Database Migration: A New Architecture for Transaction Processing in Broadband Networks

Takahiro Hara, *Member*, *IEEE*, Kaname Harumoto, *Member*, *IEEE*, Masahiko Tsukamoto, *Member*, *IEEE*, and Shojiro Nishio, *Member*, *IEEE*

Abstract—Due to recent developments in network technologies, broader channel bandwidth is becoming prevalent in worldwide networks. As one of the new technologies making good use of such broadband channels, dynamic relocation of databases through networks, which we call *database migration*, will soon be used in practice as a powerful and basic database operation. We propose two transaction processing methods to take advantage of database migration in broadband networks. These methods choose the most efficient transaction processing method between the conventional method, based on the two-phase commit protocol, and our method, using database migration. We also propose a concurrency control mechanism and a recovery mechanism for our proposed methods. Simulation results are presented comparing the performance of our proposed methods and the conventional transaction processing method based on the two-phase commit protocol. The results demonstrate that the effective use of database migration produces better performance than the conventional method.

Index Terms—Database migration, broadband network, distributed database system, transaction processing, concurrency control.

1 Introduction

1.1 Motivation

The recent evolution of broadband networks affects the design of database management systems [13], [15]. The most important issue regarding performance improvement of a network-wide database system is the effective use of broadband networks; however this is contrary to a typical conventional situation where minimizing the volume of data transmitted in (narrowband) networks is considered to be the primary factor for performance improvement.

The question is how we can use broadband networks efficiently. A possible answer is to make a database migrate from one site to another site through networks. We call the migration of a database *DB-migration*. DB-migration can be performed in a short period of time in broadband networks. For example, if the available bandwidth is 1 gigabit/second, a database which is 100 megabytes in size can be transferred in only 8.0×10^{-1} seconds. Therefore, the dynamic relocation of databases using DB-migration can be practically used for several purposes such as *transaction processing*.

In a conventional distributed database environment, each database is fixed at a particular site and a typical database operation is performed through several *operation request messages*. After the message exchange, the operation is consistently validated by using the *two-phase commit protocol* (2PC). Let us refer to such a fixed-database method as the *fixed-processing method*. Note that, in the fixed-processing method, transaction processing often requires many message transmissions, while if we use DB-migration, there is no need for exchanging messages after a transaction initiation site has gathered its necessary databases. Let us refer to such a method based on DB-migration as the *migration-processing method*. This means that the transaction processing time can be much shorter because the propagation delay is almost similar in a broadband network to the delay in conventional (narrowband) networks where the transmission delay is very small even when transmitting an entire database. For example, let us consider a transaction which accesses a database of 100 megabytes in New York from Tokyo, and these two points are connected by fiber optic cables with a bandwidth of 1 gigabit/second. Since the distance between these two points is about 1.2×10^7 meters and the speed of the light in a fiber cable is 2.1×10^8 meters/second, the propagation delay is at least 5.7×10^{-2} seconds even when we neglect the processing delay at both sites and the routing delay at each router or switch. On the other hand, the transfer of the whole database takes 8.0×10^{-1} seconds. Therefore, if the transaction requires more than seven rounds of communications, it is more efficient to process it locally in Tokyo by using DB-migration. If we do not neglect the processing delay and the routing delay, this threshold value becomes even smaller. With accelerating advances in broadband networks, DB-migration will soon become a powerful and basic database operation. Since the upper limit of the speed of light makes it impossible to drastically reduce the propagation delay, DB-migration will become more effective as the network scale grows.

• *The authors are with the Department of Information Systems Engineering, Graduate School of Engineering, Osaka University, 2-1 Yamadaoka, Suita, Osaka, 565-0871, Japan. E-mail: {hara, harmoto, tuka, nishio}@ise.eng.osaka-u.ac.jp.*

Manuscript received 20 Oct. 1997; revised 8 June 1998.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 105813.

It should be noted that we use relational data model terminology in this paper for simplicity of discussion. The data model is not restricted to the relational model.

1.2 Approach

In this paper, we consider the use of DB-migration for transaction processing. In environments where a database migrates, it is necessary for the system to manage the location of each database. The database location management assumed in this paper is described in Section 2.1. This issue is also discussed in Section 7.1.

One may think that it is more efficient to send a necessary fragment of a database instead of the whole database. However, this strategy imposes the following drawbacks:

- In general, the necessary parts of the database are determined while the transaction is being executed. For example, if the transaction includes the following instruction

  > [Set value of attribute *salary* to 5, 000 where the value of attribute *position* is "manager".],

  the necessary pages or tuples are not determined at the beginning of the transaction. Each necessary part must be sent to the transaction initiation site when it is required during transaction execution. Therefore, the number of data transmissions increases and thus, the performance deteriorates.
- In order to send the necessary parts of the databases, the database must be divided into considerably small partitions. This results in substantial overhead, i.e., the location management of each partition. Moreover, this requires many message transmissions to check the constraints among the attribute values.

These two drawbacks can be avoided by sending the entire database instead of sending only the necessary parts. In this paper, we use large lock granularity such as relations and thus we can determine the necessary data items at the beginning of the transaction. Even so, the transactions may have conditional branches across different relations. If there are few patterns of conditional branches, we can determine the necessary databases at the beginning by collecting these relations together into a database. Also, by gathering relations, which have a consistency constraint among them, together into a database, consistency checking does not require message transmission.

We do not consider DB-migration for personal databases of many users in the Internet or for a very large centrally managed database in the order of terabytes. We assume a closed system such as a distributed database system in an enterprise intranet with many branches located in different cities. To benefit from DB-migration, it is desirable that the databases in the system are not overly large (at most a total size of 1 gigabyte) and their sites are distributed in a wide area. A good candidate would be a product database on an intranet of a company with many branches around the world. If the company has 5,000 kinds of products with 100 kilobytes of data for each product, the total database size is 500 megabytes. Currently, it is easy to find such databases.

In this paper, we propose two transaction processing methods, which take advantage of DB-migration in broadband networks. These methods choose the more efficient method of either the fixed-processing method or the migration-processing method. The first method simply considers the processing of each transaction independently. Each time a transaction is processed, either the migration-processing method or the fixed-processing method is dynamically chosen in order to obtain better performance in a given system environment. In the second method, successive execution and the relationships among multiple transactions are taken into account considering the complex structure of transactions such as *nested transactions* [12]. The selection from two transaction processing methods is determined by the access patterns of the transactions (i.e., the statistics regarding the transactions which have accessed a certain database). Owing to these methods, the transaction processing throughput is expected to be significantly increased compared to a conventional distributed database system.

We also propose a concurrency control mechanism and a recovery mechanism for our proposed transaction processing methods. Since DB-migration is an operation which occupies a database for some time, a deadlock for DB-migration may frequently occur and the performance may deteriorate. The former mechanism will control the database operations, including DB-migration, which are issued concurrently. It is necessary to prevent the transaction processing throughput from deteriorating in environments where data contention is a significant factor. The latter mechanism will execute an efficient recovery process for migratory databases.

In order to evaluate the performance of the proposed transaction processing methods in comparison with the fixed-processing method, simulation studies were undertaken. The results show that effective use of DB-migration greatly contributes to improving the performance of transaction processing.

The remainder of the paper is organized as follows. The assumed network model, as well as the distributed database model, is presented in Section 2. In Section 3, we propose two transaction processing methods based on DB-migration. Section 4 and Section 5 describe the mechanisms for concurrency control and recovery, respectively. Section 6 describes simulation results. In Section 7, we discuss several important issues pertaining to the possible extension of our proposed methods. Then, in Section 8, we demonstrate the effectiveness of our proposed methods in comparison with several previous works. Finally, we summarize the paper in Section 9. We note that some of the results of this paper have been reported in [4].

2 System Model

In this section, the basic system model used for discussing our proposed methods is described. Then, the communication times necessary to execute the migration-processing method and the fixed-processing method are evaluated.

2.1 Model

The system environment is assumed to be a distributed database system which is constructed in a broadband

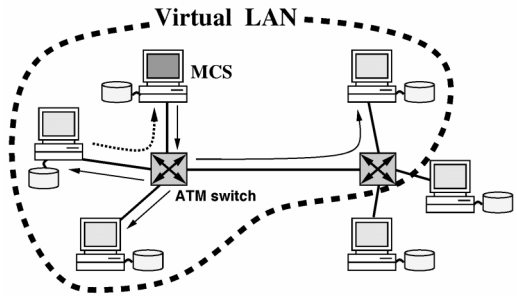


Fig. 1. VLAN and message broadcasting.

network with a broadcast facility such as ATM *virtual* LAN. In ATM networks, a virtual LAN (VLAN, for short) can be constructed independent of a physical network topology. A VLAN is realized by choosing a certain site as a *MCS* (multicast-server) and establishing a point-to-multipoint PVC connection from the MCS to each site on the VLAN. Each site can broadcast data through the VLAN by sending data to the MCS (see Fig. 1). ATM VLAN is considered to be the most suitable network for our proposed methods because its mechanism for bandwidth reservation is desirable for the burst transmissions caused by DB-migration. Also, the broadcast facility in ATM-VLAN enables easy system management. In the following, we assume an ATM VLAN for simplicity of discussion. It should be noted that we are not limited to only an ATM-VLAN. We only assume a broadband network with a broadcast facility and that a broadcast message is first sent to a specific site such as a MCS.

