

Fragment Allocation in Distributed Database Design

YIN-FU HUANG AND JYH-HER CHEN

*Institute of Electronics and Information Engineering
National Yunlin University of Science and Technology
Yunlin, Taiwan 640, R.O.C.
E-mail: huangyf@el.yuntech.edu.tw*

On a Wide Area Network (WAN), fragment allocation is a major issue in distributed database design since it concerns the overall performance of distributed database systems. Here we propose a simple and comprehensive model that reflects transaction behavior in distributed databases. Based on the model and transaction information, two heuristic algorithms are developed to find a near-optimal allocation such that the total communication cost is minimized as much as possible. The results show that the fragment allocation found by the algorithms is close to being an optimal one. Some experiments were also conducted to verify that the cost formulas can truly reflect the communication cost in the real world.

Keywords: distributed databases, fragment allocation, allocation model, communication cost, heuristic algorithms

1. INTRODUCTION

Distributed database design involves the following interrelated issues: (1) how a global relation should be fragmented, (2) how many copies of a fragment should be replicated, (3) how fragments should be allocated to the sites of the communication network, and (4) what the necessary information for fragmentation and allocation is. These issues complicate distributed database design. Even if each issue is considered individually, it is still an intractable problem. To simplify the overall problem, we address the fragment allocation issue only, assuming that all global relations have already been fragmented. Thus, the problem investigated here is determining the replicated number of each fragment and then finding a near-optimal allocation of all fragments, including the replicated ones, in a Wide Area Network (WAN) such that the total communication cost is minimized.

For a read request issued by a transaction, it may be simple just to load the target fragment at the issuing site, or it may be a little complicated to load the target fragment from a remote site. A write request could be most complicated since a write propagation should be executed to maintain consistency among all the fragment copies if multiple fragment copies are spread throughout the network. The frequency of each request issued at the sites must also be considered in the allocation model. Since the behaviors of different transactions maybe result in different optimal fragment allocations, cost formulas should be derived to minimize the transaction cost according to the transaction information.

Received December 14, 1999; revised March 3 & June 20, 2000; accepted July 31, 2000.
Communicated by Arbee L. P. Chen.

Many reports on database allocation have been published. In 1982, Chang developed a theory of fragment allocation and designed a network flow algorithm to solve the database allocation problem [1]. Because the database allocation problem is NP-complete, some heuristic algorithms, such as the knapsack problem solutions [2] and branch-and-bound techniques [3], were adopted to solve the problem. To reduce the complexity of the 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 a greedy algorithm that tries to improve the initial feasible solution [4, 5]. Furthermore, there have been some works on the database performance, such as analyses of file redundancy [6] and database allocation [7]. Recently, the database allocation problem was formulated as a zero/one integer programming [8], and even incorporated with a specific concurrency control mechanism [9, 10]. For an Ethernet-based local area network, some analytic approaches have been used to find the response time of transactions to deal with the database allocation problem [11]. Though the model proposed by Raghuram et al. [12] was very comprehensive, involving a few constraints, such as the computational power of each site and the maximum desired time of response to a request, the replicated copies problem was not considered. One work described a machine learning based time invariant fragmentation method that acquires knowledge about the data usage patterns for each node and used simulation to show its effectiveness [13]. Lin et al. presented data allocation algorithms to achieve the minimum overall communication cost [14]. Besides allocating data, a mathematical modeling approach and a genetic algorithm developed by March and Rho allocate operations to nodes [15]. For a system requiring high availability, Park and Baik proposed a transaction commit probability model and a genetic algorithm to minimize the processing cost [16].

Although a large amount of researchers have proposed models and algorithms designed to allocate fragments in a distributed database, most of their models are very complicated and not well understood. Thus, it is difficult to use them in a real environment. Here, we propose a simple and comprehensive model that reflects transaction behavior in distributed databases. To show our performance, we compare our results with those of Lin et al. [14] since most of their assumptions and the measure parameter (i.e., communication cost) in the model are the same as ours. Lin et al. proposed an approximation algorithm called SIMPLE for a simple data allocation problem. For each fragment f_i , the algorithm SIMPLE starts to allocate copies of f_i to those nodes j with $B_{ij} \geq 0$. The symbol B_{ij} denotes the total data volume of f_i required to send to node j to process the transactions issued at node j , minus the total data volume of f_i required by the transactions issued at all the nodes to update fragment f_i . Then, it finds other nodes to which copies of f_i can be allocated in order to greedily reduce the overall communication cost. The results show that the fragment allocation found by our algorithms is close to being an optimal one and is better than that found by Lin et al. Some experiments were also conducted to verify that the cost formulas can truly reflect the communication cost in the real world.

