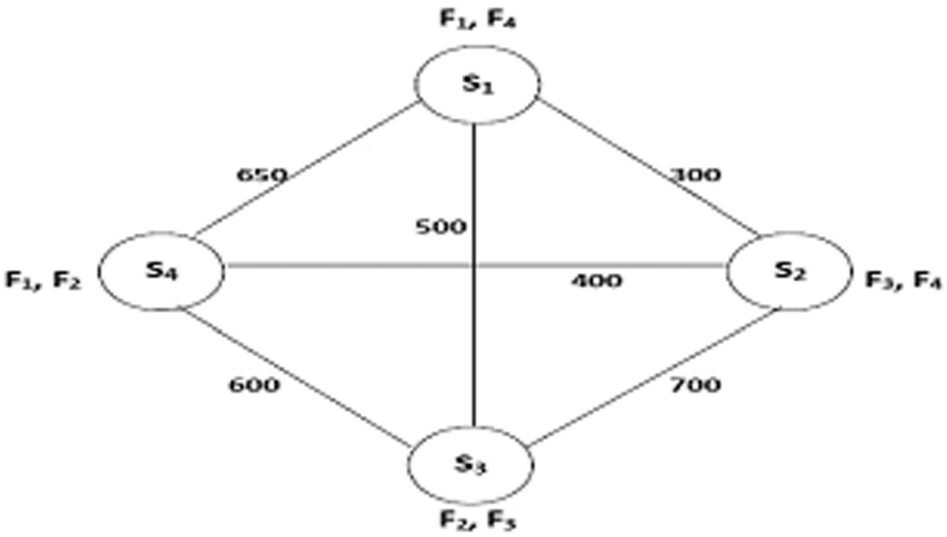

Figure 1: Distributed database sites with distance and allocated fragments

The entries of Access_Log table at site S_4 at a particular point of time is shown in table 4. The Access_Log at site S_4 shows that till 27-Mar-2015, total number of accesses made for fragment F_1 from sites S_2 and S_3 is 2. i.e. $n_1^2 = n_1^3 = \alpha = 2$. During the simultaneous access on 28-Mar-2015, total number of accesses made for fragment F_1 from sites S_2 and S_3 becomes 3, i.e. $n_1^2 = n_1^3 = 3 > \alpha$. The last two transactions are on hold because they indicate that there may be possibility of reallocation of fragment F_1.

Table 4
Access_Log at site S_4

AFID	ASID	ADateTime	DataVol
F_1	S_4	23-Mar-2015 11:15:19	150
F_1	S_2	23-Mar-2015 15:30:45	200
F_2	S_1	24-Mar-2015 16:58:56	180
F_1	S_4	25-Mar-2015 11:12:13	200
F_1	S_2	25-Mar-2015 12:54:55	274
F_1	S_3	26-Mar-2015 14:15:16	160
F_1	S_3	27-Mar-2015 16:17:18	300
F_1	S_2	28-Mar-2015 12:50:59	- - -
F_1	S_3	28-Mar-2015 12:50:59	- - -

Now, using Eq. (1) calculating following data transfer between different sites for fragment F_1:

- The average volume of data transferred between site S_4 and all other sites for fragment F_1
  = (150 + 200 + 200 + 274 + 160 + 300) bytes/6 = 1284 bytes/ 6 = 214 bytes
- The average volume of data transferred between site S_4 and site S_2 for fragment F_1
  = (200 + 274) bytes/2 = 474 bytes/2 = 237 bytes
- The average volume of data transferred between site S_4 and site S_3 for fragment F_1
  = (160 + 300) bytes/2 = 460 bytes/2 = 230 bytes

Since 237 bytes >214 bytes and 230 bytes > 214 bytes, and $n_1^2 = n_1^3 = 3 > \alpha$ so sites S_2 and S_3 both qualified for fragment F_1 to be reallocated. The site at greater distance from site S_4 will be selected for reallocation of the fragment F_1. Now using Site Distance Matrix fragment stored at site S_4 following information can be drawn:

Distance between site S_4 and S_2: D_4^2 =400 km and

Distance between site S_4 and S_3: D_4^3 =600 km

Therefore fragment F_1 is reallocated to site S_3 and removed from the site S_4 and Fragment Allocation Matrix is updated accordingly at each site. The site S_3 is at greater distance from current site S_4 where fragment F_1 is allocated, has to travel a lot as compare to site S_2, which results delay in operation. Reallocating fragment F_1 to site S_3 results overall faster access. Similarly dynamic fragment allocation for non-redundant distributed database system can also be performed.

5. COMPARISON

The proposed extended algorithm – TTVDCA-RNR for redundant and non-redundant dynamic fragment allocation is theoretically compared with other non-redundant algorithms proposed by [4] and [5] as well as with other redundant algorithms proposed by [6] and [7] on the basis of different properties as shown in table 5 and table 6:

Table 5
Comparison with non-redundant algorithms

Paper Reference → Property ↓	*Proposed Algorithm (TTVDCA-RNR)*	*[4]*	*[5]*
Redundant reallocation capability	Yes	No	No
Handling of multiple sites simultaneously qualified for reallocation	Yes	Yes	No
Additional query required to get target data in Access_Log table	No	Yes	Yes
Efficiency	Higher	Lower	Lower

The proposed algorithm TTVDCA-RNR has dual capability to reallocated fragment dynamically in redundant and non-redundant manner, while algorithms proposed by [4] and [5] have only capability of non-redundant reallocation. TTVDCA-RNR and algorithm proposed by [4] have capability of handling the case when multiple sites qualify for fragment reallocation while algorithm proposed by [5] does not have such capability. Target data is the data within Time Constraint (β) upto current access time. While algorithm TTVDCA-RNR does not require additional query to get target data, algorithms proposed by [4] and [5] require.Moreover algorithm TTVDCA-RNR is more efficient as compared to algorithms proposed by [4] and [5] because in TTVDCA-RNR data in Access_Log is always within Time Constraint (β) upto current access time, so no additional checking is required for data to be within Time Constraint (β) upto to current access time.

