AN INTEGRATED STRATEGY FOR DATA FRAGMENTATION AND ALLOCATION IN A DISTRIBUTED DATABASE DESIGN

ISMAIL OMAR HABABEH
School of Computing
Leeds Metropolitan University
E-mail
ismail.hababeh@uaeu.ac.ae

NICHOLAS BOWRING
School of Computing
Leeds Metropolitan University
E-mail
N.Bowring@lmu.ac.uk

MUTHU RAMACHANDRAN
School of Computing
Leeds Metropolitan University
E-mail
M.Ramachandran@lmu.ac.uk

ABSTRACT

A distributed database is structured from global relations, fragmentation and data allocation. A global relation can be divided into fragments and each fragment may itself contain a relation. The fragmentation describes how each fragment of the distributed database is derived from the global relations. The data allocation allows the allocation of discrete sets of fragments to the sites of the computer network supporting the distributed database.

The objective of the present work is to develop a strategy for distributed database design that is simple and useful to achieve the objectives of data fragmentation, allocation, and replication. It has been designed to fragment and allocate data in a distributed relational database system using different types of computers on a network.

KEY WORDS

Data partitioning, segments, clustering, fragments, benefit value, data allocation.

1. INTRODUCTION

A distributed system is a collection of independent computers that appear to the users of the system as a single computer [1]. The trend in the computer field is toward decentralization. The driving force governing the movement away from centralized toward distributed systems is that they have better performance than a single large centralized system [2]. Academic, industrial and governmental organizations have been using distributed databases to support their needs. This use was accelerated by the advance in telecommunication systems and satisfied the geographical dispersed information.

This paper presents an approach for fragmentation and allocation of data in a distributed relational database and shows a way of grouping sites into clusters to which fragments would be allocated.

In this approach, the database relations will be partitioned into pair-wise disjoint fragments, which will be allocated to clusters and their respective sites according to an allocating algorithm. This approach describes a method to minimize the transactions communication cost by distributing the database relations over the sites, and increasing data availability and integrity by allocating multiple copies of the same database fragments over the sites.

2. BACKGROUND

Existing distributed database methodologies are limited in their theoretical and implementation parts. They don't deal with distributed database issues separately, don't optimize transaction response time, don't test their performance on different types of network connectivity, and present exponential time of complexity.

Various strategies have already been described that effectively partition data across distributed systems. Naturally, there are benefits and drawbacks to all schemes. Minyoung and Yang-sun [3] have proposed a methodology for partitioning and allocating data effectively over a network for PC-based distributed database design. The researchers present a cost model and propose a heuristic procedure for merging mixed fragments (grid cells), based on the joint cost and the frequencies of the transactions accessing the cells. The purpose of merging cells is to minimize the global transaction processing cost. Because the sequence of attributes has no meaning in a relation, the possible combinations of horizontal merging can be minimized to a time computation of complexity bounded by $n(n-1)/2$ instead of 2^{n-1}.

Navathe, Karlapalem, and Minyoung [4] have presented a methodology for generating a mixed fragmentation scheme (horizontal and vertical) for the initial distributed database design phase. They form a grid on a relation, which suggests all possible ways that

the global relation may be partitioned in a distributed database system. This approach needs to incorporate performance evaluation methods for merging grid cells, and to articulate the architecture and functions that a database server should have.

Xuemin, Maria, and Yanchun [5] have investigated the allocation of database fragments to a network so that the overall communication cost for processing a given set of transactions is minimized.

Chun-Hung Cheng, Wing-Kin Lee, Kam-Fai Wong [6] have explored the use of a genetic search-based clustering algorithm for data partitioning to achieve high database retrieval performance. They formulate the clustering problem in data partitioning as a Travelling Salesman Problem (TSP) and propose 2 genetic operators SE and SP, as well as modified version of ER operators to solve the associated (TSP). A vertical partitioning technique is used in their algorithm, and they show that their model is applied to solve the horizontal partitioning problem.