This paper is organized as follows. In Section 2, a transaction's behaviors are analyzed, and an allocation model is proposed. Then, two heuristic algorithms are developed to find a near-optimal allocation in Section 3. To verify that the cost formulas are able to truly reflect the communication cost in the real world, some experiments were conducted and are presented in Section 4. Finally, a conclusion is given in Section 5.

2. THE ALLOCATION MODEL

2.1 Fragment Allocation Problem

Before beginning our exploration of fragment allocation, the allocation problem must be clearly defined. Here, we will only address a WAN environment since the impact of storing fragment copies on the sites of a LAN is not very significant. Assume that we have a WAN consisting of sites $S = \{S_1, S_2, \dots, S_m\}$, on which a set of transactions $T = \{T_1, T_2, \dots, T_q\}$ is running, and a set of fragments $F = \{F_1, F_2, \dots, F_n\}$, into which all global relations have been partitioned during the fragmentation phase of distributed database design. To make the allocation problem more general, we consider that it involves not only determining the number of copies of each fragment, but also finding the optimal allocation of each fragment copy in F to S , according to the information given by the network and T . As for the definition of optimality, there are two different measures in general [17]:

1. Minimal cost: The cost function consists of the cost of storing each F_j on site S_k , the cost of querying F_j at site S_k , the cost of updating F_j at all sites where it is stored, and the cost of data communication.
2. Performance: Two well-known strategies are to minimize the response time and to maximize the system throughput at each site.

In this paper, the optimality measure adopted in our allocation model is the minimal cost. Furthermore, for a WAN with a limited bandwidth of 50 Kbps, I/O access time and CPU processing time are not the major factors that should be considered in minimizing the total cost. Thus, the allocation problem is simplified to that of allocating fragment copies to sites such that the total communication cost is minimal.

2.2 Information Requirements

Before we derive the cost formulas, some information must be analyzed in advance; that is (1) the quantitative data about a database, (2) the transaction behavior, (3) the site information, and (4) the network information [18].

Database information. The size of each fragment, called $\text{size}(F_j)$, must be defined since it plays a major role when computing the communication cost.

Transaction information. In the model, we have two access matrices, RM and UM , that describe the retrieval and update behaviors of all the transactions. The elements r_{ij} (or u_{ij}) in RM (or UM) specify the access frequency of fragment F_j in transaction T_i . Since not all the fragments are accessed by a transaction, the entries in the matrix may be 0. An example is shown as follows:

RM:

	F ₁	F ₂	F ₃	F ₄	F ₅
T ₁	2	3	0	0	0
T ₂	2	0	0	1	0
T ₃	0	0	3	0	0
T ₄	3	0	2	0	0

UM:

	F ₁	F ₂	F ₃	F ₄	F ₅
T ₁	0	0	0	1	2
T ₂	0	3	0	0	0
T ₃	2	1	0	1	0
T ₄	0	0	0	0	3

In the access matrices RM and UM, transaction T₃ retrieves fragment F₃ three times and updates fragment F₁ twice, F₂ once, and F₄ once for each run.

When a transaction accesses a fragment, not all the tuples of the fragment must be retrieved or updated. The number of tuples to be retrieved or updated is not the same for all transactions. Therefore, we define a selectivity matrix SEL that indicates the percentage of a fragment F_j to be accessed in transaction T_i:

SEL:(%)

	F ₁	F ₂	F ₃	F ₄	F ₅
T ₁	0.1	0.1	0	0.3	0.2
T ₂	0.1	0.3	0	1	0
T ₃	2	5	0.1	0.5	0
T ₄	0.5	0	10	0	4

In the selectivity matrix SEL, transaction T₃ retrieves only 0.1 percent of fragment F₃ and updates 2 percent of fragment F₁, 5 percent of F₂, and 0.5 percent of F₄.

Furthermore, we also need to define a frequency matrix FREQ that indicates the execution frequency of all the transactions issued at each site:

FREQ:

	S ₁	S ₂	S ₃	S ₄
T ₁	0	2	3	1
T ₂	0	3	0	0
T ₃	2	0	1	0
T ₄	0	0	4	0

Transaction T₃ shown in matrix FREQ indicates that it is run twice at site S₁ and once at site S₃.

Site information. The site information in a network could be the storage and processing capacity; in general, the information is viewed as the constraints in an allocation model. To simplify the allocation model, we will not consider them here.

