International Journal of Research in Engineering and Innovation
(IJREI)
journal home page: http://www.ijrei.com
ISSN (Online): 2456-6934

International Journal of Research in Engineering and Innovation
ISSN: 2456-6934
Vol-2, Issue-3
International Journal of Research in Engineering and Innovation (IJREI)

An improved algorithm for query driven data allocation in distributed database systems

Vinod Kumar[1], Anil Kumar Kapil[2]

[1]*Professor, Department of Computer Science, Gurukul Kangri University, Haridwar.*
[2]*Professor, Faculty of Mathematics and Computer Sciences, Motherhood University, roorkee, Uttarakhand*

Abstract

A major component of the query execution cost is the data transfer cost. That is, the total cost involved in moving the data from the sites where it is located to the site where the query is issued. Therefore, to reduce the query execution cost it is desirable to minimize the data transfer cost. This can be possibly achieved by optimally allocating the database objects or fragments to the various sites of the DDBMS.

Keywords: Data transfer, Distributed database system, Algorithm.

1. Introduction

Distributed database systems have been a phenomenal success in terms of facilitating organization and processing of large volume of data. Distributed relational database technology is nearly two decades old and have developed the main idea of maintaining consistency of distributed data and querying data. In distributed database system, a query requires data to be accessed from one or more sites. The cost of executing the query depends on the location of the query as well as the data. The data locality of a query determines the amount of data transfer incurred in processing the query, the higher the data locality the lower the data transfer costs. Thus, one is faced with the data allocation problem, the aim of which is to increase the data locality.
The query processing consists of decomposing the queries and data localization. The data localization involves: identifying the fragments accessed by the query and generating the query operator tree. These generated query operator trees are further processed by taking into account a given query execution strategy and the information of the fragment size to generate a fragment dependency graph.
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 Chu [6]. 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.

1.1 Literature Review

A file allocation problem in the environment of a distributed database was analyzed in Ramamoorthy and Wah [12]. They developed a heuristic approximation algorithm for a simple file allocation problem as well as for the generalized file allocation problem.
Ceri et al. [1] considered the problem of file allocation for typical database applications with a simple model of transaction execution taking into account the dependencies between accesses to multiple fragments. The Knapsack problem solution [2] and branch and bound techniques were further adopted to solve the problem. Chang [5] developed a theory of fragment allocation and designed a network flow algorithm to solve the database allocation problem. In [13] limited processing and network capacities are assumed, transmission of the result of a query to the results site is not

considered; only transmission between fragments are considered, rendering the problem into cluster analysis problem.

Ceri proposed a simple method that ignores replication at the beginning and finds an optimal non–replicated solution. Then replication is handled by applying greedy algorithm that tries to improve the initial feasible solution [3, 4].

Cornell and Yu [7] proposed a strategy to integrate the treatment of relation assignment and query strategy to optimize performance of a distributed database system. Though they do take into consideration the query execution strategy, the solution, they came up with, is a complicated linear programming solution. The main problem in their approach is the lack of simplicity in both incorporation of the query execution strategy and the solution procedure.

There have been many linear programming formulations proposed for data allocation problem [11]. The main problem with these approaches is the lack of modeling of the query execution strategy. Lin et al. also developed a heuristic algorithm for minimum overall data transfer cost, by considering replicated allocation of fragments and both read and update transactions [10]. The optimization heuristic iterates between finding minimum cost query strategy and minimum cost data allocation until local minimum for the combined problem is found. A new class of query optimization algorithms, known as iterative dynamic programming for query optimization was suggested by Donald and Stocker [8].

If all of the data required by a retrieval request is located at the requesting site then only local processing is needed. However, if some data is not located at the requesting site, then data must be accessed from and possibly processed at, other sites. A typical user can access the complete database from any site, a DDBMS processes and executes a user's query by accessing the data from multiple sites.