Motzkin D. [7] has developed a distributed database design tool that provides for increased fault tolerance and data availability. She has focused on how to assign replications of fragments sites in a way that increases the fault tolerance of the distributed database. The design tool has two components: Initial Optimal Fragment Distribution (IOFD), and Fault Tolerance Enhancement (FTE) where each component is composed of three units: input parameters, design algorithm, and output. The algorithms used in this model can utilize parallel processing, where fragmentation and fragment assignment can be executed in parallel**.**

Tamhankar and Ram [8] have developed a comprehensive methodology for fragmentation and distribution of data across multiple sites such that design objectives in terms of response time and availability for transactions, and constraints on storage space are adequately addressed. Daudpota and Nadeem [9] have constructed a formal model of data allocation and have derived an algorithm to fragment and allocate the relations. Their work is not applied to the distributed applications, which have different network connectivity (LAN/WAN).

Stonebraker [10] described a new architecture for distributed data (Mariposa), which involves the design of an experimental distributed data management system providing high performance in an environment of high data mobility and heterogeneous host capabilities. This approach doesn't guarantee the ability of all sites to process a given portion of a given query.

Peddemors and Hertzberger [11] have described the first phase realization of a distributed database system in which an iterative process is used to build the distributed database system. Each phase has a set of objectives, spans a limited amount of time, adds functionality, and the output of every phase serves as input for the next phase. However, in their work the generic server interface is not easily usable; for every application, a new server interface has to be written.

Papastavrou, Samaras, and Pitoura [12] have developed a Java-based distributed client/server applications over the web to use mobile agents between the client program and the server machine, to provide database connectivity, processing and communication, and to eliminate the overheads of the existing methodologies. In case of executing multiple transactions from the web client on the same database module, it needs to connect, authenticate and disconnect them separately, and the cost of transactions in this way becomes costly.

Lee, Shi, Y., and Stolen, J. [13] Mahmood, Khan, H.U, and Fatmi, H.A [14], and March and Rho [15] have studied allocating data over geographically dispersed sites connected by data communication networks. They have not covered post-allocation of data, and they consider files reallocation only, and the data and operation allocation problems are independent and can be solved simultaneously.

H.Lee,Y.-K.Park, G.Jang, S.-Y.Huh [16] have proposed a heuristic methodology for determining file and workload allocation simultaneously on a LAN. This method minimizes the response time for processing transactions. Only transactions with same properties are routed to the same server, which does not guarantee the minimization of the communication cost. Their assumption of Non-redundant allocation decreases the reliability of the system, and the impact of storing fragment copies on the sites of the LAN is not very significant.

Yin-Fu Huang, Jyh-Her Chen [17] have proposed a heuristic algorithm that reflects transaction behavior in distributed database. Their model determines the replicated number of each fragment and finds a near-optimal allocation of all fragments in a WAN such that the total communication cost is minimized. The fragments accessed by a transaction are all assumed independent, which is not the case in the real world. This method neglects site information like storage and processing capacity. Their model was applied on a LAN network instead of WAN. They did not minimize the transaction response time, and they consider the CPU processing time and I/O access time as minor factors in minimizing the total cost in the environment of WAN.

3. THE STRATEGY DESIGN

3.1 PARTITION THE DATABASE

Partitioning the data across distributed systems is essential and it can be done in different ways. The research described in this paper discusses partitioning a database into pair-wise disjoint fragments by using a horizontal partitioning technique, in which the records of a relation are assigned to different disjoint fragments such that the relation can be obtained by union of the disjoint fragments. In this type of fragmentation, all of the information in the record is used, or else none is. This method guarantees the ability of all sites to process a given portion of a given transaction, but the other partitioning methods need to incorporate performance evaluation methods for merging grid cells, and to articulate the architecture and functions that a database server should have.

Since a horizontal fragment technique is used; the number of database segments is equal to the number of applications. The global database is segmented through the application sites using the relational operators SELECT, JOIN, and SEMIJOIN [18]. These segments are then split into fragments which are pair-wise disjoint (to avoid allocating unnecessary records to the fragments for a given application), and based on a horizontal partitioning technique each fragment is either completely required by a transaction, or it is not used at all. There is no partial use.

