A PUBLICATION OF MOTHERHOOD UNIVERSITY, ROORKEE
UGC SL.NO. 64498

Motherhood International Journal of Multidisciplinary Research & Development
A Peer Reviewed Refereed International Research Journal
Volume II, Issue II, February 2018, pp. 90-104
ONLINE ISSN-2456-2831

A QUERY DRIVEN SITE CAPACITY CONSTRAINED HEURISTIC SCHEME FOR OPTIMAL ALLOCATION OF FRAGMENTS IN DISTRIBUTED DATABASE SYSTEMS

Vinod Kumar, Professor, Department of Computer Science, Gurukul Kangri Vishwavidyalaya, Hardwar, Uttarakhand, India
Anil Kumar Kapil, Professor, Faculty of Mathematics and Computer Sciences, Motherhood University, Roorkee, Uttarakhand, India

Abstract
In this paper we do not consider replicated allocation of fragments, and leave it as a separate problem because there are efficient solutions [3] that can be applied to replicate fragments once an initial non-redundant allocation scheme is generated. Further the problem of query driven data allocation is considered with the following point of views:

1. Representing and evaluating the set of queries accessing the distributed database system
2. Using this information for the formulation and final solution for the data allocation problem.

1. INTRODUCTION

Distributed database management systems are important because they provide certain performance, reliability and availability. However, a critically important issue is how to process queries requiring data from several locations. In general, satisfying a user request, in a Distributed database environment, involves the following major steps:

1. Accessing the network directly to determine where the requested data is located,
2. Determining an access strategy that specify which copy of data to access and when.
3. Considering the location where the data will be processed.
4. Deciding the mechanism of routing data.

2. LITERATURE REVIEW

The data allocation problem has been first studied in terms of file allocation problem in a multi computer system, and later on as a data allocation problem in distributed database system. The file allocation problem does not take into consideration the semantics of the processing being done on files, whereas it must take into consideration the interdependencies among accesses to multiple fragments by a query. The problem of file allocation with respect to multiple files on a multiprocessor system was first studied by [4]. He presented a global optimization model to minimize overall processing cost under the constraints of response time and storage capacity with a fixed number of copies of each file. Casey [1] distinguished between updates and queries on files. Whereas Eswaran [6] suggested that a heuristic rather than exhaustive search approach is more suitable.

A file allocation problem in the environment of a distributed database was analyzed by Ramamoorthy and Wah [10]. They developed a heuristic approximation algorithm for a simple file allocation problem as well as for the generalized file allocation problem. Ceri [2] considered the problem of file allocation for typical databases applications with a simple model of transaction execution taking into account the dependencies between accesses to multiple fragments.

A data transfer cost minimization model for the allocation of the distributed database to the sites is discussed by Apers [11]. The author came up with a very complicated approach to allocation relations by first partitioning them into innumerable number of fragments, and then allocating them. In the problem addressed by [11], the fragmentation schema is one of the outputs of the allocation algorithm. This curtails the applicability of this methodology when fragmentation schema is already defined and allocation scheme must be generated. Cornell et. al [5] proposed a strategy to integrate the treatment of relation assignment and query strategy to optimize performance of a distributed database system.

There have been many linear programming formulations proposed for data allocation problem [9, 12]. The main problem with these approaches is the lack of modeling of the query execution strategy. Lin et. al [8] also developed a heuristic algorithm for minimizing overall data transfer cost, by considering replicated allocation of fragments and both read and update transactions.

3. NOMENCLATURE & DEFINITIONS

n	Number of fragments in the Distributed database system.
m	Number of sites in the Distributed database system.
c	Capacity of each site.
N_S_I	Number of sites Involved.
NSI	A counter for comparison.
DTC(,)	A matrix of data transfer cost among the sites where each element $C_{i,j}$,i= 1 …n, j= 1 … m, represents data transfer cost between site i and site j.
Capacity ( )	An array which contains the storing capacity of each site.
Alloc ( )	An array which signifies the boolean information of fragment allocation having value 1, if a fragment is allocated to site j, and 0, otherwise.
COST(, 0)	An array containing the indices value of matrix DTC(,) which signifies the fragment number.
COST(, 1)	An array containing the indices value of matrix DTC(,) which signifies the site number.
COST(, 2)	An array containing the actual cost value .
ASSIGN ( )	An array storing the cost values of allocated fragments.
SITE( )	An array storing those site numbers on which the fragments are allocated.
FRAG( )	An array storing the indices of allocated fragments.