All sites in the network participate in the distributed database system and each site has its local database management system. The entire database is partitioned into a number of local databases where each local database is initially distributed to a particular site. In what follows, the term *database* is used for a local database for purposes of simplicity. We do not assume any particular data model or any partitioning strategy for the database.

In this system, we also make the following assumptions:

- Except for the broadcast messages in the VLAN, all communication is performed over a point-to-point SVC connection which is established on demand.
- If a SVC connection is established once for processing a transaction, it is not necessary for the connection to be re-established during the execution of the transaction.
- We assign a unique *site identifier* to each site in the system. The set of all sites in the system is denoted by $\boldsymbol{S} = \{S_1, S_2, \cdots, S_n\}$, where n is the total number of sites and $S_j (1 \le j \le n)$ is a site identifier.
- We assign a unique *database identifier* (DB-id) to each database located in the system. The set of all databases is denoted by $\boldsymbol{D} = \{D_1, D_2, \cdots, D_m\}$, where m is the total number of databases and $D_j (1 \le j \le m)$ is a DB-id.
- Let d denote the average propagation delay between two arbitrary sites, and let d_{MCS} denote the average propagation delay between the MCS and an arbitrary site.
- Database access does not become a performance bottleneck for DB-migration. Since the rapid development of memory technologies will allow computers to commonly have a main memory in the gigabyte range in the near future, very high-speed database access will be achievable using main memory database system techniques [2], [3]. Even using a main memory database, memory bandwidth or computational power may become a performance bottleneck if those at the end-system are poor. However, considering the rapid and constant developments of computer systems, it seems that memory bandwidth and computational power will soon catch up with network bandwidth. Thus, we also assume that they do not become a performance bottleneck for DB-migration.
- The migration of indexes attached to a database can be carried out by applying the mechanism we proposed in [6].

2.2 Communication Time

In this subsection, we estimate the communication times of the fixed-processing method and the migration-processing method. These estimations are used for the adaptive selection of transaction processing methods in Section 3. Since the exact analysis of the communication time of a transaction becomes very complex, we analyze the approximate communication time.

2.2.1 Fixed-Processing Method

A communication process of the fixed-processing method is illustrated in Fig. 2. First, before exchanging operation request/reply messages, we must establish a SVC connection from the transaction initiation site to each site that holds databases relevant to the execution of the transaction. Let k be the number of sites holding relevant databases, and let $C(k)$ denote the time required for establishing all the SVC connections from the transaction initiation site to the k sites to be involved in the transaction. Note that if we use the fixed-processing method in combination with the migration-processing method, the transaction initiation site must know the location of each database. The method by which each site knows the database locations is described in Section 2.2.2.

Once the connection is established, we assume that n messages are exchanged for request/reply operations, which takes time nd. Note that the transmission delay of the message is negligible because the size of each message is

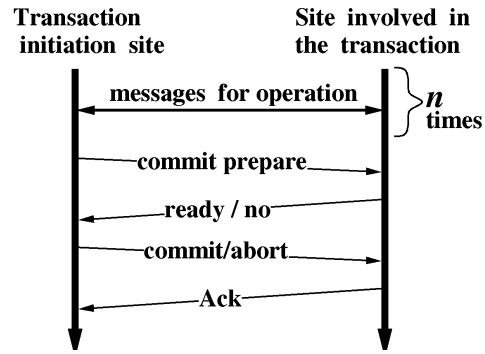


Fig. 2. Communication process of the fixed-processing method.

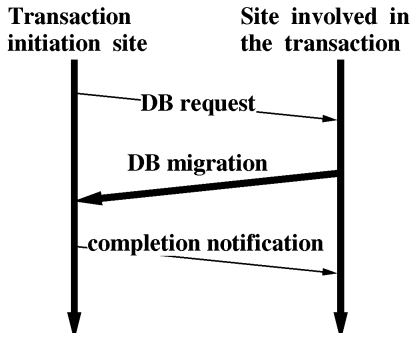


Fig. 3. Communication process of the migration-processing method.

very small compared with the bandwidth of the assumed communication channel. Finally, if the transaction is not read-only, two rounds of message transmissions for the 2PC are performed, which takes time 4*d*. Since the second round for two phase commit can be done asynchronously, it is difficult to estimate the effective propagation delay of these rounds in practice. For simplicity, we assume the delay to be 4*d* however this is not a precise estimate. Also, optimizations such as "early prepare" are not considered here.

Thus, the communication time required for the fixed-processing method, denoted by T_{fix}, is expressed by the following equation:

$$T_{fix} = \begin{cases} (n+4)\cdot d + C(k) & \text{(not read-only transaction)} \\ n \cdot d + C(k) & \text{(read-only transaction)} \end{cases} \quad (1)$$

2.2.2 Migration-Processing Method

As shown in Fig. 3, the migration-processing method can be realized using three communication operations:

1) a message transmission from the transaction initiation site requesting the databases involved in the transaction (*DB request*),
2) database transmissions from the sites which hold the databases involved in the transaction (*DB migration*), and
3) a message transmission from the transaction initiation site notifying that DB-migration has been completed (*completion notification*).

Let us call the databases involved in each transaction the *target databases*.

First, the broadcast message for DB request is transmitted. This process takes $2d_{MCS}$ amount of time since the broadcast message should be transmitted once to the MCS from which it is disseminated to all the destination sites. Note that it is difficult to formulate the exact propagation delay from the MCS to each site at which target databases reside because there are sites which do not have any target database and the delay to them is not considered. However, for simplicity of analysis, we have assumed this propagation delay is d_{MCS} overall.

Since the DB request message is sent to every site on the VLAN, all sites can get information on the location of each database associated with an invoked transaction. In this way, it is possible for each site to maintain information on database locations.

Following the transmission of the DB request message, SVC connections are established, from k sites which hold the target databases, to the transaction initiation site. As in the case of the fixed-processing method, this process takes $C(k)$ amount of time. After the connections have been established, the databases are transmitted to the transaction initiation site through the SVC connections. Let $Size(D_j)$ denote the size of database D_j, $\boldsymbol{D}_N$ denote a set of the databases which do not reside at the transaction initiation site but are necessary for processing the transaction, and B_M denote the reserved bandwidth for DB-migration. Since the transmission delay for DB-migration is not negligible even if we use broadband networks, this process costs

$$d + \sum_{D_j \in \boldsymbol{D}_N} Size(D_j) / B_M .$$

Finally, a broadcast message is sent for the completion notification of DB-migration. This process takes $2d_{MCS}$ amount of time, as in the case of the transmission time for the DB request message. Since this message is forwarded to every site, all sites can recognize the completion of the database relocation. After this message is received, the sites which transmitted the databases can delete the database copies they had kept in preparation for a migration failure.

In conclusion, the communication time of migration-processing method, denoted by T_{DB}, is expressed by the following equation:

$$T_{DB} = d + 4d_{MCS} + C(k) + \frac{\sum_{D_j \in \boldsymbol{D}_N} Size(D_j)}{B_M}. \quad (2)$$

3 Transaction Processing Method Based on DB-Migration

The results obtained in Section 2.2 show that the transaction processing time of the fixed-processing method depends on the number of message transmissions (n) and the processing time of the migration-processing method depends on the total size of the databases

$$\left(\sum_{D_j \in \boldsymbol{D}_N} Size(D_j) \right).$$

Based on this fact, we propose two transaction processing methods which adaptively choose either the fixed-processing method or the migration-processing method. However, if the system cannot reserve sufficient bandwidth for DB-migration, the fixed-processing method is selected.

3.1 Simple Method

Let us define the variable t_1 as follows:

$$t_1 = T_{DB} - T_{fix}. \quad (3)$$

Then, in the simple method, the processing method is chosen according to the following rule:

if $t_1 < 0$ then use the migration-processing method
else use the fixed-processing method

That is, according to Equation 1 and Equation 2, this method chooses the processing method with a shorter communication time for the given parameter values. To obtain the value of t_1, we have to evaluate

1) the location of each database, and
2) the value of

$$\sum_{D_j \in \boldsymbol{D}_N} Size(D_j).$$

Thus, every site on the VLAN must have an information table which maintains the size and the location of each database. Then the value of

$$\sum_{D_j \in \boldsymbol{D}_N} Size(D_j)$$

can be calculated after the databases involved in the transaction are identified. Note that it is inefficient to broadcast the updated size information for the information table whenever the size of databases changes. Therefore, the broadcast message is sent only when the difference between the real size of each database and the corresponding size information in the information table exceeds a certain threshold value.