A major component of the query execution cost is the data transfer cost. That is, the total cost involved in moving the data from the sites where it is located to the site where the query is issued. Therefore, to reduce the query execution cost it is desirable to minimize the data transfer cost. This can be possibly achieved by optimally allocating the database objects or fragments to the various sites of the DDBMS. In the model presented here, we do not consider replicated allocation of fragments. The basic aspects of the problem are the same as discussed in the previous chapter. However, in the solution process 'Move Small' query execution strategy has been adopted.

2. 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.
NSI	Number of site involvement.
UDTC(,)	A matrix of unit cost of data transfer among the sites.
FRAG( )	An array storing the size of each fragment.
R(,)	A data transfer relation size matrix.
FREQ( )	A matrix storing the access frequencies of the queries at the given site.
Sc ( )	An array storing site combinations.
site(F_i)	Site of allocation of fragment F_i .

3. Assumptions

The present method is based on the following assumptions:

1. The number of sites is equal to the number of fragments.
2. The number of site involvement, is the greatest lower bound of (n/c).
3. Replication of fragment is not allowed, that is the allocation policy is static.
4. The sites are fully connected.
5. Each fragment is allocated to at least one site.
6. The number of fragments allocated to each site does not exceed the maximum capacity of that site.
7. The inter site distance is unity.

4. Allocation Problem

Let there be a set of fragments F = {F_0, F_1, ………….…, F_{n-1} } and a network consisting of sites S = { S_0, S_1, …, S_{m-1} } on which a set of applications Q={q_0, q_1, …, q_{g-1}} are running. Let the m sites be connected by communication network. A link between two sites S_i and S_j has a positive integer $C_{i,j}$ associated with it giving the cost for 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 executions of queries at all sites can be represented by a matrix FREQ (,) of order mxn. A query may access one or more fragments.

4.1 Query Execution Strategy

The optimal orderings of binary operations is based on a 'Move Small' query execution strategy in distributed databases. Which can be stated as:

"If a binary operation involves two fragments located at two different sites then ship the smaller fragment to the site of the larger fragment".

Here, the objective of the data allocation is (i) to minimize the total data transfer cost to process all the queries by using 'Move Small' 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.

4.1.1 Evaluation of Query

The construction of operator tree is an essential starting step for the evaluation of a query. An operator tree is a tree in which a leaf node is a relation stored in the database, and a nonleaf node is an intermediate relation produced by a relational algebraic operator. The sequence of operations is directed from the leaves to the root, which represents the answer to the query.

Example 3.1: Let us consider the Example 3.1 again where a query

"Find the names of employee other than J. Doe who worked on the CAD/CAM project for either one or two years." is to be performed by accessing relations E{Eno, Ename, Title}, G{Eno, Jno, Resp, Dur} and J{Jno, Jname, Budget}.

It can be processed as given below:

```
SELECT Ename
FROM J,G,E
WHERE G.Eno=E.Eno
AND G.Jno=J.Jno
AND Ename<> "J.Doe"
AND J.name= "CAD/CAM"
AND (Dur=12 OR Dur =24);
```

The intermediate relations, generated after query restructuring phase, are J', G', E', G" and J". In case of 'Move Small' query execution strategy the corresponding fragment dependency graph is generated by shipping small fragments to the larger fragments site. The fragment dependency graph represents the fragment-nodes (like Site(J), Site(G), etc) i.e. a site where the fragments are located, and a query node Site(Q), where the query is initiated (i.e. query site). A cost value is attached to each edge of the graph corresponding to the amount of data that may be transferred if the fragments corresponding to the two nodes of the edge are located at different sites, or the location of the fragment and querying site are different. For example, in Figure 4.2, relations E and G are located at different sites then it will incur Size (E') data transfer cost to process the join between relation E' and G' when using 'Move Small' query execution strategy.
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 = [$C_{i,j}$] depicting the cost of transporting a unit of data from site S_i to site S_j.
3. A set of g queries Q= {q_0, q_1, …, q_{g-1}} and a matrix FREQ= [$FREQ_{i,j}$] showing the frequency of q_j initiated at S_i.
4. A matrix D giving the amount of data needed from a fragment to be transported to the site of another fragment is derived from the fragment dependency graph.
5. A matrix R = [R_1, R_2 ,…., R_n] where each element corresponds to the size of a relation.
6. A vector V = [V_j] 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.