3.1 ASSUMPTIONS

The proposed technique is based on the following assumptions:

1. The number of sites and number of fragments are equal.
2. The number of site involvement is the greatest lower bound of (n / c).
3. The fragment, once allocated to a site, can not be allocated to any other site(s).
4. Replication of fragments is not allowed.
5. The sites are fully connected.
6. Each fragment is allocated to at least one site.
7. No site allocates more fragments than the maximum permissible number.
8. The inter site distance assumed to be unity.

3.2 ALLOCATION PROBLEM

Formally, the allocation problem can be stated as follows:

Let F = {F_0 , F_1, ... ,F_{n-1} }be a set of fragments and S = { S_0 , S_1, … ,S_{m-1} } be set of n sites connected by communication network on which a set of applications Q={q_0 , q_1, …, q_{g-1}} are running. A link between two sites S_i and S_j has a positive integer $C_{i,j}$ associated with it, giving the cost of a unit data transferred from site S_i to site S_j. If the two sites are not directly connected by a communication link then the cost for the unit data transferred is given by the sum of the costs of the links of a chosen path from site S_i to S_j. Each query q_g can be executed from any site with a certain frequency. Let $FREQ_{i,j}$ be the frequency with which query q_i is executed from site S_j . These frequencies of execution of queries, at all sites, can be represented as a matrix FREQ(,) of order mxn. A query may access one or more fragments.

3.2.1 Query Execution Strategy

The optimal orderings of binary operations is based on a query execution strategy in distributed databases. A query execution strategy can be :

1. *Move Small:* If a binary operation involves two fragments located at two different sites then ship the smaller fragment to the site of the larger fragment.
2. *Query Site:* Ship all the fragments to the site of origin of query and execute the query.

Here, the objective of the data allocation is (i) to minimize the total data transfer cost to process all the queries by using 'Query Site' query execution strategy (ii) to maximize the locality of the fragments for executing the queries (iii) to incorporate the query execution strategy when a query needs to access fragments from multiple sites and reduce the total data transfer cost to process all the queries.

For example in Figure 3.2, relations E and G are located at different sites then it will incur Size (E') data transfer cost, if Site(Q) is different from the site where relation E is located.

The inputs to the data allocation problem are

1. A set of n fragments F ={F_0, F_1, F_2, …., F_{n-1} }.
2. A set of m sites S = {S_0,S_1, S_2,….,S_{m-1} } and a matrix UDTC (,) = [$C_{i,j}$] containing the cost of transporting a unit of data from site S_i to site S_j for each i, j.
3. Q= {q_0,q_1,…,q_{g-1}},a set of g queries and a matrix FREQ= [$FREQ_{i,j}$] containing the frequency of q_j initiated at S_i, for each i,j.
4. The fragment dependency graph, for each of the queries along with the estimates of the intermediate sizes.
5. A vector V = [V_j], having the limit on maximum number of fragments that can be allocated at site S_j. This models the storage constraint of each site in data allocation.

On the basis of the above inputs we develop a cost model for total data transfer incurred to process all the queries.

4. COST FUNCTIONS OF DATA TRANSFER IN DDBMS

We define an allocation of fragments, as the optimal allocation, that optimizes the total data transfer cost within the constraints. So, it is desirable to know the size of data for every fragment that may be required by any site for processing a query. The fragments located at different sites may be of different sizes. Thus, in the fragment dependency graph for 'Query Site' query processing strategy, the size of the data of a fragment required by query site does not vary with the location of other fragments, since there is no dependency between the fragments accessed by the query.

Let $r_{i,j}$ be defined as the size of the data of F_j , needed to be transported to the site where query q_i is initiated. The corresponding matrix R is of order gxn. Let the frequency of query q_i, initiated at site S_j, be $FREQ_{i,j}$. And let the query q_i request for the fragment F_k and each request require $r_{i,k}$ amount of data transfer from the site where F_k is located. The amount of data, needed to be transferred from the site where fragment F_k is allocated to the site S_i where the query is initiated, is given by matrix FREQ(,) of order mxn.

Thus, the amount of data transfer for query q_i , from site S_j, can be expressed as

$$ADT = \sum_{j=0}^{n-1} FREQ_{i,j} * r_{i,j} \qquad \textbf{(1)}$$

The total data transfer cost is:

$$ADTC = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} C_{site(Fk),i} * ADT_{i,j} \qquad \textbf{(2)}$$

The communication cost $C_{i,j}$ represents the communication in terms of bytes transferred, between the site (F_k) and site(F_i).

