An Interactive Data Assignment Algorithm for Non-Replicated DDBMSs

Hassan I. Abdalla[1], Abdelmonim M. Artoli[2]
[1]College of Technological Innovation, Zayed University, P.O. Box 144534, Abu Dhabi 11543, UAE
[2]Computer Science Department, College of Computer and Information Sciences, King Saud University
P.O.Box 51178, Riyadh 11543, Kingdom of Saudi Arabia
hassan.abdalla@zu.ac.ae; aartoli@ksa.edu.sa;

ABSTRACT

In this work, we propose an optimized data assignment algorithm for time dependent non-replicated distributed database systems based on a modified Dijkstra shortest path algorithm findings. Fragment assignment is determined from the access cost matrix and a threshold value obtained from the query matrix and site access record. Transmission, storage and computational overhead costs are computed and compared with existing algorithms. This model uses fragment preferred site technique to ensure non-replication at sites and to avoid constraints violation if fragment replication occurs. The algorithm is demonstrated through detailed example and proved robust in terms of transmission cost while it lags behind in terms of storage cost.

Keywords: Data Assignment, Real-time DDBS, Non-Replicated Databases, Interactivity.

1. INTRODUCTION

Real-time interactivity of fragment assignment in distributed databases is gaining more attention in recent years due to the vast expansion in the available state-of-the-art database technologies. Critical real time applications include, but are not limited to medical, military and business fields. With the increase in hardware capabilities both in memory access and CPU time, a need for robust algorithmic techniques is always there to make benefit of these developments. However, these algorithms did not consider real-time non-replicated environments. The main obstacle towards this goal remains finding an optimized method to calculate shortest path while re-allocating fragments to sites in real time.

The rest of this paper presents paper focus and rationales in section 2. The literature review is expressed in section 3. The methodology for interactive non replicated dynamic data assignment is given in section 4, the model demonstration is made in section 5. Finally, paper conclusions and future work are presented in section 6.

2. RESEARCH FOCUS AND RATIONALE

In this work, we propose an optimized data assignment algorithm for time dependent non-replicated distributed database systems based on a modified Dijkstra shortest path algorithm finding. Fragment assignment is determined from the access cost matrix and a threshold value obtained from the query matrix and site access record. Transmission, storage is seen reduced. This model uses fragment preferred site technique to ensure non-replication at sites and to avoid constraints violation if fragment replication occurs. The implemented algorithm proved robust in terms of computational overhead and transmission cost while it lags in terms of storage cost.

The proposed work will allow 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. However, attribute replication is not allowed between sites within the same cluster.

Since our method is dynamic and based on the extracted information from the existing DDBS, any changes in sites queries and their frequencies will affect the re-allocation process when repeated.

3. LITERATURE REVIEW

Numerous mathematical methods have been tested for finding the optimal shortest path for the targeted allocation site. Dynamic environments change over time is normally dealt with via computing access frequency [1-5]. Several dynamically allocating fragments algorithms have reported in literature [3, 6-8]. Most of these algorithms are based on using modified Dijkstra's shortest path techniques which are unfortunately expensive as complexity is O(n^2) just for site selection. It is to be noted that Floyd – Warshall algorithm is hardly used due to its complex implementation effort though might be more efficient [9]. In [3], a thresholding algorithm for re-allocation based on data access patterns with and without time constraints was proposed. Recently, an efficient design technique for cost optimization in distributed database systems has been reported by [9, 10, 15]. This algorithm was based on the threshold technique of [2, 3]. It migrates fragment (Fi) to site (Sj) via optimizing the query cost, average transfer cost, and local access counter over several time intervals. The shortest path technique of Dijkstra was used to move fragment (Fi) to the new location. However, the algorithm of [9] did not consider real-time non-replicated database systems (DBSs). The aim of this work is to extend the algorithm of [9] to a more robust and interactive data assignment model.

4. METHODOLOGY

Our approach consists of the following major phases:

Phase 1: Fragmentation, in a network environment that consists of m sites at time t, where each site has n(t) fragments.

Phase 2: Transmission, using the proposed optimized algorithms. Migration conditions need to be satisfied for fragment Fi to be moved from site Sh to site Sj

Phase 3: Fragments Assignment or allocation, where ach site has two constraints: fragment limit (FL) and site capacity (C).