5. Cost functions of data transfer in DDBMS

The fragment dependency graph of every query processing strategy models two types of data transfer cost. The first type of cost is due to moving the data from the sites where the fragment is located to the site where the query is initiated. The second type of cost is due to moving the data from the site where one fragment is located to the site where another fragment is located. In this case, the size of the fragment required by query site does not vary with the location of other fragments as there is no dependency between the fragments accessed by the query.
Let $r_{i,j}$ be defined as size of data of F_j needed to be transported the site where query q_i is initiated. Let the frequency of query q_i, initiated from site S_j, be $FREQ_{i,j}$ in a unit time interval. And let q_i request for F_k and each request require $r_{i,k}$ amount of data transfer from the site where F_k is located. So, the amount of data, need to be transferred from where fragment F_k is allocated to the site S_i where the queries are initiated, is given by matrix FREQ (,) of order mxk.. Then amount of data transfer can be calculated as:

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

The total data transfer cost

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

The communication cost $C_{i,j}$ represents the communication in terms of bytes transferred, between the site (F_k) and site(F_i).
The second type of data transfer cost incurred in the 'Move Small' query processing strategy is transporting the fragment from one site to the other site in order to perform the binary operations (i.e., join, union etc). In this case, the amount of data of a fragment required by a site varies with the allocation of other fragments.
Let $d_{\lambda,\lambda'}$ define the size of data from fragment F_λ that needs to be transported to the site where $F_{\lambda'}$ is located so as to execute some binary operation. Let the corresponding matrix, D(,), be of order kxk. But this is dependent on the query that is to be processed. Therefore, the query accesses both the fragment and extract the information about how much data needs to be transferred from site where one fragment is located to the site where another fragment is located. This information is extracted by the fragments dependency graph generator which processes the query operator trees on fragments by applying a query execution strategy and represents it in the fragment dependency graph.
Let $d^{\gamma}_{\lambda,\lambda'}$ be the size of data of F_λ need be transported to the site

where $F_{\lambda'}$ is located to process q_γ . And let the corresponding matrix be D^γ. Then the amount of data that need be transported from the site of F_λ to the site of F_λ' is given by:

$$d_{\lambda,\lambda'} = \sum_{j=0}^{k-1}\left(\sum_{i=0}^{n-1} FREQ_{i,j}\right) * d^\gamma_{\lambda,\lambda'} \qquad (3)$$

Let site (F_λ) denote the site where fragment F_λ is located. Then the total transportation cost, t, is given by:

$$t=\sum_{\lambda=0}^{k-1}\sum_{\lambda'=0}^{k-1} C_{site(F\lambda),\, site(F\lambda')} * d_{\lambda,\lambda'} + \sum_{i=0}^{m-1}\sum_{\lambda=0}^{k-1} C_{site(F\lambda),\, i} * ADT_{i,\lambda} \qquad (4)$$

where the first term gives the data transfer cost incurred to process the binary operation between the fragments located at different sites, and second term gives the data transfer cost incurred to transfer the results of the binary operations of fragments to the site where the query is initiated.

6. The proposed method and algorithm

In ‘Move Small’ query processing strategy, the smaller fragment is moved to the site where larger fragment is located, to process the binary operation of the query. In such a query processing strategy the data transfer cost will play a major role for incurring the query processing cost. The optimal fragment allocation scheme can minimize this incurred query processing cost. A fragment is allocated to a site in such a way that extensive data transfer cost is avoided and the capacity of the site suit to the execution environment of the system. The proposed algorithm involves stepwise refinement of Data Transfer Cost among the sites by applying the recursive method, an array storing those site numbers on which the fragments are allocated sc(), and an array FRAG( ), containing the sizes of their fragments during allocation process of m fragments to n sites. The frequency of each query is stored in an array FREQ( ). These fragments are assigned to the sites in such a way that the total data transfer cost remains minimum.