5. THE PROPOSED METHOD AND ALGORITHM

A fragment is allocated to a site in such a way that extensive data transfer cost is avoided and the capacity of the site suits to the execution environment of the system. The proposed algorithm involves stepwise refinement of matrix of Data Transfer Cost (DTC (,)) among the sites, an array storing those site numbers on which the fragments are allocated (SITE ( )) and an array containing those fragment numbers, which get allocated (FRAG ( )) during allocation process of m fragments to n sites. These fragments are assigned to the sites in such a way that the total data transfer cost remains minimum.

A brief description of the proposed method is as follows:

1. Arrange the DTC (,) in ascending order and store in an array COST (, 2). Store row and column indices into array COST (, 0) and COST (, 1) respectively.

   Begin with the first element of the COST (, 2), COST (, 0) and COST (, 1), and assign to arrays ASSIGN ( ), SITE ( ) and FRAG ( ) respectively.

2. Proceed with next values of COST (, 0) and COST (, 1) that give rise to two cases:

2.1 Check the SITE ( ) and FRAG ( ) values corresponding to the previously assigned values. If COST (, 1) differs the previous value and COST (, 0) equals the previous value (it indicates that capacity is available and fragment get allocated), store these values to arrays SITE ( ) and FRAG ( ) respectively, and store the actual data transfer cost of the fragment into ASSIGN(,). Update the information regarding site and fragment accordingly.

2.2 Check the SITE ( ) and FRAG ( ) values corresponding to the previously assigned values. If value of COST (, 0) and COST (, 1) differ than their previously assigned values (it shows that capacity is available, allocation is not made yet) store these value to SITE ( ) and FRAG ( ) respectively, and store the actual data transfer cost of the fragment into ASSIGN(,). Update the information regarding site and fragment accordingly.

3. When both of the above cases fail then search a site in the array SITE ( ). If the site indices match with the corresponding indices of COST (, 0), check its capacity, if available, allocate the fragment to the site and update the information regarding SITE ( ) and FRAG ( ).
4. Continue the above process till all the fragments are allocated.
5. Get the total data transfer cost by summing the values in the array ASSIGN ( ).

6. IMPLEMENTATION OF THE ALGORITHM

Consider a distributed database system with 4 fully connected sites S_0, S_1, S_2 and S_3 and four relations E, F, G and H. Let there be only one query. Let this query be initiated from site S_0 with frequency 2, from site S_1 with frequency 3 , from site S_2 with frequency 4 and from site S_3 with frequency 1.

Let the sizes of the intermediate fragments be Size (E') = 10, Size(F')=15, Size(G')=25 and Size(H')= 5. Since there is only one query the corresponding matrix R=[10, 15, 25, 5] and the matrix FREQ =[2, 3, 4, 1] showing frequencies of the query to the corresponding sites S_0, S_1, S_2, and S_3. Let the limit vector V=[2, 2, 2, 2]. Then the amount of data transfer can be expressed as:

ADT= FREQ *R

i.e.

$$\text{ADT}(,)= \begin{array}{c|cccc} & \mathbf{E} & \mathbf{F} & \mathbf{G} & \mathbf{H} \\ \hline \mathbf{S_0} & 20 & 30 & 50 & 10 \\ \mathbf{S_1} & 60 & 45 & 75 & 15 \\ \mathbf{S_2} & 40 & 60 & 100 & 20 \\ \mathbf{S_3} & 10 & 15 & 25 & 5 \end{array}$$

$$\text{UDTC}(,) = \begin{array}{c|cccc} & \mathbf{S_0} & \mathbf{S_1} & \mathbf{S_2} & \mathbf{S_3} \\ \hline \mathbf{S_0} & 0 & 2 & 5 & 4 \\ \mathbf{S_1} & 2 & 0 & 3 & 1 \\ \mathbf{S_2} & 5 & 3 & 0 & 2 \\ \mathbf{S_3} & 4 & 1 & 2 & 0 \end{array}$$

Then, the data transfer cost among the sites is given by

$$\text{DTC} = \text{UDTC}(,) * \text{ADT}(,)$$

i.e.