Phase 4: Non-Replication, to maintains non-replication by ensuring that a fragment is allocated to only one site. Our model adopts a technique called Fragment Preferred Site (FPS) to be calculated on each site to decide on fragment assignment to the most suitable site. The site with the highest FPS value will be the candidate site to host the intended fragment

Compared to the work presented in [2, 3, 13-15], this work will try to propose a new algorithm that can interactively minimize the transmission cost given the site constraints and time dimension. The proposed technique is a potential improvement to the existing similar ones in enhancing the overall performance of the DDBSs.

The proposed model will start by determining the shortest path values by running Dijkstra algorithm used in [12] in every site to obtain the shortest path from one site to every other site. This process is consistently repeated in real time until the shortest path between sites is obtained.

Our model will adapt a technique called Fragment Preferred Site (FPS) to be calculated on each site to decide on fragment replication to the most suitable site. The site with the highest FPS value will be the candidate site to host the intended fragment.

4.1. DDBS ENVIRONMENT

In a network environment that consists of *m* sites at time *t*, where each site has *n(t)* fragments currently distributed as shown in Figure1. Every fragment $F_i(t)$ at site $S_j(t)$ has two variables; the $LAC_i(t)$ (representing the number of local accesses to fragment $F_i(t)$ at site $S_j(t)$, and the $RAC_i(t)$ (representing the number of remote accesses to fragment F_i at site S_j and time *t*). Every site $S_j(t)$ has two constraints: Capacity C_j (indicating that no site will receive more than its capacity) and Fragment Limit FL_j (representing the maximum number of fragments each site can handle). The following migration conditions need to be satisfied for fragment F_i to be moved from site S_h to site S_j at time *t*. The condition for fragment migration for each time step reads:

$$S_j.F_i.RAC > S_j.F_i.LAC, \quad j = 1,2,\ldots, m \qquad (1)$$

$$\sum_{j=1}^{m} QC_h^{\,j} > \sum_{i=1}^{n} QC_h^{\,k}, \ \ k,h = 1,2,\ldots, m\,,\ m <> j \quad (2)$$

with the query cost given by

$$\sum_{j=1}^{m} QChj = DRhj * TChj \qquad (3)$$

and the fragment transmission cost obtained from

$$FTC = \left(\sum\left(\mathrm{Req}_{ij} * \mathrm{TC}_{hj,}\right),\ 1 <= j, h < m\right) \qquad (4)$$

The threshold function at each time step is suggested to be

$$Threshold\ value\ = \left(\sum_{j=1}^{m}\sum_{i=1}^{n} Req_{ij} * TC_{hj}\right) / n \qquad (5)$$

$$FTC > threshold\ value \qquad (6)$$

Equation (1) states that at any time the remote access counter should be greater than the local access counter for the fragment *(F_i)* to be moved. Equation (2) states that the average query cost between site S_h *and* site S_j should be higher than average query cost between site S_h and all other sites accessing fragment F_i (i ≠ j). Equation (3) computes the query cost between site S_h and site S_j. The query cost between sites S_h and S_j can be calculated by multiplying the transmission cost between the two sites (TC_{hj}) by the volume of data being transmitted (DR_{hj}) which represents the data obtained by executing site S_j queries that request a particular data volume from fragment F_i allocated at site S_h. Equation (4) gives fragment transmission cost between sites S_h and site S_j for fragment F_i. Equation (5) calculates the threshold value. Equation (6) states that the fragment transmission cost should be greater than the threshold value. It is to be noted that all these calculations are measured in real-time.

When performing fragment assignments the following constraints have to be complied with at any time throughout the assignment process:

$$\sum_{i=1}^{n} q_{ij} * Z_j < C, \qquad 1 \le j \le m \qquad (7)$$

$$\sum_{i=1}^{n} q_{ij} = 1,\ 1 \le i \le n \qquad (8)$$

$$\sum_{j=1}^{m} q_{ij} < FL, 1 \le i \le n \qquad (9)$$

Equation (7) indicates that no site should receive more than its capacity. Equation (8) maintains non replication by ensuring that a fragment allocation to only one site. Equation (9) puts a limit to the number of fragments each site can handle called fragment limit (*FL*). Table 1 shows the notation used in this work. All variables are time-dependent.

