Design and Implementation of DB-MANα: Does Database Migration Work Well in a Real Environment?

Toyokazu Akiyama†, Shinobu Sakai††, Akimichi Umigai‡, Takahiro Hara‡‡, Masahiko Tsukamoto‡‡, Shojiro Nishio‡‡

†Applied Information Systems Division, Cybermedia Center, Osaka University,
5-1 Mihogaoka, Ibaraki, Osaka 567-0047, Japan
E-mail: akiyama@cmc.osaka-u.ac.jp

††Canon Inc., Japan
E-mail: sakai.shinobu@canon.co.jp

‡Toyota Motor Corporation, Japan

‡‡Dept. of Information Systems Eng., Graduate School of Engineering, Osaka University
2-1 Yamadaoka, Suita, Osaka 565-0871, Japan
E-mail: {hara,tuka,nishio}@ise.eng.osaka-u.ac.jp

Abstract

Due to the recent expansion of network bandwidth, the data propagation delay has become a significant factor which influences the system performance in place of the data transmission delay. Based on this fact, we have proposed a new technology to reduce the bad influence of propagation delay on the performance of distributed database system by relocating dynamically the database through networks, which we call *database migration*. Furthermore, we have proposed a database relocation method to choose the transaction processing method between the conventional database fixed method and the proposed database migration method by giving consideration to the transaction complexity. In this paper, we explain our distributed database system with database migration mechanism based on these proposals, and the implementation of the DB–MANα system as a prototype system. The DB–MANα system reduces the database migration time by using a main memory database technique, which induces us to add a backup management mechanism for migratory databases. Moreover, we show some measurement results for the performance evaluation of the DB–MANα system. The results show that the implemented system works well in a practical environment.

Keywords: database migration, distributed database system, transaction processing

1 Introduction

Due to the recent development of network technologies including fiber optic cables and switching technologies such as ATM (Asynchronous Transfer Mode), very high-speed data transmission in the order of gigabits/second is becoming available. The revolution in broadband networks affects the design of database management systems. The most important issue in improving the performance of a network-wide database system is how to use the network bandwidth effectively[Banerjee et al., 1993, Banerjee et al., 1998, Nishio et al., 1996], and this is contrary to the conventional systems where the minimization of data volume transmitted in (narrowband) networks had been considered as the primary factor in performance improvement.

Here, the question is how we can make efficient use of the bandwidth available in broadband networks. A feasible answer is to make databases migrate from one site to another site through networks, which we call *database migration* (*DB-migration* for short).

In the conventional distributed database environment, each database is fixed at a particular site and a typical database operation is performed through several *operation request messages*. Let us refer to such a fixed-database method as the *fixed-processing method*. On the other hand, if we use DB-migration, the number of message transmissions is reduced since the transaction initiation site can have the necessary remote databases migrate to the site. Let us refer to such a DB-migration method as the *migration-processing method*. This means that the transaction processing time can be much shortened because the propagation delay in a broadband network is almost similar to the conventional (narrowband) networks while the transmission delay is very small even for transmitting an entire database. For example, let us consider a transaction which accesses a database at New York with the size of 100 megabytes from Tokyo (Japan), and these two points are connected by fiber optic cables with the bandwidth of one gigabits/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 switch or router. On the other hand, the transmission of the whole database takes 8.0×10^{-1} seconds. Therefore, if the transaction requires more than 7 rounds of communications, it is more efficient to process it locally at Tokyo by using the DB-migration. If we do not neglect the processing delay and the routing delay, this threshold value becomes even smaller.

Based on the idea mentioned above, we have proposed transaction processing methods, which take advantage of DB-migration in broadband networks[Hara et al., 1998b]. Furthermore, we have proposed a distributed database management system, which we call the DB–MAN system, based on these methods[Hara et al., 1998a]. Since the DB–MAN system does not replicate database, there is no need to execute two phase commitment when an update query is initiated at local site, so thus we can save the communication time enough to shorten the transaction processing time. However, in [Hara et al., 1998a], only an outline of the system was given, and there were some unrealistic assumptions for a practical environment, e.g., the access pattern of the future transaction is known. In this paper, we discuss the realization of DB-migration in a practical environment. First, we explain the design and implementation of a distributed database system with a DB-migration mechanism, which we call the DB–MANα system. Then, we show some measurement results for the performance evaluation of the DB–MANα system.