Algorithm:

k = Number of the last fragment in the database (0 at the beginning)

Repeat for all relations in the database

Repeat for all pairs of segments S_i, S_j in each relation where $i \neq j$

If $S_i \cap S_j$ is not empty Then

$k = k + 1$

$F_k = S_i \cap S_j$

$F_{k+1} = S_i - F_k$

$F_{k+2} = S_j - F_k$

S_i and S_j are omitted

End if

Until all pairs of segments in each relation have been processed

Until all relations in the database have been processed

Rename the remain fragment numbers sequentially

In the case of free records, which do not belong to any fragment in any relation in the database, a new fragment should be created and added to the collection of fragments in that relation.

Algorithm:

k = Number of the last fragment number

Repeat for all relations in the database

$k = k + 1$

$F_k = R - \cup F_i$ (for all fragments F_i in relation R)

IF F_k is not empty then

Add F_k to the collected fragments of the relation

End if

Until all relations in the database have been processed

3.2 GROUPING SITES INTO CLUSTERS

Clustering is a method of storing tables that can increase I/O performance and reduce storage overhead. Sites are grouped in clusters based on the communication cost unit (cost of sending K bytes between two sites).

Grouping the sites into clusters helps to eliminate the extra communication costs between the sites during the process of data allocation. We developed an algorithm for grouping sites into clusters, and determine whether or not a set of sites assigned to a cluster. If the communication cost between two or more sites is less than, or equal to, a certain number X units (the threshold between clusters depending on the site's network system) then it will be grouped together in one cluster. Performing this procedure after partitioning the data will minimize the communication costs between clusters and sites, and the clustering algorithm in this strategy is considered the fastest way to determine the data allocation to a set of sites rather than site by site.

Algorithm:

Repeat

For I = 1 to the number of sites in the database

For J = 1 to the number of sites in the database

If $I \neq J$ and communication cost between site I and site J <= X then

Site I and site J are grouped together (at the same cluster)

End if

End for

End for

Until all sites in the database have been processed

3.3 ALLOCATING FRAGMENTS TO CLUSTERS

To determine fragment allocation at clusters and their respective sites, an algorithm based on a calculated value (benefit value) has been developed. The algorithm will determine whether the fragment is allocated to or omitted from the cluster. This method attempts to minimize the communication costs by distributing the global database over the sites, increasing availability and reliability where multiple copies of the same data are allocated.

Initially, fragments are allocated to all clusters having applications, which use the fragment, and the benefit (B) of allocating a fragment to a cluster is computed. For each cluster, the benefit of allocating the fragment to the cluster is computed as the cost of not allocating the fragment to the cluster minus the cost of allocating the fragment to the cluster.

The cost of allocating the fragment is computed as the sum of: cost of local retrievals, cost of local updates, cost of space occupied by the fragment, and cost of updates sent from other clusters (remote update). The cost of not allocating a fragment is computed as the sum of local retrievals plus the cost of local retrievals from other clusters (remote clusters). This method of data allocation minimizes the transactions total response time. If the benefit of allocating the fragment to the cluster is positive (greater than or equal zero) the fragment is allocated to the cluster, otherwise the fragment will be cancelled from the cluster.

Fragment F_i is initially allocated to the site if it satisfies the following condition: The cost of allocating the fragment to the cluster is less than the cost of not allocating the fragment to the same cluster (the fragment handled remotely). The cost taken into consideration for each cluster is the average communication for all clusters. The following variables are used in the allocating algorithm.

$FREQR(T_j,F_i,C_k)$: Average number of frequency of retrieval issued by transactions T_j's to fragment F_i at cluster C_k.

$FREQR(T_j,F_i,C_k,S_x)$: Average number of frequency of retrieval issued by transactions T_j's to fragment F_i at site S_x in cluster C_k.

$FREQU(T_j,F_i,C_k)$: Average number of frequency of update issued by transactions T_j's to fragment F_i at cluster C_k.

$FREQU(T_j,F_i,C_k,S_x)$: Average number of frequency of update issued by transactions T_j's to fragment F_i at site S_x in cluster C_k.

$RCsum(C_i)$: Sum of remote communications at cluster C_i.