3.2 Log-Statistics Method

Although the simple method chooses the more efficient method to process a transaction, this choice may be inefficient in the long run. For example, even if the fixed-processing method is estimated to be efficient for processing a single transaction, it is considered to be more efficient to use the migration-processing method at the beginning of transaction processing if the transaction initiation site continuously initiates transactions which use the same databases. Conversely, even if the migration-processing method is chosen, it is considered efficient to use the fixed-processing method if the target databases will be used continuously by the site where the databases currently reside.

The above consideration leads us to propose the log-statistics method for method selection, where the following two points are considered:

- If a site S_I frequently uses the database D_J, the priority for S_I to have D_J should be increased.

  (If the access pattern of each site dynamically changes, the priority for S_I to have D_J also dynamically changes.)

- If it is known that a site S_I will use the database D_j continuously, the priority for S_I to have D_j should be increased.

The log-statistics method chooses either method, not only by Equation (3), but also by using information on the *usage log* and the *continuous-use declaration* maintained at every site. Here, the continuous-use declaration is made to a database when the transaction initiation site uses the database continuously in the next transaction. In the log-statistics method, each site holds an information table which we call the *DB-information table*. Fig. 4 shows an example of the DB-information table. Each row corresponds to the information for one database. The attributes of the table are as follows:

DB-id: the database identifier.

Size: the size of the database.

Current Loc: the site identifier where the database currently resides.

Cnt: 1 if the site where the database currently resides makes a continuous-use declaration, 0 otherwise.

Usage-log: the history of the last several transactions: the site identifier of each transaction initiation site is listed successively if the database was used in the transaction (one column indicates one transaction).

Only the history of several recent (constant value) transactions is maintained while older information is discarded. When the system has a relatively uniform access pattern, a larger history size gives better performance since it helps to predict the precise access pattern. However, maintaining too large a history requires very large memory space and may result in a performance decrease. When the access pattern of the system frequently changes, the history size should be set to a smaller value.

DB-id	Size	Current Loc	Cnt	Usage-log				
D_1	50	S_1	1	S_1		⋯	S_2	S_1
D_2	35	S_2	0	S_1	S_4	⋯		
⋮	⋮	⋮	⋮			⋮		
D_m	40	S_4	1			⋯	S_2	S_1

Fig. 4. DB-information table.

Since the *Cnt* and *Usage-log* are updated by every transaction initiation, it is necessary to broadcast the information on such updates in every transaction. When the migration-processing method is executed, the update of the *Usage-log* can be determined directly by the contents of the broadcast messages (i.e., DB request and completion notification) and the information update of *Cnt* can be forwarded together with those broadcast messages. On the other hand, when the fixed-processing method is executed in the log-statistics method, in order to update such information in the same way as that of migration-processing method, we must broadcast the information on the target databases and the continuous-use declaration at the time the transaction is initiated. Note that this information is transmitted together with a broadcast message for a lock request which is used to avoid deadlock, and therefore the communication time becomes larger by $2d_{MCS}$ than the communication time expressed by Equation (1). Then, the communication time of the fixed-processing method in our proposed methods is expressed by the following equation:

$$T'_{fix} = \begin{cases} (n+4)\cdot d + 2d_{MCS} + C(k) & \text{(not read-only transaction)} \\ n \cdot d + 2d_{MCS} + C(k) & \text{(read-only transaction)} \end{cases} \quad (4)$$

Now, we define the objective function $f(S_I, D_j)$, which expresses the effectiveness that the database D_j resides at the site S_I, by the following equation:

$$f(S_I, D_j) = \alpha \cdot P \cdot |L| + \sum_{l=1}^{|L|} \{k_l \cdot (|L| + 1 - l)\} \quad (5)$$

where

$$\alpha = \begin{cases} 1 & \text{if the site } S_I \text{ has made the continuous-use declaration for } D_j \text{ in previous transactions or makes it in the current transaction,} \\ 0 & \text{otherwise} \end{cases}$$

P: the coefficient of the priority for continuous-use declaration (*PCUD coefficient*).

$|L|$: the size of the database usage log

(the history of the last $|L|$ transactions is recorded).

$$k_l = \begin{cases} 1 & \text{if site } S_I \text{ used } D_j \text{ in the transaction executed } l \text{ times before,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that $f(S_I, D_j)$ reflects the two points mentioned earlier, i.e., the first term expresses the priority for the continuous-use declaration and the second term expresses the database usage frequency estimated from the database usage log (more recent use of D_j is given higher priority). The PCUD coefficient determines the degree of priority given to the continuous-use declaration. $|L|$ is the scale factor used for balancing the value between the first term and the second term.

Then, we define the evaluation function $G(S_I, D_j)$ by the following equation, which expresses the effectiveness of making the database D_j migrate to the site S_I from the site S_C where D_j currently resides.

$$G(S_I, D_j) = f(S_I, D_j) - f(S_C, D_j). \quad (6)$$

Based on the definition of $G(S_I, D_j)$, the evaluation value t_2 is expressed as the average value of $G(S_I, D_j)$ for each $D_j \in \boldsymbol{D}_N$ when the site S_I initiates the transaction and requires a set of databases $\boldsymbol{D}_N$, as follows:

$$t_2 = \frac{1}{|\boldsymbol{D}_N|} \cdot \sum_{D_j \in \boldsymbol{D}_N} G(S_I, D_j). \quad (7)$$

Here, t_2 expresses the effectiveness to process the current transaction by the migration-processing method which is initiated by the site S_I.

Finally, we define the evaluation value t_{sel}, which gives the criterion for choosing one of the two methods i.e., the fixed-processing method or the migration-processing method:

$$t_{sel} = t_1 - K \cdot t_2. \quad (8)$$

Thus, in the log-statistics method, the processing method is chosen according to the following rule:

if $t_{sel} < 0$ then use the migration-processing method
else use the fixed-processing method

We call the coefficient K the *log-dependent coefficient*. Note that the value of K greatly affects the performance of the whole distributed system. Therefore, this value should be determined by database managers who have an indepth knowledge of system operations, such as the database access patterns of the transactions and the system scale.

In general terms, K and P should be large values if the system scale is large, i.e., the distance among sites is large and the size of databases is large. This is because the purpose of K and P is to do the mapping from log information to (communication) time. Also, if the access pattern of the system is not skewed (i.e., each site initiates transactions in almost the same pattern) and only continuous transactions exist, P should be relatively large value compared to K.

4 Mechanism for Concurrency Control

This section describes a mechanism of concurrency control for our proposed methods. Our mechanism differs significantly from that of a conventional distributed database systems due to the introduction of DB-migration. Since DB-migration is an operation which occupies a database for some time, transaction processing throughput may deteriorate in environments where data contention is a significant factor. To prevent this deterioration, our proposed methods must adopt the following strategies for concurrency control as well as conventional concurrency control for read and write operations on data items:

- A strategy to avoid deadlock among several DB-migration requests.
- A strategy to support read and write operations to a migrating database.

Moreover, in order to maintain the independence of the method selection mechanism and the concurrency control mechanism, we must use the following strategies:

- A strategy for prevention of DB-migration from the transaction initiation site to another site when the transaction initiation site has gathered these databases and is executing the transaction processing (migration-processing method).
- A strategy for selection of the transaction processing method considering the transactions waiting for their turns to be processed. In other words, the information, on the future locations of the target databases and on the database usage log, at a point in time when the initiated transaction is executed, must be precisely predicted.

To support the above four strategies, we extend the conventional two phase locking protocol (2PL) by adding two new locks, the *move* lock for data items and the *location* lock for databases, to the conventional read and write locks. Table 1 shows the compatibility matrix of these locks. In this table, for example, the 'write (state)-move (request)' square indicates that a write lock is already set to a data item and a request to set a move lock to the data item is initiated. In this case, the request is rejected (F).

TABLE 1
LOCK COMPATIBILITY MATRIX

request	state			
	read	write	move	location
read	T	F	T	T
write	F	F	T	T
move	T	F	F	F
location	T	T	—	—

T (True), F (False)

In our proposed methods, a transaction is processed in the following steps:

1) At the time a transaction is initiated, a broadcast message including each lock request necessary for processing the transaction is transmitted. In the case that the transaction is processed by the migration-processing method, this message can be added to the message for the DB request.
2) Utilizing the fact that a broadcast message is first transmitted to the MCS, a sequence number, which we call a *transaction identifier* (Tid), is assigned to the request message (i.e, to the initiated transaction) at the MCS.
3) At each site that holds database(s), the *lock queues* (for read, write, and move locks) are maintained for every data item of the database(s). For example, if a database consists of 10 data items, 10 lock queues are maintained for the database. The lock requests accepted from the transaction initiation site are inserted into the appropriate queues. Note that the move lock is inserted to all queues of the target database. Our assumption of large lock granularity prevents the total size of the lock request queues becoming too large when operations for a whole database occur frequently. On the contrary, if a lock level is set too small, the lock queues require a very large space and thus this causes a serious performance degradation.
4) The transaction initiation site knows the Tid assigned to its transaction based on the message returned from the MCS which the site broadcasts for the lock request. This Tid is recorded to every message for operation requests in the transaction, and then the site begins to transmit the messages to the appropriate sites.
5) At each site that holds the target databases, database operations are executed based on the compatibility matrix of locks shown in Table 1. DB-migration occurs when the move lock is set to all data items of the target database. As shown in Table 1, the move lock cannot be set where the write lock has already been set. Therefore DB-migration does not occur too frequently and each transaction is processed nearly in the order in which it was initiated.