The DB–MANα system chooses the transaction processing method between the fixed-processing method and the migration-processing method based on the method proposed in [Akiyama et al., 1999], since the migration-processing method is not always processed in a shorter period than the fixed-processing method. In the DB–MANα system, each local database is implemented as a main memory database [DeWitt et al., 1984, Garcia-Molina et al., 1984, Severance et al., 1990] to realize high-speed DB-migration. As the price of memory depriciates year by year, it becomes reasonable assumption that each server has enough memory to hold entire database. In general, a main memory database system records backup data to the disk, and when failure occurs, the system recovers the database using the backup data. However, since databases migrate in the DB–MANα system, the conventional backup management strategy could not be directly applied. Therefore, we implemented a new backup management mechanism for DB-migration.

The rest of the paper is organized as follows. The design and implementation of the DB–MANα system are described in section 2 and section 3, respectively. Section 4 describes the measurement results. Finally, we summarize the paper and discuss some future works in section 5.

2 System Design: Problems and Solutions

In this section, we describe our design of the DB–MANα system.

The DB–MANα system chooses the proper processing method at each transaction, and this is a new mechanism that the conventional systems do not have. In order to achieve this mechanism, various system parameters should be considered, such as network bandwidth, database size, and transaction access pattern. In subsection 2.1, we describe the mechanism to choose the processing method in the DB–MANα system.

In the case that a lot of transactions are initiated concurrently during DB-migration or the size of database migrating to the other site is very large, it takes long period to finish the migration. So thus concurrency control mechanism during DB-migration is required in order to keep the concurrent access performance.

The DB–MANα system uses a main memory database technique in order to realize high-speed DB-migration. Since databases migrate, the DB–MANα system requires a new mechanism which is different from that in the conventional main memory database systems. In subsection 2.2, we describe the backup management mechanism in the DB–MANα system.

After describing the above two mechanisms, in subsection 2.3, we describe the system structure of the DB–MANα system.

2.1 Processing method selection

The DB–MANα system chooses the transaction processing method based on the strategy proposed in [Akiyama et al., 1999]. In this subsection, we describe the method selection strategy.

It is considered that DB-migration becomes more effective when the database is successively accessed from the same site. For example, this can happen in an enterprise with many branches over the world. Since office databases are mainly used in daytime, if there are time differences between participant sites of the system, successively accessing site changes in turn. As another example, when a particular branch processes some statistics of sales data which is managed at several other branches, the databases stored in branch offices are effectively processed using DB-migration. In order to detect such a kind of access skew, the DB–MANα system records *access information* for each database at the site where the database resides. The attributes of the access information are as follows:

- S: the site that initiates the most recent transaction.
- P_A: the total number of pages accessed by the successive transactions.
- Q: the total number of queries included in the successive transactions.

At the end of the currently processed transaction, the information of the current transaction is compared with the access information which is recorded for each database involved in the current transaction. If S in the access information is equal to the transaction initiation site, the number of pages accessed by the current transaction is added to P_A and the number of queries included in the transaction is added to

Q. If S in the access information is not the transaction initiation site, the old access information is replaced with the information of the current transaction as new access information.

The method selector of DB–MANα decides whether DB-migration is performed or not at the beginning of a transaction. The decision of DB-migration is made in the following two rules. Here, these rules are applied to each database which is involved in the current transaction and does not reside at the transaction initiation site.

(1) In the case that the current transaction is initiated from the same site which is recorded in the access information as S, we estimate the communication time of the previous successive transactions from S. The estimation is done both if they are processed using the fixed-processing method (T_{fix}) and the migration-processing method (T_{DB}), respectively.

Here, T_{fix} and T_{DB} are estimated as in the following equations:

$$T_{fix} = P_A \cdot D_T + 2Q \cdot D_P$$

$$T_{DB} = P_{DB} \cdot D_T + 3D_P$$

P_A : the number of pages accessed by the successive transactions (the attribute in the access information).

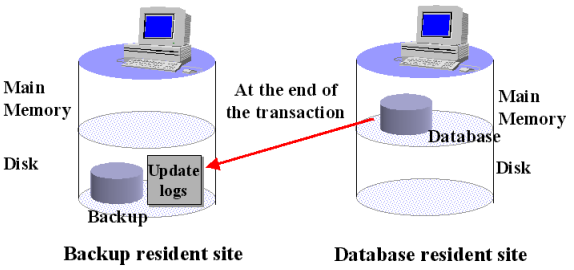
P_{DB} : the number of pages composing the target database.

D_T : the delay for sending a page out to the network (transmission delay).