$RUsum(C_i)$: Sum of remote updates at cluster C_i.

$RCsum(C_i,S_x)$: Sum of remote communications at site S_x in cluster C_i.

$RUsum(C_i,S_x)$: Sum of remote updates at site S_x in cluster C_i.

$CR(C_i)$: Average cost of retrieval at cluster C_i.

$CR(C_j,S_x)$: Cost of retrieval at site S_x in cluster C_j.

$CU(C_i)$: Average cost of update at cluster C_i.

$CU(C_j,S_x)$: Cost of update at site S_x in cluster C_j.

CC(C): Average cost of communication between clusters.

CC(S): Average cost of communication between sites.

ACC(C): Average cost of communications between clusters other than the current one.

ACC(S): Average cost of communications between sites other than the current one.

UR: Unit retrieval.

UU: Unit update.

UC: Unit communication (Bytes).

Rratio: Retrieval ratio.

Uratio: Update ratio.

$Csp(C_i)$: Average cost of space of cluster C_i.

$Csp(C_j,Sx)$: Average cost of space of site S_x in cluster C_j.

$Fsize(F_i)$: Size of fragment F_i (Bytes).

CNUsum: Sum of costs of not allocating fragment for update.

CNCsum: Sum of costs of not allocating fragment for communication.

$CA(F_i,C_j)$: Cost of allocating fragment F_i to cluster C_j.

$CA(F_i,C_j,S_x)$: Cost of allocating fragment F_i to site S_x in cluster C_j.

$CN(F_i,C_j)$: Cost of not allocating fragment F_i to cluster C_j.

$CN(F_i,C_j,S_x)$: Cost of not allocating fragment F_i to site S_x in cluster C_j.

$B(F_i,C_j)$: Benefit of allocating fragment F_i to cluster C_j.

$B(F_i,C_j,S_x)$:Benefit of allocating fragment F_i to site S_x in cluster C_j.

Algorithm:

For I = 1 to number of fragments in the database do
 CNUSUM = 0
 CNCSUM = 0
 For J = 1 to number of clusters at fragment I do
  For k = 1 to number of clusters at fragment I do
   If J ≠ k Then
    CNUSUM = CNUSUM + CU(C_j) *
             FREQU(T_k,F_i,C_k)
    CNCSUM = CNCSUM + FREQU(T_k,F_i,C_k) *
             Uratio * ACC(C)
   End if
  End for
 RUsum(C_j) = CNUSUM ;
RCsum(C_j) = CNCSUM ;
  CA(F_i,C_j) = CR(C_j) * FREQR(T_j,F_i,C_j) +
      CU(C_j) * FREQU(T_j,F_i,C_j) +
      Csp(C_j) * Fsize(F_i) + RUsum(C_j) + RCsum(C_j)
  CN(F_i,C_j) = CR(C_j) * FREQR(T_j,F_i,C_j) +
       FREQR(T_j,F_i,C_j) * Rratio * CC(C)
  B(F_i,C_j) = CN(F_i,C_j) - CA(F_i,C_j)
 End for
 CNUSUM = 0
 CNCSUM = 0
 End for

The benefit is computed for all fragments at each cluster, according to the algorithm described above. The fragments that give positive benefit results are allocated to the clusters.

3.4 ALLOCATING FRAGMENTS TO THE SITES

The fragments will be allocated to the sites of each cluster if they show positive (benefit values).

Data allocations could be increased or decreased to meet the requirements of the strategy for the purpose of availability, reliability, and integrity.

The benefit of allocating fragments to sites in clusters that are allocated by fragments are computed and described in the following algorithm.