6.1 Formal Algorithm

```
FUNCTION:

Site_Comb (sc(), start, m , k , NSI )
Site_Perm (sc(), NSI, i)
Swap(sc, i ,j)
Cost_Cal (sc, NSI, i)

/* Main Program */
 BEGIN
 Read(n)
 Read(m)
 Read(c)
 If (n% c !=0)
 NSI = int (n/ c) +1
 Else
 NSI = int (n / c)
 For i=1 to m Do
 For j =1 to m Do
 Begin
 Read ( UDTC(i,j) )
 end
 For i=1 to n  Do
 begin
 Read(  FRAG(i) )
 end
 For j=1 to m Do
 begin
 Read ( FREQ (j) )
 end
 For j=1 to m  Do
 begin
 Read( R ( j ) )
 end
 For i=1 to m Do
 begin
 sum=sum + FREQ(i)
 end
 END  /* End of Main Program */
/* Function return the Site Combination */
Site_Comb (sc(), start, m, k, NSI)
 BEGIN
 If (k>NSI)
 begin
 Site_Perm(sc, NSI, 1)
 Return
 end
 For i=start to m  Do
 begin
 sc(k) = i
 Site_Comb(sc, i+1, m, k+1, NSI)
 end
 END
 Site_Perm (sc(), m, i)     /*Function return the Site
 Permutation */
 BEGIN
 If( i= m )
 begin
 Cost_Cal(sc,  m, 1)
 end
 else
 For j=1 to m  Do
 begin
 Swap(sc, i, j)
 Site_Perm(sc, m, i+1)
 Swap(sc, i, j)
 end
 END /* End of  Funtion*/
 Swap (sc(), i, j)   /* Swap Function */
 BEGIN
 t=sc(i)
```

```
sc(i) = sc(j)
sc(j) = t
END
 Cost_Cal (sc(), m,  i)    /* Cost Calculation */
BEGIN
Total_cst=0
For i=1 to m  Do
For j=i+1 to m  Do
begin
adt = sum*FRAG ( sc(i) )
trnsf_cst = UDTC ( sc(i), sc(j)) * adt
qry_site=sum – FREQ(sc(n))
qml = qry_site*r (m)
total_cst = total_cst+ trnsf_cst +qml
end
END.
```

6. Implementation of the algorithm

Consider a distributed database system with 3 fully connected sites S_1, S_2, S_3 and three relations E, G and J. Let the sizes of the fragments be size (E') =5, size (J'')= 30, size (G'') = 25. First, we shall use the first level of the fragment dependency graph i.e. edge from Site (j) → Site (Q) to solve the initial allocation. As there is one query, we have the matrix R=[0, 0, 30] with each element corresponding to relations E', G", and J' respectively, and matrix FREQ = [3, 2, 1] with each element corresponding to the sites S_1, S_2, S_3 respectively. Let the cost matrix for unit data transfer cost from one site to another site be:

$$UDTC(,) = \begin{array}{c|ccc} & S_1 & S_2 & S_3 \\ \hline S_1 & 0 & 2 & 5 \\ S_2 & 2 & 0 & 3 \\ S_3 & 5 & 3 & 0 \end{array}$$

The matrix D(,) giving the amount of data needed from a fragment to be transported to the site of another fragment, is derived from the fragment dependency graph.

$$D(,) = \begin{array}{c|ccc} & E & G & J \\ \hline E & 0 & 5 & 0 \\ G & 0 & 0 & 25 \\ J & 0 & 0 & 0 \end{array}$$

By applying the equation 3, we get the total amount of data that must be transferred from the site where one fragment is located to the site where another fragment is located as:

$$D(,) = (3+2+1) * \begin{bmatrix} 0 & 5 & 0 \\ 0 & 0 & 25 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 30 & 0 \\ 0 & 0 & 150 \\ 0 & 0 & 0 \end{bmatrix}$$

The total data transfer cost given by equation 4.4 for the initial allocation (S_2, S_1 ,S_2) is (2*30 +2*150)+(2+1)*30 = 450. The cost value in first parenthesis corresponds to the data transfer cost incurred in transferring E' to site (G) and G" to the site (J). The second cost value corresponds to the data transfer incurred in transferring the result J' to the query site.
We now apply the recursive method to improve the initial solution so as to reduce the total data transfer cost if possible, Table 4.1 shows all feasible allocation schemes and the data transfer cost incurred for each of them.