Network information. In a WAN environment, the communication cost depicted by two components is the dominant factor in the total cost. C_{ini} is the constant cost of initiating a data packet with size p_size while $CTR_{i,j}$ is the cost of transmitting a unit of data from site S_i to site S_j . Thus, the communication cost of transmitting m_size of data can be assumed to be a linear function as follows:

$$CC(CTR_{i,j}, m_size) = C_{ini} * \frac{m_size}{p_size} + CTR_{i,j} * m_size \quad (1)$$

We assume that each site in the network is connected to another site by a logical communication link. Therefore, the cost $CTR_{i,j}$ can be specified with a communication cost matrix CTR . To simplify our problem, we assume that CTR is a symmetric matrix:

CTR	S_1	S_2	S_3	S_4
S_1	0	0.32	0.48	0.16
S_2	0.32	0	0.64	0.32
S_3	0.48	0.64	0	0.64
S_4	0.16	0.32	0.64	0

The example shown above is a communication cost matrix based on a WAN with a limited bandwidth of 50 Kbps (i.e., 0.16 ms/byte), where each element is a multiple of the value 0.16.

In a WAN, before a transaction accesses a remote fragment, it must create a virtual circuit to connect with another end. During the lifetime of the transaction, it uses the virtual circuit to send requests and get replies. At the end of the transaction, the virtual circuit is closed. To reflect the real communication of a WAN, the cost of building virtual circuits, denoted as VC_{ini} , is also considered in our model.

2.3 Cost Formulas

Here, we derive communication cost formulas to reflect a transaction's behaviors on a distributed database in a WAN environment. According to the information mentioned in Section 2.2, we try to find a near-optimal fragment allocation such that the total communication cost is minimized. Minimizing the communication cost formula can be expressed as follows:

$$\min(CC_{load} + CC_{proc}) \quad (2)$$

The communication cost formula consists of two components: the communication cost CC_{load} for data loading and the communication cost CC_{proc} for transaction processing.

CC_{load} denotes the cost of loading all the fragment copies onto the network sites before the transactions are processed. It can be expressed further as follows:

$$CC_{load} = \sum_{j=1}^n \sum_{k=1}^m FAT_{j,k} * CC(CTR_{SI,k}, size(F_j)), \quad (3)$$

where FAT is the matrix of the fragment allocation table. $FAT_{j,k}$ is 1 if fragment F_j is allocated to site S_k ; otherwise, it is 0. Furthermore, SI is a master site responsible for loading all the fragment copies onto the network sites.

Then, the cost CC_{proc} consists of three components: transaction retrieval TR_i , transaction update TU_i , and building virtual circuits VC_{ini} . It can be expressed as follows:

$$CC_{proc} = \sum_{k=1}^m \sum_{i=1}^q FREQ_{i,k} * (TR_i + TU_i + VC_{ini}), \quad (4)$$

where $FREQ_{i,k}$ is the execution frequency of transaction T_i at site S_k . Also, the costs for transaction retrieval TR_i and transaction update TU_i can be expressed more explicitly as follows:

$$TR_i = \sum_{j=1}^n r_{ij} * \min(CC(CTR_{k,s} \text{ where } FAT_{j,s} = 1, \frac{SEL_{i,j}}{100} * size(F_j))), \quad (5)$$

$$TU_i = \sum_{j=1}^n u_{ij} * (\sum_{l=1}^m FAT_{j,l} * CC(CTR_{k,l}, \frac{SEL_{i,j}}{100} * size(F_j))). \quad (6)$$

The retrieval cost TR_i indicates that among all the sites with fragment copies F_j , only the site that yields the minimum transmission cost should be selected for transaction processing. However, the update cost TU_i indicates that all the transmission costs for remote sites where fragment copies F_j are stored should be summed up. This propagation update is required to maintain consistency among all the fragment copies F_j .

3. HEURISTIC ALGORITHMS

In the preceding section, we developed a generic allocation model. Finding an optimal fragment allocation in this model is a NP-complete problem since given n fragments and m sites, there will be $(2^m - 1)^n$ different combinations. Thus, we only look for heuristic algorithms to solve the problem. In this paper, two heuristic algorithms are proposed to minimize the communication cost as much as possible.

3.1 Algorithm-1

The first heuristic algorithm has three steps. In the first step, we initialize the fragment allocation table by only considering the retrieval requests issued by transactions. A retrieval request can be processed without any communication cost by allocating its target fragments to its issuing site. From the retrieval matrix RM and the frequency matrix $FREQ$ described in Section 2.2, we can initialize the fragment allocation table.