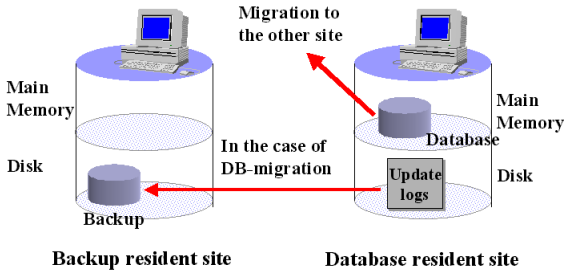
D_P : the delay for transferring data to the remote site (propagation delay).

The first term of T_{fix} formulates the transmission delay of request/reply messages for processing the queries, and the second term formulates the propagation delay of the messages. The first term and second term of T_{DB} formulate the transmission delay and the propagation delay of the migration-processing method, respectively. Here, the migration-processing method includes three data transmissions; (i) the request message transmission, (ii) the database transmission, and (iii) the completion notification message transmission. The second term is dominant in T_{fix}, and the first term is dominant in T_{DB}.

(2) If T_{fix} is larger than T_{DB}, the target database migrates to the transaction initiation site. This indicates that when the total communication time of the previous successive transactions which were initiated from the same site becomes shorter by processing them using the migration-processing method, it is concluded that there exists an access skew and we assume further transactions will be initiated from the same site. Thus, the migration of the database is decided to be performed.



a. Backup resident site



b. Database resident site

Figure 1: Location of the update log.

2.2 Backup Management

In this section, we describe the backup management strategy used in the DB–MANα system.

First, the backup for each database is maintained at a particular site, which we call the *backup resident site*. The site which creates a database is regarded as the backup resident site of the database. When a failure occurs at the site where a database resides (which we call the *database resident site*) and the recovery process is required, the backup resident site recovers the latest database from the backup recorded at the disk. This process requires that the backup is up-to-date at any time. However, propagating every database update to the backup causes many message transmissions and disk I/Os.

Therefore, an update log of the database are created when a transaction is committed and they are written to the disk. When a transaction is aborted, the log is not written to the disk. The latest database can be recovered from the update logs and the backup of the database. (This backup is not up-to-date because the update operations corresponding to the update logs have not been reflected.)

The update logs can be recorded at whether the backup resident site or the database resident site where the corresponding update operations are executed. In the case that the update logs are recorded at the backup resident site as shown in Figure 1a, the update logs are transferred to the backup resident site at the end of the transaction which includes update operations. In the case that the update logs are recorded at the database resident site as shown in Figure 1b, the update logs are written into the disk at the database resident site, and when the database migrates to the other site, the update logs are transferred to the backup resident site.

In the former case, since there is no need to trans-

fer the update logs to the backup resident site during DB-migration, DB-migration requires shorter time. However, it takes long time to commit a transaction because the database resident site has to communicate with the backup resident site. On the other hand, in the latter case, although transaction commitment requires shorter time, it takes long time for DB-migration. In the DB–MANα system, the system administrator chooses one of the above two cases in advance based on the system characteristics.

2.3 System Structure

The system structure of the DB–MANα system is shown in Figure 2. In this subsection, we describe the outline of each module in the system. The modules, except for the *method selector* and the *backup manager*, are similar to those of the conventional distributed database system.

Interface module: When this module receives a transaction (a set of queries) from clients, the received transaction is forwarded to the processing method selector. On the other hand, when this module receives a subquery or a DB-migration request from the system at the other site, it is forwarded to the transaction processor.

Processing method selector: This module consists of the following three submodules.

- *Parser*: parses a query received from the interface module, and then constructs a *parse tree* and sends it to the optimizer.
- *Optimizer*: translates a parse tree into a *plan tree* which consists of primitive database operations, and sends it to the transaction processor. It also sends the information about the transaction to the method selector.
- *Method selector*: decides that the received transaction is processed with either the fixed-processing method or the migration-processing method.

Transaction processor: This module consists of the following three submodules.

- *Transaction executor*: processes the received plan tree based on the processing method selected by the method selector.
- *System command executor*: executes a *system command* which is received from the transaction executor or the system command executor at the local/remote site. The system command include requests for DB-migration, execution of subquery and update of *catalog* information.
- *Backup manager*: creates *update logs* based on the information received from the transaction executor, and then records them into the disk. When system failure occurs, it transmits the latest backup to the site at which the failure occurred.

Storage manager: receives an I/O request to the storage from the transaction processor, and executes the I/O operation.