In the rest of this section, we discuss how the above-mentioned four strategies are realized based on these processing steps.

4.1 Avoidance of Deadlock

In a single site database system, deadlock can be avoided using the pure 2PL, i.e., it can be avoided by setting all necessary locks for the transaction in advance of each transaction execution. However in a distributed database system, deadlock can occur when the arrival order of the messages for lock requests are reversed at several sites. This can occur even if each transaction transmits a broadcast message for all lock requests in advance transaction execution. For example, suppose that there are two transactions which are initiated from the sites S_1 and S_2 at almost the same time but S_1's transaction is initiated slightly earlier. Here, we assume that both transactions require two operations (write or move) to D_1 and D_2 and arrivals of lock requests to D_2 are reversed for some reason, i.e., a lock from S_2 arrives earlier than that from S_1. In such a case, S_1 has set a lock to D_1 and is waiting to set a lock to D_2, and S_2 has set a lock to D_2 and is waiting to set a lock to D_1. This is a typical case of deadlock in a distributed database system.

Here, since the Tid allocated at the MCS is a sequence number, the transactions can be put in a unique order. If somehow the arrival order of the messages for lock requests are reversed at several sites, we can revise the order by comparing the Tids. Therefore, deadlock cannot occur.

4.2 Method Selection Considering the Waiting Transactions

To implement this strategy, each site in a system must precisely predict the future information necessary for method selection. In the simple method, each site monitors the broadcast messages for DB requests (i.e., requests for locks) from other sites and then predicts the future locations of the databases when all waiting transactions are processed. Thus, each site maintains information on both the real current locations and the future locations of the databases. Method selection is performed using the latter information.

In the log-statistics method, each site also monitors the broadcast messages for lock requests in the case of the fixed-processing method and then predicts the future database usage log when all waiting transactions are processed. The site also maintains both information on the database usage log of the last $|L|$ transactions, which have already been processed, and that of the last $|L|$ transactions which have arrived at the system. Method selection is performed using the future information on the location and the database usage log. The predicted future information is correct unless DB-migration fails or a transaction abortion occurs. In the next section, we will discuss the recovery from such cases.

The system also avoids improper DB-migration caused by incorrectly predicted information on the database locations. The transaction initiation site adds the predicted locations of the target databases to the broadcast message for a DB request (lock request). At each site that holds the target databases, the real future locations caused by the last DB request and the predicted locations in the new accepted message are compared. Only when the comparison shows that the locations are the same is the move lock

inserted to the lock queues of all the data items of the target databases. Otherwise, the DB request is cancelled.

4.3 Support for Operations to a Migrating Database

While a database is migrating, its copy is maintained at a sender site preparing for migration failures. In this way, the system can continue to execute the read and the write operations to the migrating database by operating on the copy. After confirming the completion of DB-migration, the system stops the operation on the copy and sends the log information for the operations executed on the copy to the database receiver site. Then, the operations executed on the copy are reflected in the real database. After completing a series of the above processes, the system starts to execute the operations on the database at the database receiver site. The operation request messages, which arrive at the sender site after sending the log information, are forwarded to the database receiver site.

4.4 Protection of DB-Migration from Other Transactions

In our proposed methods, after the move locks have been set to all data items of a database, DB-migration starts and all their move locks are unset. Moreover, the location lock is set to the database. As shown in Table 1, the move lock cannot be set where the location lock is set to the database. Since the location lock is not removed until the transaction for gathering the databases using DB-migration is completed, the transaction initiation site can process the transaction centrally. This eliminates the possibility of losing any databases due to their removal being initiated by other sites.

5 Recovery Mechanism

5.1 Database Recovery

Since we assume main memory databases and main memory is a volatile resource, an efficient recovery method is essential. Several recovery methods for main memory database have been proposed such as those in [3]. Most of these methods record only the redo logs on disk to reduce disk I/O. In such methods, redo and undo logs are maintained in main memory while the transaction is active, and when the transaction commits, the redo logs are recorded on disk and the undo logs are discarded. When a memory failure occurs and the database is lost, the database is recovered using its disk backup and its recorded redo logs. Contrary to conventional disk-based database systems, a database buffer is not necessary and thus, undo logs need not be recorded on disk because it is not essential to maintain consistency between the database buffer and the database itself when the failure occurs. The undo logs in memory are used only to recover the database when the transaction aborts. Note that since undo log are not written on disk, the checkpointing has to be performed atomically.

In such methods, the following processes are performed at the checkpoint and recovery point.

Checkpoint:

- The transaction identifier of the last transaction committed before the checkpoint is written to disk.
- The end address of the last redo log written to disk before the checkpoint is recorded on disk.
- The backup (on disk) of the database is updated and the redo logs, which are maintained on disk, are discarded.

Recovery point:

- From the backup database on disk and the redo logs which were written after the last checkpoint, the database on memory is recovered.

Conventional recovery mechanisms such as those in [3] are used in our case to maintain database consistency against system failures and/or transaction abortions. However, conventional methods should be extended to cope with DB-migration. From a performance perspective, we must decide the following points according to the features of the system:

1) Whether the backup of a database on disk is maintained at a particular site (*home site*) or it also migrates to the database receiver site when the database migrates.
2) Whether the redo logs on disk are maintained at the sites where the corresponding updates are performed or if the logs are collected at the site where the database currently resides.

These decisions significantly affect the processing time at the checkpoint and the recovery point. We propose three methods for making these decisions.

5.1.1 The Complete Shadowing (CS) Method

When a database migrates, both its backup and redo logs on disk also migrate to the database receiver site in the following manner. The sender site transfers the redo logs which were recorded after the last checkpoint (as well as the information which was recorded at checkpoints) together with the database. Then, the receiver site creates the backup from the received database and writes the received redo logs (and the other information) to disk. Here, the received information is also maintained in memory to speed up the execution of checkpoints and the next DB-migration.

In the CS method, when a checkpoint is performed or a failure occurs, DB-migration does not cause any special operations. The recovery process is the same as that of the conventional method (see Fig. 5a).

5.1.2 The Log Shadowing (LS) Method

When a database migrates, only its redo logs migrate to the database receiver site. This method is the same as the CS method except for one point: It does not create the backup. In the LS method, since the backup resides at a particular site (the home site), the operations at the checkpoint and recovery point should be changed as follows:

Checkpoint:

- If the backup resides at a remote site, the redo logs are transferred to the (home) site. Then, the home site updates the backup based on the received redo logs.

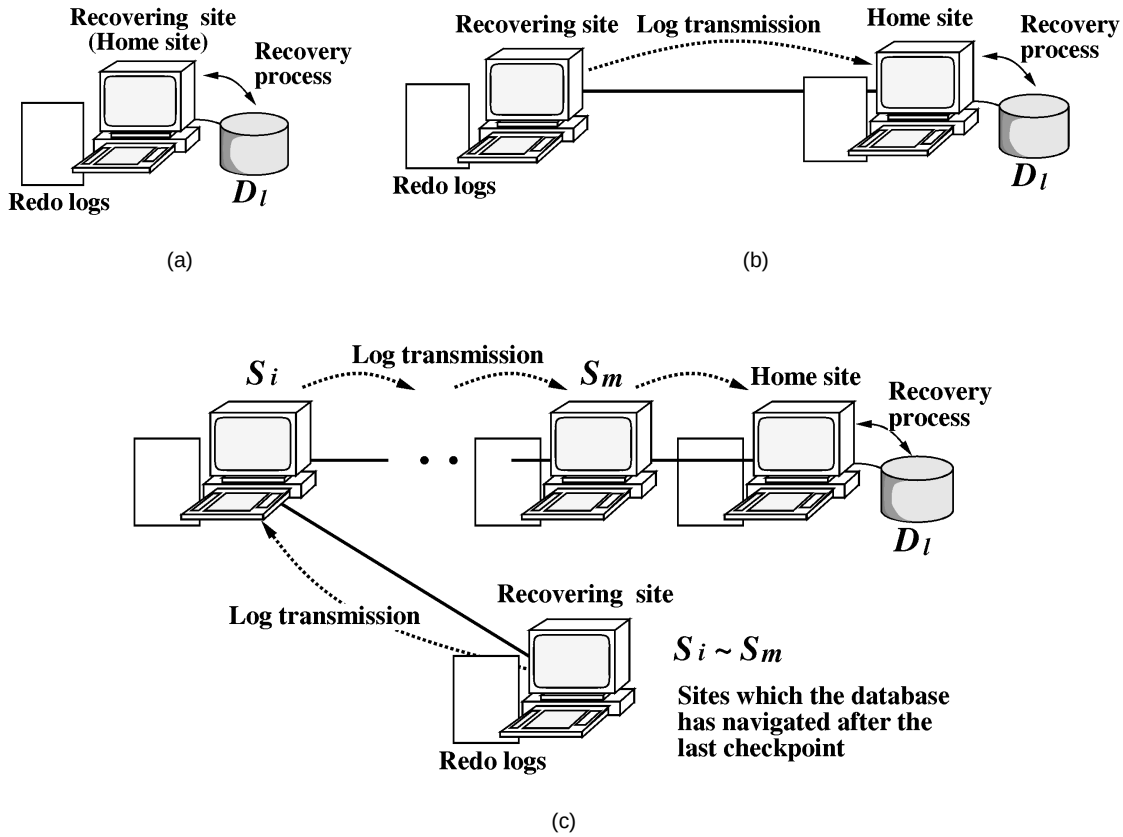


Fig. 5. Recovery process of the three methods: (a) the CS method, (b) the LS method, and (c) the NS method.