For fragment F_3 in RM, only transactions T_3 and T_4 can retrieve it. Since transactions T_3 and T_4 are issued only at site S_1 and S_3 , fragment F_3 is allocated to site S_1 and S_3 .

In the second step, we consider update requests. We know that the allocation table initialized in step 1 is the best allocation for retrieval requests since it incurs no communication cost. No more fragment copies will be necessary since this would not benefit retrieval requests but would increase the communication cost for update requests. Therefore, the issue that we consider in this step is how to remove fragment copies from the initialized allocation such that the communication cost will be reduced for all requests. Removing a fragment copy from a site greatly influences the communication cost of some requests. A local retrieval request will incur a communication cost since the target fragment copy has been removed from the local site. A remote retrieval request may or may not incur an extra communication cost, depending on whether the remote request accesses the removed copy. A local update request has no effect since originally no communication cost is involved. As for a remote update request, since the fragment copy is removed, there is no communication cost.

In short, when removing a fragment copy from a site, a local retrieval request will incur an extra cost, and a remote retrieval one may do so as well. However, a remote update request will benefit because the removed copy will not be accessed. Whenever this benefit is larger than the cost, we should remove the copy from the site. For a fragment, if the benefit is larger than the cost at several sites, then we only remove the most beneficial copy. The removal procedure is repeated continually unless no benefit is larger than the cost or unless the current copy is the only one in the WAN. Step 2 is repeated continuously until all the fragments allocated in step 1 are scanned.

In the last step, if one fragment has not yet been allocated to a site and has been updated by some transactions, we find its candidate sites according to the update matrix UM and the frequency matrix *FREQ*. Among these candidate sites, we only select one site with the least communication cost as the allocated one. Fragment F_5 in RM is not read by any transaction; however, transactions T_1 and T_4 update it in UM. Since transactions T_1 and T_4 are issued only at sites S_2 , S_3 , and S_4 , fragment F_5 will be allocated to one of them according to their communication cost.

The time complexity of Algorithm-1 is $O(nm^2q)$, where n (i.e. $|F|$) is the number of distinct fragments, m (i.e. $|S|$) is the number of sites, and q (i.e. $|T|$) is the number of transactions.

The detailed algorithm is as follows:

Algorithm-1

Input:

RM(*trans,frag*); /* Retrieval matrix */
UM(*trans,frag*); /* Update matrix */
SEL(*trans,frag*); /* Selectivity matrix */
FREQ(*trans,site*); /* Frequency matrix */
CTR(*site,site*); /* Communication cost matrix */

Output:

FAT(*frag,site*); /* Fragment allocation table */

Function:**Benefit**(*frag*,*site*); /* saved cost of removing *frag* from *site* */**Begin**

$$Benefit = \sum_{S_k \in S} \sum_{T_i \in T} FREQ(T_i, S_k) \times UM(T_i, frag) \\ \times CC(CTR_{S_k, site}, \frac{SEL(T_i, frag)}{100} \times size(frag));$$

return *Benefit*;**End****Function:****Cost**(*frag*,*site*); /* increased cost of removing *frag* from *site* */**Begin**For S_k in **S** do**begin**Let *n* be the site that has the **least network delay** from S_k
when S_k retrieves *frag*;**if** (*n* = *site*)**begin**

$$T1 = \sum_{T_i \in T} FREQ(T_i, S_k) \times RM(T_i, frag) \\ \times CC(CTR_{S_k, n}, \frac{SEL(T_i, frag)}{100} \times size(frag));$$

Let *n1* be the site that has the **next least network delay** from S_k
when S_k retrieves *frag*;

$$T2 = \sum_{T_i \in T} FREQ(T_i, S_k) \times RM(T_i, frag) \\ \times CC(CTR_{S_k, n1}, \frac{SEL(T_i, frag)}{100} \times size(frag));$$

$$Cost = Cost + (T2 - T1);$$

end**end**return *Cost*;**End****Function:****MinDelay**(*frag*) ; /* the least-delay site where *frag* is allocated */

/* omitted here */

Function:**NumFragCopy**(*frag*); /* number of *frag* in the system */

/* omitted here */

Begin

Step 1 /* Initialize a best solution for retrieval */

For T_i in \mathbf{T} , F_j in \mathbf{F} , S_k in \mathbf{S} do

if ($\mathbf{RM}(T_i, F_j) * \mathbf{FREQ}(T_i, S_k) > 0$)

$\mathbf{FAT}(F_j, S_k) = 1$;