Table1: Our Model Notations

Z	fragment size
FI	accessed Fragment Identifier
QC_h^j	Average of query cost between S_h and S_j
QC_h^k	Average of query cost between S_h and any other site S_k
Req_{ij}	Equal 1, if F_i is required by S_j and 0, otherwise
RF_{rj}	Retrieval frequency of retrieval operation from site j
UF_{uj}	Update frequency of update operation from site j
QF_{ij}	Access frequency of the ith query at site j
V	Volume of fragment allocation i (characters)
SC	Storage cost ($ / 5,OOO char/month)
X_{ij}	Equal 1, if F_i allocated to site S_j and 0, otherwise
TC_{ij}	Cost for site *Si* accessing a fragment located at site *Sj*

4.2. THE ALGORITHM

The algorithm starts by determining the shortest path values by running *Dijkstra* algorithm used in [10] in every site to obtain the shortest path from one site to every other site. This process is consistently repeated in real time until the shortest path between sites is obtained. Having a weighted directed graph *G = (V, E)*, with a weighted function *W: E → R* mapping edges of real valued weights with sites S_1, S_2, S_3 and S_4 (see Figure 1). The shortest path matrix would look like the one in Table 2;

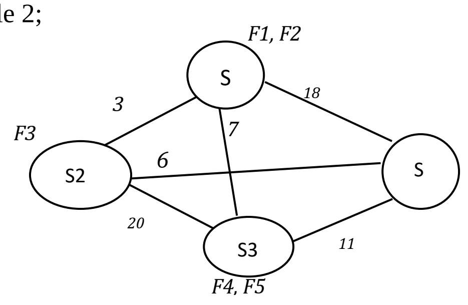

Figure 1: Network Sites at time *t*

Table 2: Shortest Path Matrix

Source Site	Destination Site	Path	Path value
S1	S2	S1----S2	3
S1	S3	S1----S3	7
S1	S4	S1---S2---S4	9
S2	S3	S2---S1---S3	10
S2	S4	S2----S4	6
S3	S4	S3----S4	11

Each fragment is accessed by at least one site. And based on these Site Access Records (*SAR)* shown in *table 7*, the Access Cost Matrix *(ACM*) matrix is constructed as shown in *table 3*. Accesses are represented by the *ACM* matrix. $ACM_{i,j}$ gives the number of times site S_j accesses fragment F_i.

Table 3: Access Cost Matrix (ACM)

S/F	F_1	F_2	F_3	F_4	F_5	F_6
S_1	1	1	0	1	0	0
S_2	1	3	1	0	2	2
S_3	2	0	0	1	1	1
S_4	0	1	2	0	0	2

The transmission cost matrix *TCM (*Table *4)* gives the cost of accessing fragment F_i by site S_j. By multiplying these two matrices (*ACM* by *TCM)*, the Fragment Usage Matrix (*FUM*) is produced. And based on the *FUM* matrix the threshold value is computed. The site with the highest average access cost for fragment F_i will be the candidate site to host that fragment.

Table 4: Transmission Cost Matrix (TCM);

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

5. NON-REPLICATION TECHNIQUE

Our model adopts a technique called Fragment Preferred Site (FPS) to be calculated on each site to decide on fragment replication to the most suitable site. The site with the highest *FPS* value will be the candidate site to host the intended fragment, namely,

$$FPS\ (S_j,F_i) = \sum_{i=1}^{n} (QF_{hi} * QS_{hi}),\ 1 <= j <= m \quad (10)$$

$$SQ_{hi} = \sum_{i=1}^{n} TCM_{hi} + \sum_{i=1}^{n} FUM_{hi},\quad 1 <= h <= h \quad (11)$$

The fragments Preferred Site(FPS) technique gives the number of fragment accesses, i.e. it determinses how many times each site query (SQ) access the intended fragment. This is used if there are accesses for the same fragment F_i from more than one site with the same access cost for that fragment. In this case, If many sites require the same fragment (F_i), then run *FP*S technique at all sites requesting $F_{i,}$ and assign the fragment to site (s_j) having the highest *FPS.*