Recovery point:

- If the backup resides at a remote site, the redo logs are transferred to the home site. Then the home site recovers the database based on its backup and the received redo logs (see Fig. 5b).

5.1.3 The No-Shadowing (NS) Method

When a database migrates, neither the backup nor the redo log migrate to the receiver site. The redo logs are maintained at each site where the corresponding updates are performed. Therefore, the operations in checkpoint and recovery point should be changed as follows:

Checkpoint:

- By navigating backward through the migration sites used by the database after the last checkpoint, the redo logs are collected. After that, the collected redo logs are transferred to the home site. Then, the home site updates the backup based on the received redo logs.

Recovery point:

- By navigating backward through the migration sites used by the database after the last checkpoint, the redo logs are collected. After that, the collected redo logs are transferred to the home site. Then, the home site recovers the database based on its backup and the received redo logs (see Fig. 5c).

5.1.4 Discussion

These three recovery methods significantly differ from each other in the time costs incurred by the additional functions related to DB-migration and checkpoint. As for the additional costs of DB-migration, the CS method requires the longest time followed by the LS method. Since the CS method writes the backup to disk at every migration, this method requires a much longer time than the other two methods. On the other hand, for the additional costs of the checkpoint, the NS method requires the longest time followed by the LS method. Since the NS method requires many message transmissions, more time is required than for the other two methods. In a practical environment, we should select the optimal recovery method based on the features of the system.

5.2 Recovery of Information for Method Selection

As mentioned above, in the log-statistics method, the transaction initiation site must predict the future information on the locations of the target databases and the database usage log. Considering the fact that transaction abortions and DB-migration failures affect such information, we must additionally consider the recovery of the information. In our proposed methods, the following three levels are provided for the recovery:

Level 1:

- Recovery is not executed when either transaction abortions or DB-migration failures occur.

Level 2:

- If a DB-migration failure occurs, the transaction initiation site (i.e., the receiver site of the DB-migration) transmits a broadcast message to notify the other sites of the failure. At each site that holds a database, the site cancels lock requests from the transactions which have Tids larger than that of the transaction which has experience DB-migration failure. Moreover, at every site, the future information on the database usage log is discarded. Then, the site re-initiates the transaction using the information on the real location of the database and the database usage log of the last $|L|$ transactions which have already been processed.
- Recovery is not executed for transaction abortions.

Level 3:

- The same recovery as level 2 is executed in the case for DB-migration failures.
- If a transaction abortion occurs, the transaction initiation site broadcasts the abort message to all sites instead of transmitting this message only to the sites at which the target databases reside. At each site that holds a database, the site cancels lock requests from the transactions whose Tids are larger than that of the aborted transaction. Moreover, at every site, the future information on the database usage log is discarded. Then, the site re-initiates the transaction using the actual database location information and the database usage log of the last $|L|$ processed transactions.

In environments where transactions are not frequently initiated (i.e., there are few transactions that are waiting to be processed at any given instant), the recovery level should be set to level 1 since it has the least overhead. On the other hand, in environments where transactions are frequently initiated, the recovery level should be carefully chosen. The accuracy of the database location information significantly affects the transaction processing time. Hence, we should seriously consider whether or not to execute the recovery of DB-migration failures. In general, the system does not execute the recovery of DB-migration failures in environments where transactions are frequently initiated and the ratio between the frequency of DB-migration failure and transaction initiation (the *migration-failure/transaction ratio*) is high. In these cases, the recovery level is set to level 1. In environments where transactions are frequently initiated and the migration-failure/transaction ratio is low, and if transaction abortions and transaction initiations are frequent, the recovery level is set to level 2. Otherwise, it is set to level 3.

Note that, in the simple method, the system provides only two recovery levels for transaction abortions since this method does not use the database usage log.

6 Simulation

In this section, we present simulation results regarding the performance evaluation of our proposed methods.

6.1 Simulation Details

We assume that a distributed database system is constructed in an ATM VLAN. The basic parameters used in the simulations are summarized in Table 2. The number of database sites and the number of databases, $|\boldsymbol{S}|$ and $|\boldsymbol{D}|$, are 20. The parameters for data transmission are based on the prospect of distributed systems of a worldwide scale being available in the near future. More specifically, the values of the propagation delays, d and d_{MCS}, represent the worldwide ATM VLAN with fiber optic cables in which the average hop count between two arbitrary sites is about 10, and the average switching delay at an arbitrary ATM switch is 5 milliseconds (this value is based on the switching delays of the commercial ATM switches of currently available). The value of the bandwidth reserved for DB-migration, B_M, is based on the prospect that network bandwidth will soon become a few gigabits/second everywhere in the world. While it is difficult to know the available end-to-end bandwidth in advance, we assume no reservation failure occurs in the simulations. In practical environments, if the bandwidth reservation had failed, the system would try to reserve less bandwidth than the first reservation request or give up the migration-processing method. In the research field of network technology, several methods have been proposed to determine the available end-to-end bandwidth in advance. If such methods can be applied in practice, we will incorporate those methods with our proposed methods.

As for the characteristics of databases and transactions, we make the following assumptions:

- Each database consists of 30 data items. If the data model is relational, a data item may be a single relation.
- The assumed size of the whole database,

$$\sum_{D_j \in \boldsymbol{D}} Size(D_j),$$

  is about 0.9 gigabytes.
- Each transaction consists of a sequence of at least one write operation. A write operation can only be executed after the completion of the previous operation in the sequence. This assumption is made for purposes of simplicity because the impact of the read/write rate on system performance depends on the system features.
- Because our proposed methods use main memory databases, the execution time of each write operation is far shorter compared with the communication time for transaction processing. Thus, we assume that the operation execution time is negligible. Moreover, we assume that the I/O time for DB-migration is negligible for the same reason.
- Both system failures and transaction abortions do not occur.

The wide value range of n' is used to represent transactions of various complexity. These parameters, including those in Table 2, affect the communication time of the fixed-processing method or the migration-processing method. We

TABLE 2
PARAMETER CONFIGURATION FOR THE SIMULATION EXPERIMENTS

parameter	meaning of the parameter	value
$\|S\|$	number of the sites in the VLAN	20 (site identifier S_1, ⋯, S_{20})
$\|\boldsymbol{D}\|$	number of databases in the VLAN	20 (DB-id D_1, ⋯, D_{20})
$Size(D_j)$	size of the database D_j	$30 + 1.5_{j-1}$[Mbytes] (j = 1, ⋯, 20)
p_i^1	transaction initiation probability 1 at site S_i	0.025 (i = 1, ⋯, 10), 0.05 (i = 11, ⋯, 18), 0.15 (i = 19), 0.3 (i = 20)
p_i^2	transaction initiation probability 2 at site S_i	0.025 (i = 11, ⋯, 20), 0.05 (i = 3, ⋯, 10), 0.15 (i = 2), 0.3 (i = 1)
d_{MCS}	propagation delay from the transaction initiation site to the MCS	0.12 [seconds]
d	propagation delay between two sites	0.12 [seconds]
B_M	reserved bandwidth for migrating the databases	1.0 [Gbps]
$C(k)$	time required for establishing SVC connections	assume that $C(k) = 0.3k$ [seconds]
n'	number of operations	determined as an integer value at random from 1 to 30.
$\|L\|$	size of the usage log	20

have determined these parameters to model an environment where the average communication times of the fixed-processing method and that of the migration-processing method are almost the same. Thus, if we change the parameters, and the communication time of one method becomes much longer than that of the other one, the system performance as a whole approaches the case whereby only one method is used. This results in a shorter communication time because of the adaptive nature of our proposed methods.

In our simulation experiments, 10,000 transactions are processed. We changed the transaction initiation probability of each site S_i in every 1,000 transaction initiations between p_i^1 and p_i^2 in order to examine the adaptability of our proposed methods. Based on this parameter configuration, our simulation studies were executed using the following steps:

1) A transaction initiation site S_I is determined according to the transaction initiation probability of each site.
2) $|\boldsymbol{D}_U|$, the number of target databases, is determined as an integer value at random from 1 to $|\boldsymbol{D}|$. ($|\boldsymbol{D}_U| \geq |\boldsymbol{D}_N|$: $\boldsymbol{D}_U$ is a set of the target databases, and $\boldsymbol{D}_N$ is a set of the databases which do not reside at the transaction initiation site but are necessary for processing the transaction. Thus, $\boldsymbol{D}_N$ is a subset of $\boldsymbol{D}_U$.)