Step 2 /* Check whether a fragment should be removed from one site */

For F_j in \mathbf{F} do

 While ($\mathbf{NumFragCopy}(F_j) > 1$)

begin

 Let S_k be the site with $\mathbf{FAT}(F_j, S_k) = 1$

 and a maximum value of ($\mathbf{Benefit}(F_j, S_k) - \mathbf{Cost}(F_j, S_k)$);

 if ($(\mathbf{Benefit}(F_j, S_k) - \mathbf{Cost}(F_j, S_k)) > 0$)

$\mathbf{FAT}(F_j, S_k) = 0$

else

break;

end

Step 3 /* If a fragment has not yet been allocated in the system but it has been updated by some transactions, select a least-delay site where the fragment is to be placed. */

For F_j in \mathbf{F} do

 if ($\mathbf{NumFragCopy}(F_j) = 0$ and $\mathbf{UM}(T_i, F_j) > 0$)

begin

$S_k = \mathbf{MinDelay}(F_j)$

$\mathbf{FAT}(F_j, S_k) = 1$;

end

End

3.2 Algorithm-2

The second heuristic algorithm also has three steps. Its first step and last step are the same as those in the first algorithm. In the second step, we scan all the entries of the allocation table initialized in step 1 according to their weights in descending order instead of scanning them row by row as in Algorithm-1. The weight of one entry is defined as the amount of saved data that is accessed by remote update requests when a fragment copy is removed from a site. The weight can be calculated according to the matrices FREQ, UM, SEL, and the fragment size $\text{size}(F_j)$. Each time, we select the largest weight value among the un-scanned entries and check whether its benefit is larger than its cost. If it is, the fragment copy is then removed from the site unless this copy is the only one in the WAN. Step 2 is repeated continuously until all the entries are scanned. The time complexity of Algorithm-2 is the same as that of Algorithm-1, i.e. $O(nm^2q)$.

The detailed algorithm is as follows:

Algorithm-2:

Input:

$\mathbf{RM}(\text{trans}, \text{frag});$ /* Retrieval matrix */

UM(*trans,frag*); /* Update matrix */
SEL(*trans,frag*); /* Selectivity matrix */
FREQ(*trans,site*); /* Frequency matrix */
CTR(*site,site*); /* Communication cost matrix */

Output:

FAT(*frag,site*); /* Fragment allocation table */

Variable:

WT(*frag,site*); /* Weight table */

Function:

Benefit(*frag,site*); /* saved cost of removing *frag* from *site* */
 /* same as **Algorithm-1** */

Function:

Cost(*frag,site*); /* increased cost of removing *frag* from *site* */
 /* same as **Algorithm-1** */

Function:

MinDelay(*frag*) ; /* the least-delay site where *frag* is allocated */
 /* omitted here */

Function:

NumFragCopy(*frag*); /* number of *frag* in the system */
 /* omitted here */

Begin

Step 1 /* **Step 1** in **Algorithm-1** */

Initialize a best solution for retrieval;

Step 2 /* Calculate the weight value */

For F_j in **F** do

For S_k in **S** do

$$\begin{aligned}
 \mathbf{WT}(F_j, S_k) = & \sum_{S_i \in S \wedge S_i \neq S_k} \sum_{T_i \in T} \mathbf{FREQ}(T_i, S_i) \times \mathbf{UM}(T_i, F_j) \\
 & \times \frac{\mathbf{SEL}(T_i, F_j)}{100} \times \mathit{size}(F_j);
 \end{aligned}$$

Step 3 /* Check whether a fragment should be removed from one site */

While (there exist any un-scanned entries in **WT**) do

begin

Let $\mathbf{WT}(F_j, S_k)$ be the un-scanned entry with the largest weight;

if (($\mathbf{FAT}(F_j, S_k) = 1$) and ($\mathbf{Benefit}(F_j, S_k) > \mathbf{Cost}(F_j, S_k)$))

and ($\mathbf{NumFragCopy}(F_j) > 1$))

$\mathbf{FAT}(F_j, S_k) = 0$;

end

Step 4 /* Step 3 in Algorithm-1 */

If a fragment has not yet been allocated in the system but it has been updated by some transactions, select a least-delay site where the fragment is to be placed;

End

4. NUMERICAL AND EXPERIMENTAL RESULTS

First, we computed the communication costs of an optimal allocation and both heuristic allocations, respectively, using the cost formulas derived in Section 2.3. We also compared our results with those of Lin et al. [14] since most of their assumptions and the measure parameter (i.e., communication cost) in their model are the same as ours. Then, we verified that the cost formulas were able to truly reflect the communication cost in the real world by conducting some experiments.