7. Conclusion

'Move Small' query execution strategy involves two steps first an allocation is to be found and then it is refined to become an optimal allocation to minimize data transfer cost. In [9] the first step is achieved by applying the Max Flow-Min Cut approach. The allocation thus obtained, is used to improve the data transfer cost and fragment allocation by applying Hill Climbing Algorithm. The overall time complexity of the algorithm presented in [9] is approximated as $O(m*(m-1))^{m-1}$ assuming equal number of fragments and sites.
On the same scale, we first use the algorithm developed in the previous chapter for 'Query Site' query execution strategy and further use a recursive technique for 'Move Small' query execution strategy, to improve the data transfer cost and get the optimal allocation of fragments. The time complexity of our algorithm is calculated as $O(m^4+(m+1)!)$ which is much lower as compared to that mentioned above. The time complexity comparison is shown in Table 1 and graphically depicted in Figure 1.

Table 1: Time Complexity Comparison

Size (n, m)	Earlier Method [KARL97] $O((m*(m-1))^{m-1})$	Present Method $O(m^4+m+1!)$
3,3	36	240
4,4	1728	376
5,5	160000	1345
6,6	21600000	6336
7,7	5489031744	42721

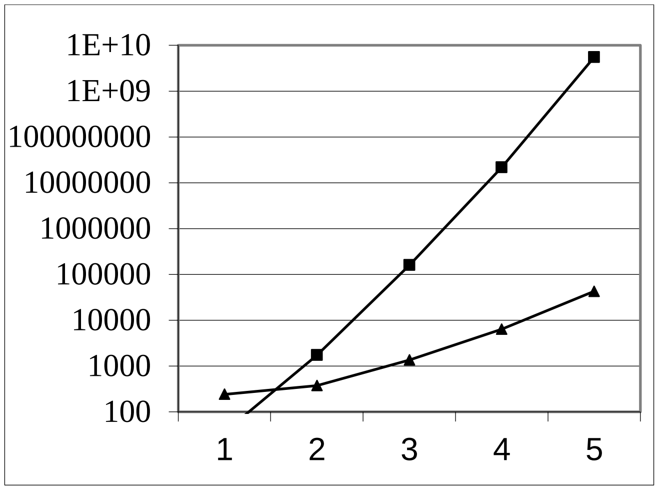

Figure 1: Time Complexity Comparison

References

[1] 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.

[2] 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.

[3] Ceri S., Navathe S. B. and Wiederhold G., "Distributed design of logical database schemes", IEEE Transactions on Software Engineering, vol. 9, No. 4, pp. 487-503, 1983.

[4] Ceri S. and Pernici B, "DATAID-D : Methodology for distributed database design", Computer Aided Database Design, Amsterdam: North-Holland, pp. 157-183, 1985.

[5] Chang S. K. and Liu A.C, "File allocation in a distributed database", International Journal of Computer Information Sciences, vol. 11, no. 5, pp. 325-340, 1982.

[6] Chu W.W., "Optimal file allocation in multiple computer system", IEEE Transactions on Computers, C-18(10), 1969.

[7] 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.

[8] Donald Kossman and K. Stocker, "Iterative Dynamic programming: A new class of query optimization algorithms", ACM Transactions on Database Systems, vol. 25, no. 1, pp. 43-82, March 2000.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Sacco G., "Distributed query evaluation in local area networks". IEEE Data Engineering Conference, pp. 510-516, 1984.