5.1. MODEL DEMONISTRATION

a. Assuming that there are a number of *n* fragments distributed across *m* sites, and each site can initially contains one or more fragments. Queries may access several fragments allocated at different sites. And each site has two constraints: fragment limit (*FL*) and site capacity (*C*) as shown in Table 5.

b. A separate data structure called Site Access Record (*SAR*) is kept for each site. The *SAR* stores information about fragments accesses at each site, denoted by SAR^k_j , indicating the k^{th} accesses by site S_j, where (*k* = 1,2,3,…to unlimited access numbers, and *j* = 1,2,3,...*m*). *SAR* stores the following information for each access:

1. Accessed Fragment Identifier (*AFI)*, see *table 6*.
2. Accessing Site Address (ASA).
3. Data Record (DR) on executing a query *Q* from site S_i on fragment F_i located at site S_j (in bytes).
4. Access Time (AT) by site S_j to fragment F_i.
5. Access Counter (AC) to keep the number of access times for each accessed site.

c. Also for each site a data structure named Access Counter Record (*ACR*) is kept for every fragment in the site. The *ACR* record stores the following information about the fragment after each access:

i. Candidate Site Address (CSA): The address of the site that incurred a query cost value that is higher than that of all other sites over a time interval (*t1* to *t2*). Initially, *CSA* is set to the address of the site where the fragment is currently located.
ii. The number of local accesses to the stored fragment (*LAC).*
iii. The number of remote accesses to the stored fragment (*RAC*).
iv. The time of the candidate site address selection (*TCS*).

For each locally stored fragment, initialize both the local and remote counters to zero (*LAC* = 0, *RAC* = 0).

5.2. RUNNING EXAMPLE

The following example is to test the validity of our algorithm. In this example there is a network of four sites in which six fragments are initially distributed according to a random assignment method. Our proposed assignment method will be tested based on the information presented in *tables (5 and 6).*

Table 5: Database Fragments Sizes

Fragment	F_1	F_2	F_3	F_4	F_5	F_6
Size	810B	620B	900B	660B	521B	701B

Table 6: Sites Sizes

Site	S_1	S_2	S_3	S_4
Capacity	2350B	1650B	2020B	3200B
Fragment Limit	3	1	3	4

We first apply the minimum algorithm of [9, 12] on the communication cost matrix to obtain the shortest path matrix. Then we keep the shortest paths values based on the given network graph. Then**,** assuming that multiple accesses are performed, the site access record (*SAR*) is constructed (data not shown). And based on the sites access records (*SAR*), the Access Cost Matrix (*ACM*) will be constructed as mentioned earlier and shown in Table 3. The fragment usage matrix is computed by multiplying the *ACM* matrix by the *TCM* matrix (*FUM = ACM * TCM*), *FUM* matrix is presented in Table *7*;

Table 7: Fragments Usage Matrix (FUM)

F/S	S_1	S_2	S_3	S_4
F_1	23	37	25	44
F_2	33	9	54	58
F_3	41	8	38	4
F_4	9	21	9	29
F_5	19	16	32	19
F_6	55	24	54	19

Based on *FUM* matrix, the threshold value can be calculated for fragments and sites individually when needed. Finally, based on threshold values and the access counter records, the migration decision will be made for each fragment as shown in Table *8*.

Table 8: Migration Decision

F_I	Migration (Y/N)	Constraints violations (Y/N)	Threshold value	Migration Fails (F)/successes (S)
1	N	N	24, 34.6	F
2	Y	N	45,5, 32	S
3	N	N	-	F
4	N	N	-	F
5	Y	Y	24, 23.3333	Migration fails because violation of S_2 constraints
6	Y	N	36,75, 32.68	S

6. CONCLUSIONS and FUTURE ASPECTS

In this work, based on a modified Dijkstra shortest path algorithm findings, we proposed an optimized data assignment algorithm for time dependent non-replicated distributed database systems. Using a toy example, we have demonstrated the efficiency and robustness of a proposed real time distributed database system for non-replicated real time environnent. The proposed algorithm is set to be the most efficient and since it has improved the performance through minimizing traffic and modifying shortest path calculation, the algorithm was found to be robust. However, the algorithm lags behind with respect to the storage space factor due to the added complexity which will be dealt with in a future work. In future work, an experimental implementation is going to be extensively made on big datasets in such way to significantly improve and extend the work of [16]. In doing so, we examine algorithm efficiency with other algorithms in literature.