4.1 Numerical Results

In Section 2, we specified three constant parameters in the allocation model, that is, (1) the constant cost of initiating a data packet C_{ini} , (2) the size of a data packet p_size , and (3) the cost of building virtual circuits VC_{ini} . To match the real experimental environment built in the following subsection, the appropriate values given to these parameters were as follows: $C_{ini} = 0.032$ ms/byte, $p_size = 6250$ bytes, and $VC_{ini} = 65.5676$ ms. Furthermore, our model was based on a WAN with a limited bandwidth of 50 Kbps (i.e., 0.16 ms/byte). Thus, each element of the communication cost matrix CTR was a multiple of the value 0.16.

Two hypothetical environments were assumed in the analysis. One consisted of three transactions, three fragments, and four sites, and the other consisted of three transactions, five fragments, and four sites. Though the environments were very simple, the number of possible combination allocations was 3375 in the former and 759375 in the latter. Ten cases in which data were generated randomly for each environment were considered. The communication costs of both environments are shown in Table 1 and Table 2, respectively. From the tables, we can see that the communication costs of both heuristic algorithms are close to that of the optimal allocation, especially for Algorithm-1. In some cases, the result obtained using the heuristic algorithms is the optimal allocation. Furthermore, both algorithms performed better than Lin's. The rank percentages of all the algorithms are also given in the tables. The ranks of the algorithms could be easily obtained by calculating the costs of all possible combinations and comparing them. Here, all the ranks of our algorithms listed in both tables are among the top 10%. It is interesting that Lin's algorithm was not as good as what they claimed. Even worse, it was very unstable since its results changed irregularly.

Comparing two heuristic algorithms, we find that Algorithm-1 performed better than Algorithm-2 in most cases. The major difference between Algorithm-1 and Algorithm-2 is the scanning method used to remove a fragment copy from a site. Algorithm-1 checks whether a fragment copy should be removed from a site in a fragment-by-fragment order. However, Algorithm-2 checks the removal based on how

many data of a fragment copy will be updated at a site; that is, the more data in a fragment copy is updated, the earlier the fragment copy is removed from a site. In general, the performance of Algorithm-1 was worse than that of Algorithm-2 only if the total amount of updated data in the fragments scanned later was **much more** than that in the fragments scanned earlier. However, this situation did not occur frequently; thus, Algorithm-1 was better than Algorithm-2 most of the time. To see how close the results obtained using both heuristic algorithms are to the optimal allocation, we also show in bar charts the communication cost in Fig. 1 and Fig. 2.

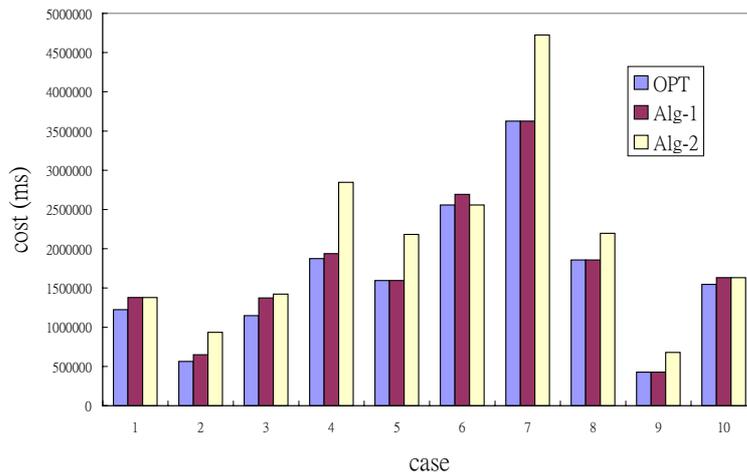


Fig. 1. Three transactions, three fragments, and four sites.

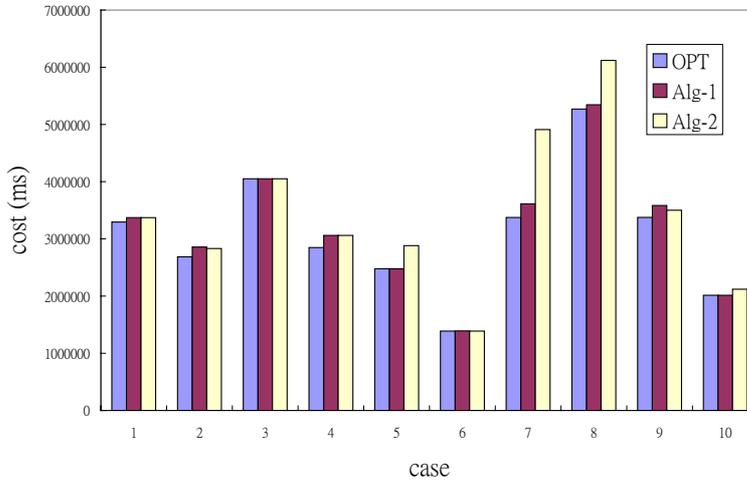


Fig. 2. Three transactions, five fragments, and four sites.