$$\text{DTC}(,) = \begin{pmatrix} 0 & 2 & 5 & 4 \\ 2 & 0 & 3 & 1 \\ 5 & 3 & 0 & 2 \\ 4 & 1 & 2 & 0 \end{pmatrix} * \begin{pmatrix} 20 & 30 & 50 & 10 \\ 60 & 45 & 75 & 15 \\ 40 & 60 & 100 & 20 \\ 10 & 15 & 25 & 5 \end{pmatrix}$$

$$= \begin{array}{c|cccc} & E & F & G & H \\ \hline \mathbf{S_0} & 360 & 450 & 750 & 150 \\ \mathbf{S_1} & 160 & 255 & 425 & 85 \\ \mathbf{S_2} & 300 & 315 & 525 & 105 \\ \mathbf{S_3} & 220 & 285 & 475 & 95 \end{array}$$

Step 1: BEGIN

Input **: 4, 4, 2 ,**

$$DTC(,) = \begin{array}{c|cccc|} & \textbf{E} & \textbf{F} & \textbf{G} & \textbf{H} \\ \textbf{S}_0 & 360 & 450 & 750 & 150 \\ S_1 & \textbf{160} & \textbf{255} & \textbf{425} & \textbf{85} \\ S_2 & \textbf{300} & \textbf{315} & \textbf{525} & \textbf{105} \\ S_3 & \textbf{220} & \textbf{285} & \textbf{475} & \textbf{95} \end{array}$$

Step 2: **NSI = 1**

```
    l=1
    For i=0 to 15 Do
    begin
     Capacity (,) =2
      Alloc(,) =0
       end;
```

Step 3: Check the no of site involvement which is N_S_I =2

Step 4: f=4*4

```
    p=0
    For i=0 to 3 Do
     For j=0 to 3 Do
     begin
      If(p<=f)
      COST(p,0)=i
     COST(p,2) = DTC(i,j)
    p = p+1
   end;
```

```
Step 5: For i=0 to 15 DO
        For j=1 to 16 Do
          begin
                Step 5.1: Arrange the values of COST(p,2) in ascending order and swap
                the values of COST(p,0) and COST(p,1) accordingly.

                COST(p,0) = (1,3,…,0,2)
            COST(p,1) = (3,3,…,2,2)
            COST(p,2) = ( 85,95,…,525,750 )
      end;
```

Step 6: After assigning the first element of COST(p,2) to ASSIGN( ) , COST(p,0) to SITE( ) and COST(p,1) to FRAG( ) we get

ASSIGN( ) = {85}

SITE( ) ={1}

FRAG( ) = {3}

Update the other information to the corresponding site and fragments accordingly.

```
Step 7: For j=1 to 16 Do
        begin
          After finding the first element search the remaining element, update and store their
          corresponding information accordingly.
        Step 7.1: Check whether l == n ?
                If yes. Goto step 9 otherwise continue with next step 7.2
        Step 7.2: Check whether NSI < N_S_I
                If yes.  goto step 7.3 otherwise goto step 7.6
        Step 7.3, 7.4 & 7.5: The allocations obtained after these steps are
```

Site	Fragment	Cost
1	0	160
3	1	285

and

ASSIGN( ) = { 85, 160, 285 }

SITE( ) = { 1, 1, 3 }

FRAG ( ) = { 3, 0, 1 }

Step 7.6: Search the COST(p,1) for the remaining unallocated fragment (s) and goto step 7.7

Step 7.7&7.8: The allocation obtained after these steps is :

Site	Fragment	Cost
3	2	475

and

ASSIGN( ) = { 85, 160, 285, 475 }

SITE( ) = { 1, 1, 3, 3 }

FRAG ( ) = { 3, 0, 1, 2 }

end;

Step 8: Total_Cost =0

For i=0 to 4 Do

Total_Cost = 0 + (85 + 160 + 285 + 475)

Total_Cost = 1005

Step 9: Stop

7. CONCLUSION

The Ford and Fulkerson Algorithm, which has earlier been applied for the solution of this problem [7] using Max-Flow Min–Cut approach, first generates a graph of ($M = m + n + 2$) nodes involving ($N = m + mn + n$) edges. The solution steps also depend on the maximum capacity C of all the edges involved in the graph resulting into time complexity of O(MNC). It can be further approximated as $O[(6m^4 + 8m^3) C]$ assuming equal number of fragments and sites.

On the same scale, the time complexity of our algorithm is $O(m^4)$ which is much lower as compared to that mentioned above. Further, for the present day high capacity networks, the earlier algorithm becomes totally uncontrollable because its complexity depends on the capacity. Thus, our algorithm is faster as well as simpler as it involves only comparison, assignment and just m addition operations.