   If S_I has already made a continuous-use declaration for certain databases, and the value of $|\boldsymbol{D}_U|$ is smaller than the number of databases which have been declared for continuous use, a new value is assigned to $|\boldsymbol{D}_U|$ such that the value is equal to the number of these databases.
3) As the target databases, $|\boldsymbol{D}_U|$ databases are selected. Note that if S_I had made a continuous-use declaration in the previous transaction, those databases declared for continuous use must be included in these $|\boldsymbol{D}_U|$ databases.
4) The number of write operations n' is determined as an integer value at random from 1 to 30. Note that, among n' write operations, $n'|\boldsymbol{D}_N|/|\boldsymbol{D}_U|$ (operations) are for the databases $\boldsymbol{D}_N$ and the remainder are for the databases $\boldsymbol{D}_U - \boldsymbol{D}_N$. Then, within this constraint, the databases and the data items of the databases which are the targets for the operations are selected randomly.
5) The transaction continuity parameter β is determined. This parameter is determined randomly from the values 0, 1, and 2. It declares how many transactions S_I will be successively initiated after the completion of the current transaction.
6) If the value of β is larger than 0, the transaction initiation probability of S_I increases by 1.00 in the successive β transaction(s). Then, the continuous use databases are determined at random from the target databases.
7) In the case of our proposed adaptive methods, the fixed-processing method or the migration-processing method is chosen.
8) The transaction is processed according to the chosen method as per our proposed adaptive methods.

6.2 Evaluation of Our Proposed Methods

As the first simulation experiment, we verify the effectiveness of the method selection of our proposed methods. In the simulations, we measure the average transaction processing time of the conventional fixed-processing method and the our proposed methods using an environment where no two transactions are processed concurrently (i.e.,

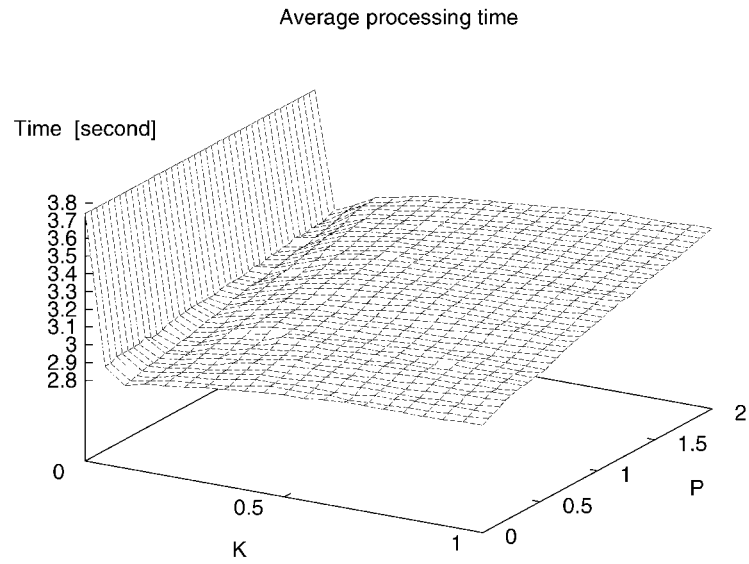


Fig. 6. Effects of *K* and *P* on the transaction processing time of the log-statistics method. (Note: *K* = [0, 1], *P* = [0, 2])

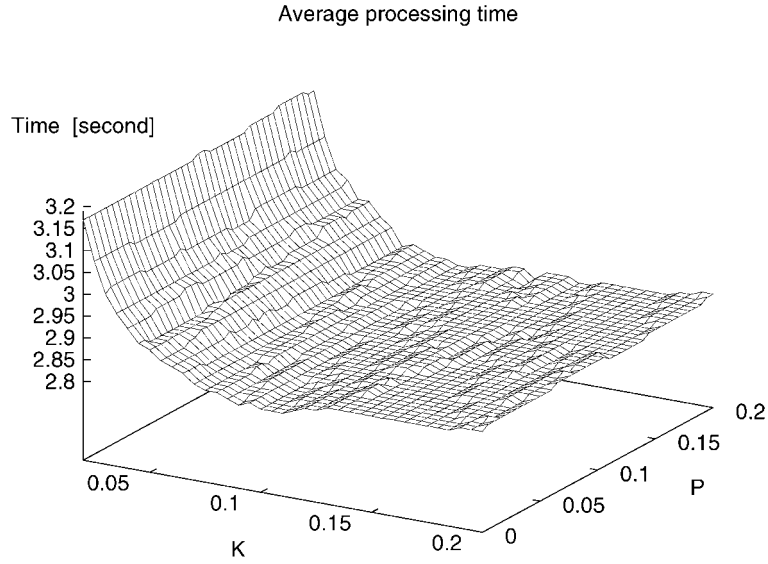


Fig. 7. Effects of *K* and *P* on the transaction processing time of the log-statistics method. (Note: *K* = [0.02, 0.20], *P* = [0, 0.20])

the concurrency control is not necessary). We also show how the PCUD coefficient *P* (which determines the degree of priority given to the continuous-use declaration) and the log-dependence coefficient *K* (which determines the degree of priority given to the access pattern information) of the log-statistics method affect the transaction processing time.

6.2.1 Effect of the Coefficient P and K

We investigate the effect of the PCUD coefficient *P* and the log-dependence coefficient *K* on the transaction processing time of the log-statistics method. System performance is too sensitive to changes of *K* and *P* to permit their optimal values from being analytically determined. Hence, simulation was used.

Fig. 6 shows the average transaction processing time of the log-statistics method. Here, we changed the value *P* by 0.05 from 0 to 2 and the value *K* by 0.05 from 0 to 1. The x-axis indicates the value of *K*, and the y-axis indicates the value of *P*. From the figure, it is shown that the processing time changes drastically according to the values of *P* and *K*. Good results are observed in the range of (*K*, *P*) = ([0.02, 0.20], [0.00, 0.20]) in this environment.

Now, to investigate the behavior in more detail, we evaluate the transaction processing time of the log-statistics method changing *K* by 0.05 from 0.02 to 0.20 and *P* by 0.05 from 0.00 to 0.20. Fig. 7 illustrates this simulation result. The transaction processing time indicates the lowest value (2.81 seconds) when (*K*, *P*) = (0.090, [0.000, 0.005]), (0.100, [0.000, 0.040]), (0.105, [0.000, 0.035]), and (0.110, [0.000, 0.010]).

To make good use of this method in real systems, the values of *P* and *K* must be carefully determined. As we

have used approximate analysis in this simulation, there may be differences between simulation results and the results obtained from real environments. Therefore we should choose the values of P and K which give good performance over a wide range in the simulation, rather than the values which give good performance within a narrow range.

6.2.2 Performance Comparison

Here, we compare the transaction processing time of the four methods: the fixed-processing method, the migration-processing method, the simple method, and the log-statistics method. We present two simulation results in the following environments:

E1: The environment expressed by Table 2, where P and K are set to 0.02 and 0.10, respectively. These values for P and K are nearly in the middle of the wide range which was shown to give good performance in Section 6.2.1. Note that E1 models an environment in which there are various transaction sizes, i.e., n' changes uniformly from integer values 1 to 30, and $|\boldsymbol{D}_U|$ also changes uniformly from integer values 1 to 20 in every transaction.

E2: The environment expressed by Table 2 except for the condition determining $|\boldsymbol{D}_U|$ and n'. Note that E2 models an environment in which the transaction size is almost uniform, i.e., n' and $|\boldsymbol{D}_U|$ are determined based on a normal distribution (cutting the negative range), where the mean and standard deviation of n' are 15 and 2.5, respectively, and those of $|\boldsymbol{D}_U|$ are 10 and 1.67, respectively. P and K are also given as 0.02 and 0.10, respectively.

Table 3 shows the simulation results. Each value in the table shows the average transaction processing time. These results demonstrate the effectiveness of our proposed methods. Among the four methods, the log-statistics method gives the shortest processing time in both environments. The performance of the simple method deteriorates in environment E2 since the method selection principle according to the communication time of a single transaction does not work effectively in the environment where the communication times of the fixed-processing method and that of the migration-processing method are almost the same in every transaction.

TABLE 3
AVERAGE PROCESSING TIME OF THE FOUR METHODS

	E1 [seconds]	E2 [seconds]
fixed-processing	5.52	5.20
migration-processing	5.41	5.19
simple method	3.74	5.19
log-statistics method	2.81	2.41

6.3 Evaluation of the Concurrency Control Mechanism

As for the second simulation experiment, we verify the effectiveness of the concurrency control mechanism of our proposed methods. In the simulations, we measure the average transaction processing time required for the conventional fixed-processing method and our proposed methods, where data contention is a significant factor.