4.2 Experimental Results

We present here some experiments that were conducted to verify that the cost formulas are able to truly reflect the communication cost in the real world. Due to lack of a WAN environment and the unpredictable nature of the delay incurred by routers, we used an Ethernet-based local area network (LAN) to simulate transaction behavior on a WAN. The LAN consisted of 24 nodes that were SPARCstation 10s. To reduce the execution time in the experiments, we only studied one case for each environment mentioned in the preceding subsection. The first experiment investigated case 4 in Table 1, and the second experiment investigated case 8 in Table 2. Comparing the results obtained using the cost formulas and those obtained in the experiments, shown in Table 3 and Table 4, we find that the costs obtained using the cost formulas are very close to those obtained in the experiments though the latter are a little higher than the former. The difference is due to the overhead of the test programs and TCP-IP protocols. This result shows that the cost formulas used in our allocation model can truly reflect the communication cost in the real world.

Table 1. Three transactions, three fragments, and four sites.

Case	Optimal allocation	Algorithm-1		Algorithm-2		Lin's [14]	
	cost(ms)	cost	rank(%)	cost	rank(%)	Cost	rank(%)
1	1224075	1378210	1.10	1378210	1.10	1795353	10.00
2	564752	648852	0.31	934761	5.57	1312252	54.97
3	1147698	1372918	2.31	1421718	3.14	1585974	5.12
4	1875700	1937240	0.07	2846769	7.94	3046549	8.04
5	1594654	1594654	0.01	2183002	2.40	3733447	66.58
6	2557942	2692440	0.10	2557942	0.04	4234606	14.53
7	3625830	3625830	0.01	4722324	3.81	5199185	24.94
8	1857423	1857423	0.01	2196880	0.19	2645746	57.47
9	428467	428467	0.01	679825	2.61	741934	17.65
10	1545781	1630822	0.13	1630822	0.13	2442776	64.18

Table 2. Three transactions, five fragments, and four sites.

Case	Optimal allocation	Algorithm-1		Algorithm-2		Lin's [14]	
	cost(ms)	cost	rank(%)	cost	rank(%)	Cost	rank(%)
1	3293666	3371014	0.10	3371014	0.10	3517044	3.54
2	2686039	2857771	0.90	2830963	0.67	3800433	6.00
3	4051053	4051053	0.04	4051053	0.04	5449736	23.65
4	2847373	3060208	1.73	3060208	1.73	4050648	8.16
5	2475976	2475976	0.01	2882810	0.49	5024765	46.71
6	1389074	1389920	0.04	1389074	0.04	4832245	48.81
7	3373766	3611517	0.22	4910838	3.45	7036745	55.53
8	5269344	5344124	0.16	6121470	3.59	6837320	4.82
9	3376089	3583986	0.33	3501309	0.24	6649312	38.61
10	2014804	2014804	0.01	2121248	0.25	2660374	2.17

Table 3. Experiments and cost formulas for case 4 in Table 1.

	Optimal allocation	Algorithm-1	Algorithm-2
Experiments	1983627	2086520	2935374
Cost formulas	1875700	1937240	2846769

Table 4. Experiments and cost formulas for case 8 in Table 2.

	Optimal allocation	Algorithm-1	Algorithm-2
Experiments	5387463	5481648	6282376
Cost formulas	5269344	5344124	6121470

5. CONCLUSIONS

In this paper, for a fragment allocation problem, we have proposed a simple and comprehensive model that can reflect transaction behavior in distributed databases. Based on the model and transaction information, two heuristic algorithms have been developed to find a near-optimal allocation such that the total communication cost is minimized as much as possible. The results show that the fragment allocation found by our algorithms is close to being an optimal one and is better than that found by Lin et al. Some experiments were also conducted to verify that the cost formulas can truly reflect the communication cost of in the real world. However, some issues were not considered here. First, an “efficient” algorithm to find an optimal allocation was not explored here. Second, the fragments accessed by a transaction are not all independent in the real world, an issue we did not address. These issues are worth investigating in the future.