Table 6
Comparison with redundant algorithms

Paper Reference → Property ↓	*Proposed Algorithm (TTVDCA-RNR)*	*[6]*	*[7]*
Handling of multiple sites simultaneously qualified for reallocation	Yes	Yes	No
Fragment reallocation frequency flexibility	Yes	No	No
Prior information of query frequency requirement	No	Yes	No
Efficiency	Higher	Lower	Lower

For redundant fragment reallocation, the algorithm TTVDCA-RNR and algorithm proposed by [6] have capability of handling the case when multiple sites qualify for fragment reallocation while algorithm proposed by [7] does not have such capability. The algorithms proposed by [6]and [7] do not have Fragment reallocation frequency (FRF) flexibility. FRF flexibility means – the administrator of distributed database implementing the proposed algorithm has the flexibility to tighten or relax the condition for fragment reallocation. It is proportional to Time Constraint (β) / Access Threshold (α).

$$FRF\ \alpha\ (\beta / \alpha) \tag{2}$$

The proposed algorithm TTVDCA-RNR provide this facility by increasing/decreasing values of Access Threshold (α) and Time Constraint (β), to tune the algorithm to work best to fulfil the infrastructural, functional and business requirements of the organization. The algorithm proposed by [6] is working under the constraint of prior information of query frequency. Prior information of query frequency requirement means advance information about how many times a query will access a particular fragment. While the algorithm proposed by [7] and TTVDCA-RNR do not have such limitations. The efficiency of algorithm TTVDCA-RNR is better as compared to algorithms proposed by [6]and[7]; because in TTVDCA-RNR only two calculations are performed in general using Access_Log table;Firstly calculating total no. of accesses made by a site for particular fragment and secondly finding average volume of data transmitted between accessed fragment site and accessing sites. While algorithm proposed by [6] has to perform a lot of calculations – such as calculating cost of storing fragments, cost of querying from fragments, cost of updating fragments and cost of communication using several tables, which takes more time to response. Algorithm proposed by [7] also performs a lot of calculations – such as calculating fragment read cost, fragment update cost, fragment joining cost and fragment migration cost using several tables, which requires more time to response.

In view of the above comparisons it can easily be said that algorithm TTVDCA-RNR is better than all other algorithms proposed by [4], [5], [6] and [7] and improves the overall system performance.

6. CONCLUSION

The distributed database is the need of the current age. The performance of distributed database is highly depends on the efficient allocation of fragments to sites. In this paper, an extended efficient approach to redundant and non-redundant fragment allocation algorithm TTVDCA-RNR is proposed. Calculations performed on the basis of hypothetical data supports that the proposed extended algorithm TTVDCA-RNR is more capable; and the comparisons of proposed algorithm with other redundant and non-redundant fragment allocation algorithms shown that the current proposed algorithm is more efficient; and consequently improves the overall system performance. In future, experimental implementation of algorithm TTVDCA-RNR can be performed.

REFERENCES

[1] S. Ceri and G. Pelagatti,*Distributed Databases: Principles & Systems*, (Edition 2008), McGraw-Hill International, New Delhi, India, 2008.

[2] S. Agrawal, V. Narasayya and B. Yang, "Integrating vertical and horizontal partitioning into automated physical database design", *Proc. of ACM SIGMOD International Conf. Management of Data*, Paris, France, pp. 359-370, June 2004.

[3] M.T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, (3rd Edition), Springer Science+Business Media, LLC 2011, New York, USA, 2011.

[4] R. Kumarand N. Gupta, "An extended approach to non-replicated dynamic fragment allocation in distributed database systems", *IEEE Xplore*, ICICT-2014, 978-1-4799-2900-9/14/$31.00, pp. 861-865, 2014.

[5] N. Mukharjee, "Non-replicated dynamic fragment allocation in distributed database systems", *Springer-Verlag Berlin Heidelberg,* CCSIT, Part I, CCIS 131, Bangalore, pp. 560-569, January 2011.

[6] H.I. Abdalla, "A new data re-allocation model for distributed database systems", *International Journal of Database Theory and Application*, Vol. 5, No. 2, pp. 45-59, 2012.

[7] S. Kamali, P. Ghodsnia and K. Daudjee, "Dynamic data allocation with replication in distributed systems", *IEEE Explore,* 978-1-4673-0012-4/11/$26.00, 2011.

[8] T. Ulus and M. Uysal, "Heuristic approach to dynamic data allocation in distributed database systems", *Pakistan Journal of Information and Technology*, Vol. 2, No. 3, pp. 231-239, 2003.

[9] A. Singhand K.S. Kahlon, "Non-replicated dynamic data allocation in distributed database systems", *International Journal of Computer Science and Network Security*, Vol. 9, No. 9, pp. 176-180, 2009.

[10] R. Kumar and N. Gupta, "Non-redundant dynamic data allocation in distributed database systems", *International Journal of Computer Applications*, (Special Issue on ICNICT, No.6), pp. 06-10, 2012.

[11] A.A. Amer and H.I. Abdalla, "A heuristic approach to re-allocate data fragments in DDBSs", *Information Technology and e-Services (ICITeS) International Conference*, Sousse, Tunisia, pp. 01-06, March 2012.

[12] R. Kumar and N. Gupta, "Dynamic data allocation in distributed database systems: a systematic survey", *International Review on Computers and Softwares*, Vol. 8, No. 2, pp. 660-667, 2013.