The parameters used in the simulations are the same as those in Table 2 except for B_M: we execute the simulations in two cases B_M = 1.0 [Gbps] and 2.0 [Gbps]. The interval of transaction initiation is determined based on exponential distributions. Fig. 8 and Fig. 9 show the average transaction processing time for each of the four methods where the mean transaction interval was changed by 0.1 from 0.1 to 20.0. Fig. 8 and Fig. 9 show the results in the cases of B_M = 1.0 [Gbps] and B_M = 2.0 [Gbps], respectively. In both figures, the x-axis indicates the mean transaction interval.

From these figures, it is shown that the processing time of the migration-processing method is significantly affected by the available network bandwidth. It is also shown that, in both cases, the performance of the migration-processing method is the first to deteriorate as the transaction initiation interval gets shorter. This is because contention between DB-migration requests frequently occur in this method when transactions are initiated in short intervals. As for our proposed methods, both methods give almost the same performance for concurrency control as the fixed processing method. These results show that the concurrency control mechanism of our proposed methods prevents the performance deterioration caused by DB-migration in environments where data contention is a significant factor.

The simulation results also demonstrate the effectiveness of the adaptive method selection of our proposed methods. Among the four methods, the log-statistics method gives the shortest processing time in both cases where B_M = 1.0 [Gbps] and 2.0 [Gbps].

7 IMPLEMENTATION ISSUES

In this section, we discuss several implementation issues with respect to extending our proposed methods to WANs (Wide Area Networks).

7.1 Location Management

In our assumed environment, location management can be implicitly performed in the way we described so far, i.e., monitoring every broadcast messages. However, to extend our proposed methods to environments where a broadcast facility is not available such as ATM-WANs, an explicit database location management method is required.

Based on the location management methods for mobile computing such as those given in [10], [16], we have proposed several database location management methods in [5]. Some of the methods use a specific site as a *location management server* (LMS) which receives notification of new database locations after each migration. In the remaining methods, only sites participating in DB-migration monitor database location, and the location of the target database is detected by navigating the sites through which the target database has migrated. It is possible to extend our proposed methods to environments where a broadcast facility is not available by incorporating the methods mentioned above.

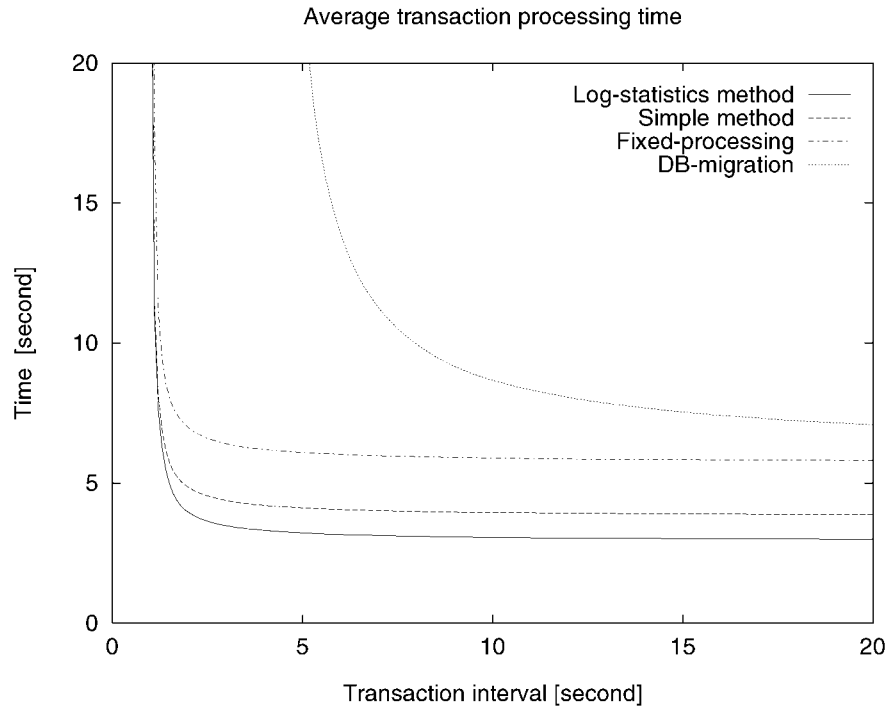


Fig. 8. Relation between the transaction initiation interval and the transaction processing time of the four methods. (Note: bandwidth = 1.0 [Gbps])

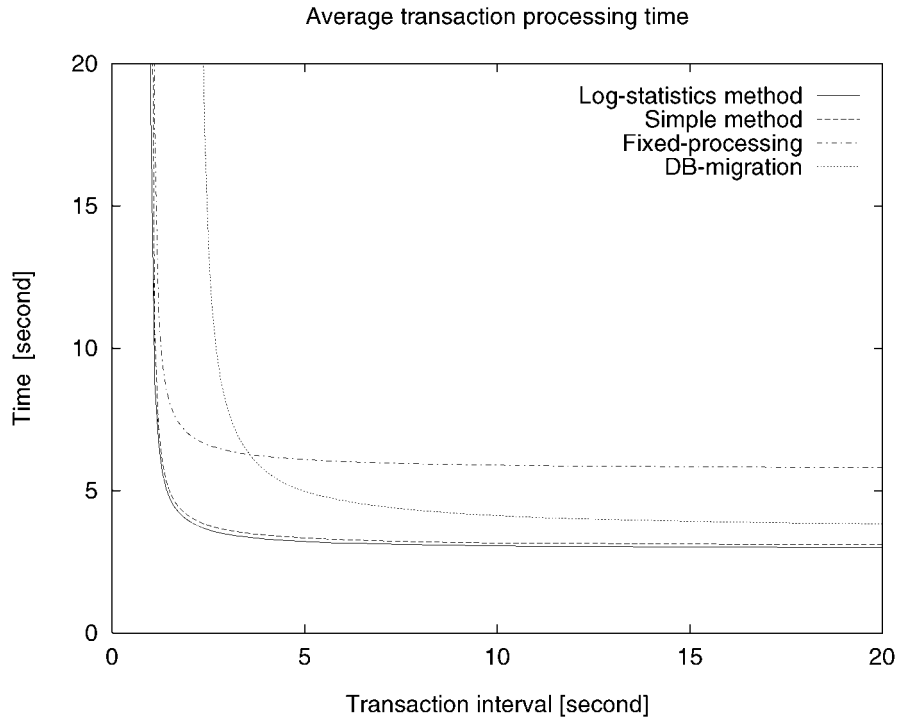


Fig. 9. Relation between the transaction initiation interval and the transaction processing time of the four methods. (Note: bandwidth = 2.0 [Gbps])

7.2 Heterogeneity

For communication in a heterogeneous computer environment, there are several unified binary representations of structured data, such as ASN.1 (Abstract Syntax Notation One) which is specified by International Standardization Organization (ISO), XDR (eXternal Data Representation) which was originally developed for RPC (Remote Procedure Call), and SGML (Standard Generalized Markup Language) for structured documents.

To meet the recent demand for developing database applications efficiently in heterogeneous computer environments, we have proposed a database system based on

ASN.1 [7]. By utilizing a database architecture based on the unified data expression, we can support DB-migration in heterogeneous computer environments and thus can easily develop various applications.

7.3 Replication Management

In this paper, a replica of a database in the distributed system is not created because extra overhead is inevitable in order to maintain consistency between the primary database and each replica. This is because the database at a database sender site is deleted after completing DB-migration.

We believe it is important to reduce the number of message transmissions among sites. However, the replication of databases is valid, especially in WANs, in order to improve transaction processing throughput because it reduces the number of message transmissions for database operations. Hence, many transaction processing methods based on database replication have been proposed. Several features of the system determine whether or not it is effective to make a replica.

7.4 Dynamic Container Generation

To achieve more flexible DB-migration, it should be performed before the database receiver site knows the schema of the database. This can be realized by sending the database schema information and the database size (sometimes, the indexes, the methods specified for the database, and so on) in advance of the transmission of the data contents. As soon as the receiver site receives the information, it should dynamically generate the database schema or the memory space for the database.

8 Comparisons with Related Work

There have been several studies which utilize migration of data items in the area of distributed computing. The most common usage of migration technique is for load balancing among distributed file servers by relocating the distributed data [9], [11], [14], [17]. However, since these studies do not assume the use of broadband networks, they estimate data relocation as a great overhead for distributed systems, and therefore data migration is controlled so that it is not frequently executed. On the other hand in our study, we assume broadband networks and consider DB-migration as one of the principal transaction processing methods.

Two studies [1], [8] focus on how to utilize the advantages of broadband networks for transaction processing. In [8], the *datacycle architecture* was proposed. In this architecture, database contents are periodically broadcast through a *fiber optic ring* thereby achieving high-throughput read-only transaction processing. The datacycle architecture is considered to be similar to our DB-migration approach because both approaches forward database contents through the network. However, since a specific site is responsible for processing all write operations, the datacycle architecture is not a typical distributed database architecture.