REFERENCES

1. S. K. Chang and A. C. Liu, “File allocation in a distributed database,” *International Journal of Computer Information Sciences*, Vol. 11, No. 5, 1982, pp. 325-340.
2. S. Ceri, G. Martella, and G. Pelagatti, “Optimal file allocation in a computer network: a solution method based on the knapsack problem,” *Computer Network*, Vol. 6, No. 5, 1982, pp. 345-357.
3. M. K. Fisher and D. S. Hochbaum, “Database allocation in computer networks,” *Journal ACM*, Vol. 27, No. 4, 1980, pp. 718-735.
4. S. Ceri, S. B. Navathe, and G. Wiederhold, “Distributed design of logical database schemes,” *IEEE Transactions on Software Engineering*, Vol. 9, No. 4, 1983, pp. 487-503.
5. S. Ceri and B. Pernici, “DATAID-D: methodology for distributed database design,” *Computer-Aided Database Design*, Amsterdam: North-Holland, 1985, pp. 157-183.

6. S. Muro, T. Ibaraki, H. Miyajima, and T. Hasegawa, "Evaluation of file redundancy in distributed database systems," *IEEE Transactions on Software Engineering*, Vol. 11, No. 2, 1985, pp. 199-205.
7. M. Yoshida, K. Mizumachi, A. Wakino, I. Oyake, and Y. Matsushita, "Time and cost evaluation schemes of multiple copies of data in distributed database systems," *IEEE Transactions on Software Engineering*, Vol. 11, No. 9, 1985, pp. 954-958.
8. G. M. Chiu and C. S. Raghavendra, "A model for optimal database allocation in distributed computing systems," in *Proceedings of IEEE INFOCOM '90*, 1990, pp. 827-833.
9. S. Ram and R. E. Marsten, "A model for database allocation incorporating a concurrency control mechanism," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No. 3, 1991, pp. 389-395.
10. A. M. Tamhankar and S. Ram, "Database fragmentation and allocation: an integrated methodology and case study," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 28, No. 3, 1998, pp. 288-305.
11. O. R. L. Sheng, "Database allocation in Ethernet-based local area networks: a queuing analytic approach," in *Proceedings of the Twenty-second Annual Hawaii International Conference on System Sciences*, 1989, pp. 733-742.
12. A. Raghuram, T. W. Morgan, and P. Julien-Laferrriere, "A model for determining the optimal topology and database allocation in computer network," in *Proceedings of the Eighth Annual International Phoenix Conference on Computers and Communications*, 1989, pp. 461-472.
13. A. R. Chaturvedi, A. K. Choubey, and J. Roan, "Scheduling the allocation of data fragments in a distributed database environment: a machine learning approach," *IEEE Transactions on Engineering Management*, Vol. 41, No. 2, 1994, pp. 194-207.
14. X. Lin, M. Orłowska, and Y. Zhang, "On data allocation with the minimum overall communication costs in distributed database design," in *Proceedings of the Fifth International Conference on Computing and Information*, 1993, pp. 539-544.
15. S. T. March and S. Rho, "Allocating data and operations to nodes in distributed database design," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 7, No. 2, 1995, pp. 305-317.
16. S. J. Park and D. K. Baik, "A data allocation considering data availability in distributed database systems," in *Proceedings of the International Conference on Parallel and Distributed Systems*, 1997, pp. 708-713.
17. L. W. Dowdy and D. V. Foster, "Comparative models of the file assignment problem," *ACM Computing Survey*, Vol. 14, No. 2, 1982, pp. 287-313.
18. M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, 2nd ed., Prentice-Hall International Editions, 1999.



Yin-Fu Huang (黃胤傳) received the B.S. degree in computer science from National Chiao-Tung University in 1979, and the M.S. and Ph.D. degrees in computer science from National Tsing-Hua University in 1984 and 1988, respectively. He is currently a professor in the Department of Electronic Engineering, National Yunlin University of Science and Technology. Between July 1988 and July 1992, he was with Chung Shan Institute of Science and Technology as an Assistant Researcher. His research interests include database systems, multimedia systems, performance evaluation, and operating systems.



Jyh-Her Chen (陳志和) received his B.S. and M.S. degrees from National Yunlin Institute of Technology in 1994 and 1996, respectively, all in electronic engineering. He is currently an engineer in the Comtrend Corporation. His major areas of interests are database systems, parallel processing, computer networks, and telecom.