The complexity comparison is given in Table 1 and Graph 1

Table 1 : Time Complexity Comparison

Size (F,S)	**Earlier Method [KARL 97]** $O[(6m^4 + 8m^3)]$	**Present Method** $O(m^4)$
4,4	2048	256
5,5	4750	625
6,6	9504	1296
7,7	17150	2401

The complexity comparison graph is given below:

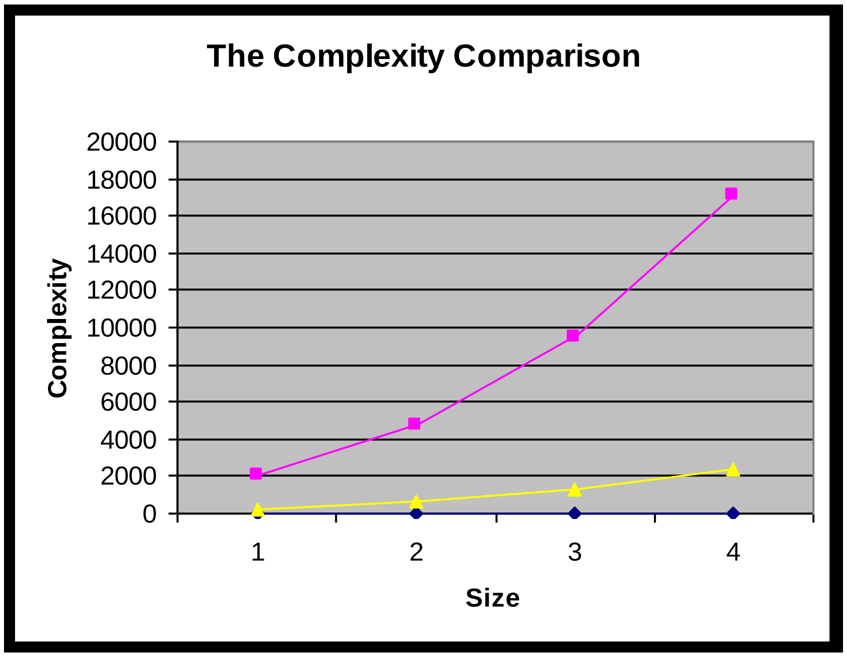

Figure 1 :Time Complexity Comparision

REFERENCES

1. Casey R.C., “Allocation of copies of a file in an information network”, *In Proceedings of Spring Joint Computer Conference, IFIPS, pp. 617-625, 1972.*

2. Ceri S., Martella G. and Pelagatti, “Optimal file allocation for a distributed on a network of minicomputers”, *In Proceedings of International Conference on Database, Aberdeen ,pp., 345-357, July 1980.*

3. Ceri S., Martella G. and Pelagatti G., “Optimal file allocation in a computer network : A solution method based on the knapsack problem”, *Computer Network, vol.6, no. 5, pp. 345-357, 1982.*

4. Chu W.W., “Optimal file allocation in multiple computer system”, *IEEE Transactions on Computers, C-18(10), 1969.*

5. Cornell D. W and Yu P. S., “Site assignment for relations and join operations in the distributed transaction processing environment”, *In Proceedings of IEEE International Conference on Data Engineering, Feb, 1988.*

6. Eswaran, K. P. “Placement of records in a file and file allocation in a computer network”, *Information Processing, pp. 304-307, 1974.*

7. Karlapalem K and Ng M. P, “Query driven data allocation algorithms for distributed database systems”, *In Proceedings of Int. Conference on Database and Expert Systems Applications, pp. 347-356, sep 1997.*

8. Lin X. –M, Orlowaska M. E., and Zhang Y.-C , “Database placement in communication networks for minimizing the overall Transmission cost”, *Mathematical and Computer Modeling, 19(1): 7-19, Jan 1994.*

9. Ram S. and Marsten R.E., "A model for database allocation incorporating a concurrency control mechanism", *IEEE Transactions on Knowledge and Data Engineering, vol. 3, No.3, pp. 389-395, 1991.*

10. Ramamoorthy C. V. and B. Wah, "The placement of relations on a distributed relational databases", *In Proceedings of first International conference Distributed Computing systems, Huntsville, Alabama, September – October, , pp 642-649, 1979.*

11. Apers P.M.G., "Data Allocation in Distributed Database", *ACM Transactionsactions on Database Systems, 13-C37:263-304, September 1988*

12. Gavish B. and Pirkul H.,"Computer and database location in distributed computer systems", IEEE Transaction on Computers, C-35(7) : 583 – 590, 1986.