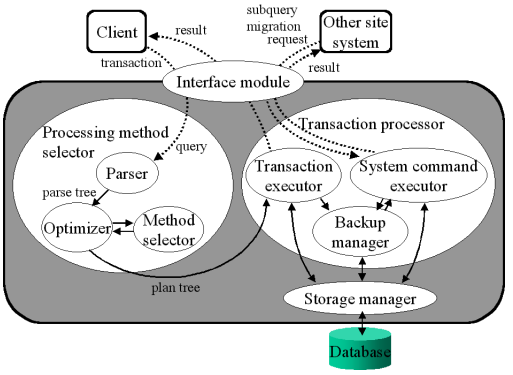


Figure 2: System structure of DB–MANα.

3 System Implementation

In this section, we describe the implementation of the DB–MANα system.

The system is implemented by modifying the source codes of the POSTGRES [Stonebraker et al., 1986, Stonebraker et al., 1990], which is an academic free relational database system developed at University of California at Berkeley (UCB). The functions we added to POSTGRES are classified into the following three categories:

- System Distribution
  - Communication management function among sites
  - Function for dividing query into subqueries
  - Two phase commitment function
- Implementation of the DB-migration facility
  - Database location management function
  - Processing method selection function
  - *Index migration* function
  - *Extended log management* function
- Implementation of the main memory database facility
  - Main memory management function (Extension of the storage management function of the POSTGRES)
  - Backup management function

The system is implemented using C language, and 30,000 lines of source codes are appended and modified to the source code of POSTGRES in order to achieve the above functions. In order to implement the functions of the system distribution, we also partially use source codes of Mariposa[Stonebraker et al., 1994, Stonebraker et al., 1996], which is a distributed database system developed at UCB. Since the POSTGRES is a relational database system, the unit of DB-migration is implemented as a relation.

In the rest of this section, we explain the detail of the two functions, the index migration and the extended log management, which are implemented to improve the performance of the system after performing DB-migration.

3.1 Index migration

In order to shorten the response time of search operations after performing DB-migration, in the DB–MANα system an index attached for a relation migrates together with the relation. In the POSTGRES, an index is implemented as a relation, and in each index, page number and offset in the page are used for tuple identification. Therefore, after performing DB-migration (and index migration), the index can be directly used at the receiver site without converting the index data.

To use an index for retrieval, every site must know the existence of the index, and thus, the information of the index must be registered to the system catalog. In the DB–MANα system, when an index is created, the information of the index is broadcasted to all sites and registered at every site. Since each site does not have to ask the existence of the index, the number of message transmissions for data retrieval can be reduced.

3.2 Extended log management

In the POSTGRES, only transaction execution status, 'commit' or 'abort', is recorded as the log information. Each tuple in a relation maintains the information of two transaction identifiers (IDs) which insert and delete the tuple, respectively. If the status of a transaction is 'commit' in the recorded log, the tuples which are inserted by the transaction are available for retrieval. On the contrary, if the status of the transaction is 'abort', the tuples which are inserted by the transaction are unavailable. When a transaction deletes a tuple, the tuple is not physically deleted, and the transaction ID is recorded to the tuple as the transaction which deleted the tuple. When a transaction updates a tuple, the transaction ID is recorded to the tuple as the transaction which deleted the tuple, then a new tuple is inserted, and the transaction ID is recorded to the new tuple as the transaction which inserted the tuple. If a transaction is rollbacked, the transaction status 'abort' is recorded to the log, and all update operations become void.

When a tuple is retrieved, the system must check an availability of the tuple by referring the log. However, since the original POSTGRES is not a distributed database system, logs are recorded at a particular site where the system runs, and this causes message transmissions when the logs do not reside at the site at which the target relation of the operation resides. Therefore, in the DB–MANα system, logs are maintained for each relation, and the logs migrate together with the relation as the relation migrates.

4 Evaluation

In this section, we present measurement results regarding the performance evaluation of the DB–MANα system. Then we make considerations of the results.

4.1 Evaluation environment

In the experiments, we measure the average response time of the DB–MANα system and compare it with the cases that transactions are processed by only using fixed-processing method and by only using migration-processing method.

The system environment and the parameters in the measurement are shown in Table 1. Several parameters are changed within the ranges of parenthetic values in the corresponding experiments.