Algorithm:

For I = 1 to number of fragments in the database do
 For J = 1 to number of clusters in fragment I do
  CNUSUM = 0
  CNCSUM = 0
 For k = 1 to number of sites at cluster J do
  For x = 1 to number of sites at cluster J do
   If k ≠ x then
    CNUSUM = CNUSUM + CU(C_k,S_x) *
             FREQU(T_x,F_i,C_k,S_x)
    CNCSUM = CNCSUM + FREQU(T_x,F_i,C_k,S_x) *
             Uratio * ACC(S)
   End if
  End for
  RUsum(S_k) = CNUSUM
  RCsum(S_k) = CNCSUM
    CA(F_i,S_k) = CR(S_k) * FREQR(T_k,F_i,C_j,S_k) +
        CU(S_k) * FREQU(T_k,F_i,C_j,S_k) +
        Csp(S_k) * Fsize(F_i) + RUsum(S_k) +
        RCsum(S_k)
    CN(F_i,S_k) = CR(S_k) * FREQR(T_k,F_i,C_j,S_k) +
         FREQR(T_k,F_i,C_j,S_k) * Rratio * CC(S)
    B(F_i,S_k) = CN(F_i,S_k) - CA(F_i,S_k)
  End for
  CNUSUM = 0
  CNCSUM = 0
 End for

The benefit is computed for all sites at each cluster that is allocated to the fragment.

3.5 FINAL ALLOCATING OF FRAGMENTS TO THE SITES

Fragments are allocated to the sites which give positive benefit results.

Algorithm:

For I = 1 to number of fragments in the database do
For J = 1 to number of clusters in fragment I do
For k = 1 to number of sites at cluster J do
IF $B(F_i,C_j,S_k) > 0$ then
Allocate Fragment F_i to Cluster C_j in Site S_k
Else
Cancel Allocation of Fragment F_i from Site S_k at Cluster C_j
End if
End for
End for
End for

3.6 COMPLEXITY OF COMPUTATION

The time complexity of this research is described as follows:
The complexity of the Define-Segment algorithm is O(A*N) where A is the number of applications, and N is the average number of records in each application. The complexity of the Define-Fragment algorithm is $O(R*N^2)$ where R is the number of relations, and N is the average number of records in each relation. The complexity of computing average retrieval and update frequencies is O(F*S*A*N) where F is the number of the fragments in the database, S is the number of sites, A is the number of applications at each site, and N is the average number of records in each application.

Since the sites sorted on the basis of their clusters in ascending order for each fragment, the strategy design model has near optimal allocation complexity bounded by $O(R*N^2 + F*S*A*N + A*N)$.

3.7 PERFORMANCE EVALUATION

The average communication cost between clusters and sites, as well as the average number of retrievals and updates are used in the proposed algorithms, because the time complexity needs for average computations is less than the time complexity when other techniques are used which depend on sorting the sites according to some computation fields.

System performance is enhanced by removing the redundant records from the database segments and by increasing availability and reliability where multiple copies of the same data are allocated. That will reduce the communication costs where the fragments are needed frequently. Figure 1 shows the distribution of segments and fragments over the sites (before and after applying our algorithms) on a sample of 45 different applications distributed over 12 sites connected through different networks.

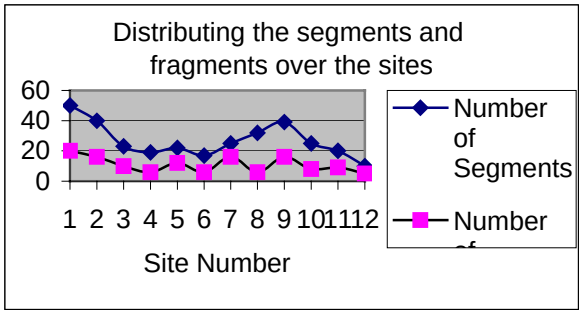

FIGURE 1. THE DISTRIBUTION OF SEGMENTS AND FRAGMENTS OVER THE SITES

4. CONCLUSION

The strategy is designed to meet the requirements of determining data fragmentation and allocation in distributed database environment, minimizing the communication cost between sites, and enhancing the performance in a heterogeneous network environment system. We described a horizontal partitioning technique that partition the database into pair-wise disjoint fragments and removing the redundant records from the database segments which enhance the system performance. A Clustering algorithm is developed to group the sites into clusters which enables the system to determine whether or not a set of sites are assigned to a cluster based on their communication costs. This will minimize the communication costs between the sites. We developed data allocation algorithms to enhance system performance by increasing availability and reliability where multiple copies of the same data are allocated. The strategy presents a near optimal allocation complexity and it can be implemented in different network environments even if the input parameters (relations, sites, data fields, records, and applications) are very large.