ACKNOWLEDGMENT

The authors would like to thank and appreciate the support received from the Research Office of Zayed University for providing the necessary facilities to accomplish this work. This research has been supported by Research Incentive Fund (RIF) Grant Activity Code: R20056–Zayed University, UAE.

RERFERENCES

[1] Hassan I. Abdalla and Abdel Monim Artoli, “Towards an Efficient Data Fragmentation, Allocation, and Clustering Approach in a Distributed Environment”, Information Journal of MDPI, 10, 112, March 2019.

[2] A. Singh and K.S. Kahlon, "Non-replicated Dynamic Data Allocation in Distributed Database Systems", IJCSNS International Journal of Computer Science and Network Security, vol. 9 no. 9, pp. 176-180, September 2009.

[3] T. Ulus and M. Uysal, “A Threshold Based Dynamic Data Allocation Algorithm - A Markove Chain Model Approach”, Journal of Applied Science, vol. 7(2), pp 165-174, 2007.

[4] Amer, A. Data Replication Impact on DDBS System Performance. Semantic Web Science and Real-World Applications, 1st ed., M. Lytras, N. Aljohani, E. Damiani and K. Chui, Ed. IGI Global, pp. 134-162, 2018.

[5] Nashat, D. and Amer, A. A Comprehensive Taxonomy of Fragmentation and Allocation Techniques in Distributed Database Design. ACM Computing Surveys, 51(1), pp.1-25, 2018.

[6] R. Karimi Adl, S. M. T. Rouhani Rankoohi, “A new ant colony optimization-based algorithm for data allocation problem in distributed databases”, Knowl. Inf. Syst., 20:349–373, 23 Jan 2009.

[7] I. Ahmad, K. Karlapalem, Y. K. Kwok and S. K. So., “Evolutionary Algorithms for Allocating Data in Distributed Database Systems”, Distributed and Parallel Databases, 11: 5-32, 2002.

[8] W.J. Lin and B. Veeravalli, “A Dynamic Object Allocation and Replication Algorithm for Distributed System with Centralized Control,” International Journal of Computer and Application, Vol. 28, no. 1,pp. 26-34, 2006.

[9] Hassan I. Abdalla, Ali Amer and H. Mathkour, “A Novel Vertical Fragmentation, Replication and Allocation Model in DDBSs”, ISI Journal of Universal Computer Science Volume 20, Issue 10, Pages 1469–1487. 2014.

[10] Amer, A., Sewisy, A. and Elgendy, T. An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs). Heliyon, 3(12), p.e00487, 2017.

[11] J. O. Hauglid, N. H. Ryeng, “DYFRAM: dynamic fragmentation and replica management in distributed database systems”, Distributed and Parallel Databases 28: 157–185, 2010.

[12] R. W. Floyd, "Algorithm 97: Shortest Path". Communications of the ACM 5 (6): 345, 1962; S. Warshall, "A theorem on Boolean matrices". Journal of the ACM 9 (1): 11–12, 1962.

[13] F. Castro-Medina, L. Rodríguez-Mazahua, A. López-Chau, I. Machorro-Cano and M. A. Abud-Figueroa, "Design of a Horizontal Data Fragmentation, Allocation and Replication Method in the Cloud," 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), Vancouver, BC, Canada, 2019, pp. 614-621, doi: 10.1109/COASE.2019.8842934.

[14] Amer, A.A., Mohamed, M.H., & AlAsri, K. ASGOP: An aggregated similarity-based greedy-oriented approach for relational DDBSs design. Heliyon, 6. 2020.

[15] Amer, A.A. On K-means clustering-based approach for DDBSs design. J Big Data 7, 31 (2020). https://doi.org/10.1186/s40537-020-00306-9

[16] Hassan I. Abdallaha, Ali A. Amer, Hassan M. Performance optimality enhancement algorithm in DDBS (POEA). Computers in Human Behavior 30, 419-426, 2014.