The method selector of the DB–MANα system chooses processing method for each database individually and the selection strategy is independent of the number of sites. Thus for simplicity, the number of sites and the number of databases are fixed to 3 and 1, respectively. We use the 100 Mbps Ethernet for communication between each two sites and the effective network bandwidth is about 80 Mbps. In order to emulate various scales of networks in our experiment environment, the propagation delay is systematically generated by the program.

The concurrency control function considering DB-migration, e.g., support for read/write operations during DB-migration, has not been implemented in the DB–MANα system. Therefore, in the experiments, we measure the average response time of the transactions when they are processed sequentially. The transaction initiation interval at each site is calculated using exponential distribution with the mean which is represented by the *average initiation interval* in Table 1. If the previous transaction is not finished (committed) when the transaction initiation time of the next transaction comes, the next transaction must wait for the previous transaction to be committed, and this waiting time is not included in the response time.

The intensive access from a specific site is realized by setting the average initiation interval of the site shorter than the other sites. The average initiation interval of the intensive access is called the *average intensive initiation interval* and another is called the *average sporadic initiation interval*. The site which issues intensive access is randomly determined and periodically changed based on the *period to change intensive access site*. The *transaction initiation interval ratio* is the ratio of the average sporadic initiation interval to the average intensive initiation interval, and this shows the degree of the access intensity.

The following notations are used in the graphs of the measurement results.

DB–MANα: The average response time of the DB–MANα system, i.e., the transaction processing method is adaptively chosen by the method selector.

MIG: The average response time where all transactions are processed using the migration-processing method. At the beginning of a transaction, all databases involved in the transaction migrate to the transaction initiation site.

FIX: The average response time where all transactions are processed using the conventional fixed-processing method.

4.2 Experiment 1: Database size

The measurement results varying the database size are shown in Figure 3.

Table 1: System environment and parameter configuration.

Network environment	
number of sites	3
network bandwidth	80 [Mbps]
propagation delay	200 [msec] (50∼400 [msec])
Database	
number of databases	1
database size	31.5 [MB] (3.5∼73.5 [MB])
Transaction	
number of queries in a transaction	10 (1∼30)
access data size	1000 [tuple] (600∼15000 [tuple])
period to change the intensive access site	400 [sec] (30∼1600 [sec])
transaction initiation interval ratio	5 (1∼20)
average intensive initiation interval	30 [sec]
average sporadic initiation interval	150 [sec] (30∼600 [sec])

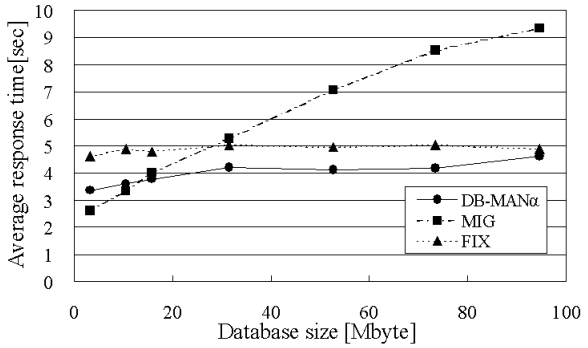


Figure 3: The relation between the database size and the average response time.

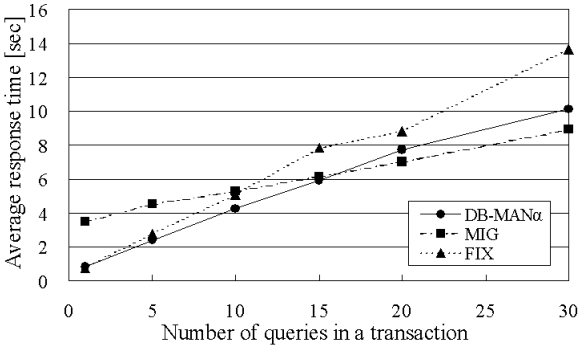


Figure 4: The relation between the number of queries in a transaction and the average response time.

When the database size is small, the time for performing DB-migration becomes shorter, and thus the migration-processing method, which requires smaller number of message exchanges, shows better results than the fixed-processing method. In this case, since the DB–MANα system nearly always chooses the migration-processing method, DB–MANα and MIG give almost the same performance. When the database size is very small, the average response time of DB–MANα is slightly larger than that of MIG. This is because the migration-processing method is chosen in the DB–MANα system only when the successive transactions are initiated at the same site, and thus the fixed-processing method is always chosen for processing the first transaction after the transaction initiation site changes.

When the database size is large, it takes long time for performing DB-migration, and thus the migration-processing method shows better results than the fixed-processing method. In this case, since the DB–MANα system nearly always chooses the fixed-processing method, DB–MANα and FIX give almost the same performance.