In the future we will focus on finding a new computation method to determine the least communication cost between sites and adding an adaptive algorithm to incorporate space and reliability constraints during the determination of fragment allocation.

REFERENCES

[1] Tanenbaum, Andrew, *Distributed Database Systems*. Prentice Hall, 1995.

[2] Sape Mullender, *Distributed Systems*, Addison-Wesley, 1993.

[3] Minyoung, Ra & Park, Yang-sun, Data fragmentation and allocation for PC-based distributed database design, *Korea: Science & Engineering Foundation* 1992.

[4] Navathe, Karlapalem, and Minyoung, A mixed fragmentation methodology for initial distributed database design. *Journal-of-Computer-and-Software-Engineering.*1995 vol.3, no.4; p.395-425.

[5] Xuemin, Maria, and Yanchun, On data allocation with minimum overall communication costs in distributed database design. *Proceedings ICCI '93. Fifth International Conference on Computing and Information (Cat. No.93TH0563-7). IEEE Comput. Soc. Press, Los Alamitos, CA, USA;* 1993; xvi+587 pp. p.539-44.

[6] Chun-Hung Cheng, Wing-Kin Lee, Kam-Fai Wong, A GeneticAlgorithm-Based Clustering Approach for Database Partitioning. *IEEE Transactions On Systems, Man, And Cybernetics-Part C: Applications and Reviews*, 2002, August Vol. 32 No. 3.

[7] Motzkin D, A distributed database design tool that provides for increased fault tolerance and data availability. *International Symposium on Engineered Software Systems 1993. Proceedings the ISESS Symposium. World Scientific, Singapore;* 1993; xvii+264. PP.222-36.

[8] Tamhankar, AM & Ram S, Database Fragmentation and Allocation: An Integrated Methodology and Case Study. *IEEE Transactions on Systems, Man. and Cybernetics-Part A. Systems and Humans*. 1998 Vol. 28. No 3. May PP. 288 – 305.

[9] Daudpota,NH, Five steps to construct a model of data allocation for distributed database systems. *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies.* 1998 vol.11, no.2; Sept.-Oct. p.153-68.

[10] Stonebraker, Michael, Aoki, Paul, Devine, Robert, Litwin, Withold and Olson, Michael, Mariposa: A new architecture for distributed data, *IEEE Database Engineering*. 1994 P. 54-65.

[11] Peddemors,AJH & Hertzberger LO, A high performance distributed database system for enhanced Internet services. Future-Generation-Computer-Systems. 1999 vol.15, no.3; April p.407-15.

[12] Papastavrou, Samaras, and Pitoura, Mobile agents for WWW distributed database access. *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337). IEEE Comput. Soc, Los Alamitos, CA, USA;* 1999; xxiii+648 pp. p.228-37.

[13] Lee, Shi, Y., and Stolen, J, Allocating Data Files Over a Wide Area network: Goal Setting and Compromise Design. *Information and Management*,1994 vol. 26, no. 2. p. 85-93.

[14] Mahmood Khan H.U, and Fatmi H.A, Adaptive File Allocation in Distributed Computer Systems, *Distributed Systems Engineering*,1994 vol. 1, no. 6. p. 354-361.

[15] March,ST & Rho,S. Allocating data and operations to nodes in distributed database design. *IEEE-Transactions-on-Knowledge-and-Data-Engineering.* 1995 vol.7, no.2; April p.305-17.

[16] H.Lee,Y.-K.Park, G.Jang, S.-Y.Huh, Designing a distributed database on a local area network: A methodology and decision support system. *Information and Software Technology.* 2000, 42 P. 171-184.

[17] Yin-Fu Huang, Jyh-Her Chen, Fragment Allocation in Distributed Database Design. *Journal of Information Science and Engineering.* 2001, 17 P. 491-506.

[18] M. Tamer Ozsu & Patrick Valduriez, *Principles of Distributed Database Systems*. 2nd ed. Prentice Hall, 1999.