In the *send-on-demand protocol* [1], some of the data items requested by the transaction are forwarded to the transaction initiation site. The protocol is further enhanced as a hybrid protocol to make use of an advantage of the datacycle architecture in handling the read operation. However, these protocols depend heavily on the datacycle architecture and are designed to suit the ring-formed network topology. Therefore, it is difficult to apply these protocols to other network topologies. Furthermore, in these protocols, since each write transaction is processed by sending data items to the transaction initiation site, the transaction processing throughput deteriorates when the total size of data items becomes large. Thus, these protocols do not work well in environments where complex transactions are initiated.

In contrast to the above approaches, we assume every broadband network and do not specify any particular network topology. Moreover, our proposed adaptive method selection achieves high transaction processing throughput in a variety of environments since it can choose either the fixed-processing method or the migration-processing method according to the access pattern of the transactions and the method's estimated communication time.

9 Conclusion

As a result of careful consideration of advanced broadband networks features, we have proposed two new transaction processing methods using DB-migration to achieve high-throughput transaction processing. We also have verified the effectiveness of our proposed methods through simulation experiments. These results show the superiority of our proposed methods:

- the adaptive selection of the transaction processing method results in shorter processing times compared to the conventional fixed-processing method, and
- the mechanism for concurrency control for DB-migration gives almost the same performance as concurrency control for the fixed-processing method.

We are now implementing our proposed methods on a practical platform in order to evaluate performance with respect to protocol overhead and other characteristics aside from throughput. The results of this evaluation will be useful when considering further improvements to our proposed methods.

As part of our future work, we will examine lock and migration granularity to determine the optimal settings for a given environment. In our discussion, it is not clear when the overhead of location management for fragments, and message transmissions for database operation and constraint checking, exceeds the overhead of sending unnecessary data in our approach. It is also not clear when the overhead of managing lock request queues exceeds the overhead of locking unnecessary data.

Moreover, we will investigate the use of DB-migration in parallel execution of database operations. In this paper, our aim was to improve the processing time of transactions, which mainly consist of sequential operations, by executing them centrally at the transaction initiation site. However, there appear to be many cases where operations can be executed in parallel and each operation takes a large amount of processing time which is not negligible compared to the communication time. In such a case, an appropriate use of DB-migration for parallel execution improves the transaction processing time.

ACKNOWLEDGMENTS

This research was supported by the Research for the Future Program of the Japan Society for the Promotion of Science, under the project entitled "Integrated Network Architecture for Advanced Multimedia Application Systems, " and by Grant-in-Aids for Scientific Research No. 09780380 and No. 10780260 from the Ministry of Education, Science, Sports and Culture of Japan.

REFERENCES

[1] S. Banerjee, V.O.K. Li, and C. Wang, "Distributed Database Systems in High-Speed Wide-Area Networks," *IEEE J. Selected Areas in Comm.*, vol. 11, no. 4, pp. 617-630, May 1993.

[2] D. DeWitt, R. Katz, F. Olken, L. Shapiro, M. Stonebraker, and D. Wood, "Implementation Techniques for Main Memory Database Systems," *Proc. ACM SIGMOD*, pp. 1-8, June 1984.

[3] H. Garcia-Molina, R.J. Lipton, and J. Valdes, "A Massive Memory Machine," *IEEE Trans. Computers*, vol. 33, pp. 391-399, May 1984.

[4] T. Hara, K. Harumoto, M. Tsukamoto, and S. Nishio, "DB-MAN: A Distributed Database System Based on Database Migration in ATM Networks," *Proc. 14th Int'l Conf. Data Eng.*, pp. 522-531, Feb. 1998.

[5] T. Hara, K. Harumoto, M. Tsukamoto, and S. Nishio, "Location Management Methods of Migratory Data Resources in ATM Networks," *Proc. ACM SAC, Symp. Applied Computing*, pp. 123-130, Feb. 1997.

[6] T. Hara, K. Harumoto, M. Tsukamoto, and S. Nishio, "Main Memory Database for Supporting Database Migration," *Proc. IEEE PacRim, Pacific Rim Conf. on Comm., Computers, and Signal Processing*, vol. 1, pp. 231-234, Aug. 1997.

[7] K. Harumoto, M. Tsukamoto, and S. Nishio, "A Database System Based on ASN.1: Its System Architecture and Database Programming Language," *Proc. ADTI, Int'l Symp. Advanced Database Technologies and Their Integration*, pp. 160-167, 1994.

[8] G. Herman, G. Gopal, K. Lee, and A. Weinrib, "The Datacycle Architecture for Very High Throughput Database Systems," *Proc. ACM SIGMOD*, pp. 97-103, 1987.

[9] T. Johnson and P. Krishna, "Lazy Updates for Distributed Search Structure," *Proc. ACM SIGMOD*, pp. 337-346, 1993.

[10] R. Kadobayashi and M. Tsukamoto, "Traffic-Based Performance Comparison of Mobile Support Strategies," ACM-Baltzer Mobile Networks and Nomadic Applications (NOMAD): *Topical J. Mobility of Systems, Users, Data, and Computing*, vol. 1, no.1, pp. 57-65, 1996.

[11] W. Litwin, M.A. Neimat, and D.A. Schneider, "LH*—Linear Hashing for Distributed Files," *Proc. ACM SIGMOD*, pp. 327-336, May 1993.

[12] J.E.B. Moss, *Nested Trans.: An Approach to Reliable Distributed Computing*. Cambridge, Mass.: MIT Press, 1985.

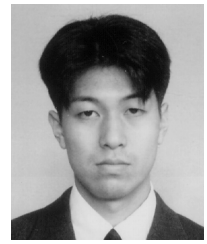
[13] S. Nishio and M. Tsukamoto, "Towards New Multimedia Information Bases in Broadband Networks" (in Japanese), *Trans. Inst. of Electronics, Information, and Comm. Engineers*, vol. 79-D-II, no.4, pp. 460-467, Apr. 1996.

[14] C. Severance, S. Pramanik, and P. Wolberg, "Distributed Linear Hashing and Parallel Projection in Main Memory Databases," *Proc. VLDB 16: Int'l Conf. Very Large Data Bases*, pp. 674-682, 1990.

[15] M. Stonebraker, P.M. Aoki, R. Devine, W. Litwin, and M. Olson, "Mariposa: A New Architecture for Distributed Data," *Proc. ICDE 10: Int'l Conf. Data Eng.*, pp. 54-65, 1994.

[16] M. Tsukamoto, R. Kadobayashi, and S. Nishio, "Strategies for Query Processing in Mobile Computing," T. Imielinski and H.F. Korth, eds., *Mobile Computing*, pp. 595-620, Kluwer, 1996.

[17] R. Vingralek, Y. Breitbart, and G. Weikum, "Distributed File Organization with Scalable Cost/Performance," *Proc. ACM SIGMOD*, pp. 253-264, 1994.



Takahiro Hara received his BE and ME degrees in information systems engineering from Osaka University in Japan in 1995 and 1997, respectively. Currently, he is a research associate in the Department of Information Systems Engineering at Osaka University. His research interests include distributed database systems in advanced computer networks such as high-speed ATM networks and mobile computing environments. He is a member of the IEEE.

Kaname Harumoto received his BE, ME, and DrE degrees in computer science from Osaka University in Japan in 1992, 1994, and 1998, respectively. Currently, he is a research associate in the Department of Information Systems Engineering of Osaka University. His research interests include technologies for database management systems, especially in combination with modern net-work technologies such as high-speed ATM networks and mobile computing environments. He is a member of three learned societies, including the IEEE and the IEEE Computer Society.

Masahiko Tsukamoto received his BE, ME, and DrE degrees from Kyoto University in Japan in 1987, 1989, and 1994, respectively. From 1989 to February 1995, he was a research engineer with the Sharp Corporation. He was an assistant professor in the Department of Information Systems Engineering at Osaka University from March 1995 to October 1996, when he became an associate professor in the same department. His current research interests include database systems, knowledge-base systems, and distributed computing systems. He is a member of seven learned societies, including the ACM, the IEEE, and the IEEE Computer Society.

Shojiro Nishio received his BE, ME, and DrE degrees from Kyoto University in Japan in 1975, 1977, and 1980, respectively. From 1980 to 1988, he was with the Department of Applied Mathematics and Physics at Kyoto University. In October 1988, he joined the faculty of the Department of Information and Computer Sciences at Osaka University in Japan. Since August 1992, he has been a full professor in the Department of Information Systems Engineering of Osaka University. His current research interests include database systems, knowledge-based systems, and distributed computing systems. He has served on the Editorial Board of *IEEE Transactions on Knowledge and Data Engineering*, and is currently associated with the Editorial Boards of *Data and Knowledge Engineering*, *New Generation Computing*, the *International Journal of Information Technology*, *Data Mining and Knowledge Discovery*, and the *VLDB Journal*. He is a member of eight learned societies, including the ACM, the IEEE, and the IEEE Computer Society.