In this experiment, the total size of data accessed by a query, *access data size*, is fixed. Therefore, the response time of FIX is scarcely affected by the database size. The DB–MANα system shows good results in most cases, because it chooses the processing method considering the database size.

4.3 Experiment 2: Number of queries in a transaction

The measurement results varying the number of queries in a transaction are shown in Figure 4.

The number of queries in a transaction represents the increment of attribute Q in the access information. When it is set to the small value, the system rarely detects the access skew. Therefore, the DB–MANα system nearly always chooses the fixed-processing method, and thus it shows almost the same performance to FIX. When the number of queries in a transaction is very small, the average response time of DB–MANα is slightly larger than that of FIX. This is because extreme successive transactions which are incidently initiated cause an inefficient choice of the migration-processing method and this degrades the performance.

When the number of queries in a transaction is large, the number of message exchanges in the fixed-processing method becomes larger, and thus migration-processing method shows better results. In this case, since the DB–MANα system nearly always chooses the migration-processing method, DB–MANα and MIG give almost the same performance. When the number of queries in a transaction is very large, even if transactions are sporadically initiated, the DB–MANα system considers that there exists an access skew and execute an inefficient DB-migration, and this degrades the performance. However, in real

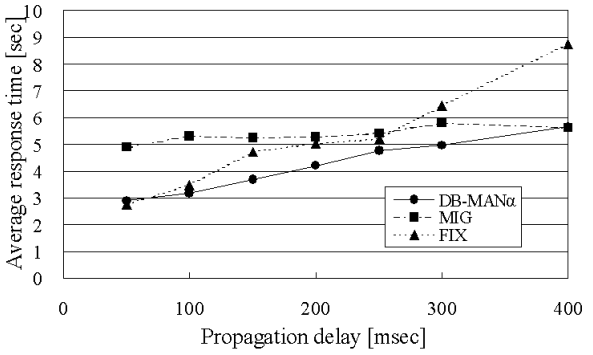


Figure 5: The relation between the propagation delay and the average response time

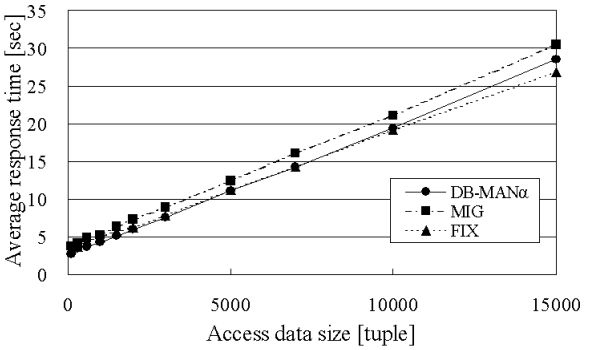


Figure 6: The relation between the access data size and the response time.

environments, a transaction which includes a large number of queries is usually initiated as a batch processing from the specific application. Therefore, we can improve the performance of the DB–MANα system in such cases by implementing the function to distinguish the client application.

4.4 Experiment 3: Propagation delay

The measurement results varying the propagation delay are shown in Figure 5.

When the propagation delay is small, the message exchanges do not take long time, and thus the fixed-processing method shows better results than the migration-processing method. In this case, since the DB–MANα system nearly always chooses the migration-processing method, DB–MANα and FIX give almost the same performance. When the propagation delay is very small, similar to the experiment 2, the average response time of DB–MANα is slightly larger than that of FIX. This is because extreme successive transactions which are incidently initiated cause an inefficient choice of the migration-processing method and this degrades the performance.

When the propagation delay is large, the number of messages exchanged heavily affects the response time in the fixed-processing method, and thus the migration-processing method shows better results than the fixed-processing method. In this case, since the DB–MANα system nearly always chooses the migration-processing method, DB–MANα and MIG give almost the same performance. When the propagation delay is very large, the average response time of DB–MANα is slightly larger than MIG. This is because the DB–MANα system always chooses the fixed-processing method for the first transaction after the transaction initiation site changes.

The DB–MANα system shows better results in most cases, because it chooses the proper processing-method considering the propagation delay.

4.5 Experiment 4: Access data size

The measurement results varying the access data size are shown in Figure 6.

When the access data size is small, it takes short time to transfer the query results in the fixed-processing method, and thus the fixed-processing method shows better results than the migration-processing method. In this case, the DB–MANα system nearly always chooses the fixed-processing method. Since, the difference in processing time between the migration-processing method and the fixed-processing method is small, the effect of performing an inefficient DB-migration is also small in the DB–MANα system. Therefore, DB–MANα shows the best performance in the case that the access data size is small.

As the access data size gets larger, though the size of data transferred as the query results get larger in the fixed-processing method, the average response time of this method gets shorter than the migration-processing method. This is because the local operations at each site and the data transmissions between the transaction initiation site and the database resident site can be executed in parallel in the fixed-processing method. To put it more concretely, the transaction initiation site can process the results of subqueries received from the remote sites while the remote sites executes the rest of subqueries. In the same way, the local operations at the transaction initiation site and the transmissions of results of subqueries from the remote site can be also performed in parallel. For these reasons, FIX shows better results than MIG. However, the method selector of the DB–MANα system does not take such parallel executions of local operations and message transmissions into account for estimating the processing time of the fixed-processing method. It is very difficult to precisely estimate it since such parallel executions are asynchronously performed and the processing time is affected by the system condition. When the access data size is large, the DB–MANα system tends to choose the migration-processing method, and however, it is not good choice in most cases. Therefore, when the access data size is very large, DB–MANα gives worse performance than FIX. Some extensions should be done to the method selector of the DB–MANα system in order to deal with such asynchronous executions.

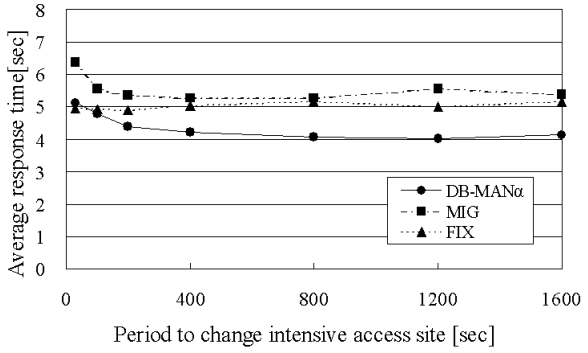


Figure 7: The relation between the period to change intensive access site and the average response time.

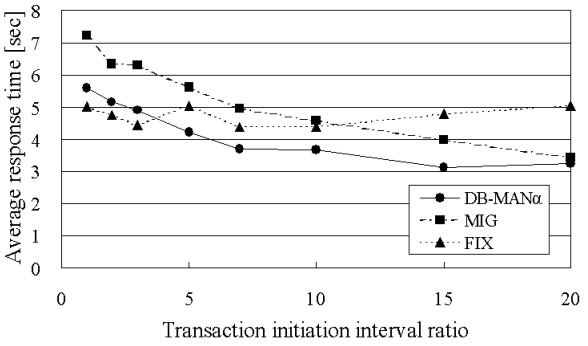


Figure 8: The relation between the transaction initiation interval ratio and the average response time.

4.6 Experiment 5: Period to change intensive access site

The measurement results varying the period to change intensive access site are shown in Figure 7.

When the period to change intensive access site is short, the number of successive transactions from a specific site is small, and thus the fixed-processing method shows better results than the migration-processing method. In this case, since the DB–MANα system nearly always chooses the fixed-processing method, DB–MANα and FIX give almost the same performance. When the period to change intensive access site is very short, similar to the experiments 2 and 3, the average response time of DB–MANα is slightly larger than that of FIX because extreme successive transactions which are incidently initiated cause an inefficient choice of the migration-processing method.

As the period to change intensive access site gets longer, the number of successive transactions from a specific site gets larger, and thus the response time of DB–MANα and MIG gets shorter. However, in MIG, sporadic transactions initiated from sites other than the intensively accessing site cause an inefficient DB-migration and this degrades the performance.

4.7 Experiment 6: ratio of transaction generating interval

The measurement results varying the transaction initiation interval ratio are shown in Figure 8.

When the transaction initiation interval ratio is small, the number of successive transactions from a specific site is small, and thus the fixed-processing method shows better results than the migration-processing method. In this case, since the DB–MANα system nearly always chooses the fixed-processing method, DB–MANα and FIX give almost the same performance. When the ratio is very small, similar to the experiments 2 and 3, the average response time of DB–MANα is slightly larger than that of FIX because extreme successive transactions which are incidently initiated cause an inefficient choice of the migration-processing method.

When the transaction initiation interval ratio is large, the migration-processing method shows better results than fixed-processing method. As the transaction initiation interval ratio gets longer, the number of successive transactions from a specific site gets larger, and the response time of DB–MANα and MIG gets shorter.

The DB–MANα system gives the best performance in most cases.

5 Conclusion

In this paper, we have designed the distributed database system based on database migration, and implemented the DB–MANα system as a prototype. The DB–MANα system has two notable functions, the method selector and the backup manager. The former properly chooses the transaction processing method and the latter manages database backups considering DB-migration. We have also implemented the index migration and the extended log management functions to improve the performance of the system after performing DB-migration.

We have measured the performance of the DB–MANα system, and verified the effectiveness of the method selector in the system. These results shows that the DB–MANα system can improve the system performance comparing with the conventional distributed database system. Especially, when successive transactions are initiated at a specific site, the DB–MANα system drastically improves the performance.

As parts of our future work, we are planning to implement the concurrency control function considering DB-migration, e.g., support for read/write operations during DB-migration. We are also planning to extend the DB–MAN system to deal with database replicas[Krishnakumar et al., 1992, Stonebraker, 1979, Thomas, 1979] in order to improve the system performance when concurrent accesses frequently occur.

6 Acknowledgment

This research was supported in part by Special Coordination Funds for promoting Science and Technology of the Ministry of Education, Culture, Sports, Science and Technology of Japan, by Grant-in-Aid

for Scientific Research on Priority Areas (C) numbered 13224064 from the Ministry of Education, Culture, Sports, Science and Technology of Japan, and by Research for the Future Program of Japan Society for the Promotion of Science under the Project "Advanced Multimedia Content Processing (JSPS-RFTF97P00501)".

References

[Akiyama et al., 1999] Akiyama, A., Hara, T., Harumoto, K., Tsukamoto, M. and Nishio, S. (1999). Database Migration Based on Access Skew Detection. *Proc. of 1999 Int'l Symp. on Database, Web and Cooperative Systems*, 1:65–72.

[Banerjee et al., 1993] Banerjee, S., Li, V.O.K., and Wang, C. (1993). Distributed Database Systems in High-Speed Wide-Area Networks. *IEEE J. Selected Areas in Comm.*, 11(4):617–630.

[Banerjee et al., 1998] Banerjee, S. and Chrysanthis, P.K. (1998). Network latency optimizations in distributed database system. *Proc. of 14th International Conference on Data Engineering*, 532–540.

[DeWitt et al., 1984] DeWitt, D., Katz, R., Olken, F., Shapiro, L., Stonebraker, M. and Wood, D. (1984). Implementation Techniques for Main Memory Database Systems. *Proceedings of ACM SIGMOD'84*, 1–8.

[Garcia-Molina et al., 1984] Garcia-Molina, H., Lipton, R.J. and Valdes, J. (1984). A Massive Memory Machine. *IEEE Transaction on Computer*, l(C-33):391–399.

[Hara et al., 1998a] Hara, T., Harumoto, K., Tsukamoto, M. and Nishio, S. (1998). DBMAN: A Distributed Database System Based on Database Migration in ATM Networks. *Proceedings of 14th International Conference on Data Engineering*, 522–531.

[Hara et al., 1998b] Hara, T., Harumoto, K., Tsukamoto, M. and Nishio, S. (1998). Database Migration: A New Architecture for Transaction Processing in Broadband Networks. *IEEE Transaction on Knowledge and Data Engineering*, 10(5):839–854.

[Krishnakumar et al., 1992] Krishnakumar, N. and Bernstein, A.J. (1992). High performance escrow algorithms for replicated databases. *Proceedings of 18th VLDB Conference*, 175–186.

[Severance et al., 1990] Severance, C., Pramanik, S. and Wolberg, P. (1990). Distributed Linear Hashing and Parallel Projection in Main Memory Databases. *Proceedings of the 16th VLDB Conference*, 674–682.

[Nishio et al., 1996] Nishio, S. and Tsukamoto, M. (1996). Towards New Multimedia Information Bases in Broadband Networks. *Transaction of the Institute of Electronics, Information and Communication Engineers D-II*, 79-D-II(4):460-467.

[Stonebraker et al., 1986] Stonebraker, M. and Rowe, L.A. (1986). The design of POSTGRES. *Proceedings of ACM SIGMOD'86*, 340–355.

[Stonebraker, 1979] Stonebraker, M. (1979). Concurrency control and consistency in multiple copies of data in distributed INGRES. *IEEE Transaction on Software Engineering*, 3(3):188–194.

[Stonebraker et al., 1990] Stonebraker, M., Rowe, L.A. and Hirohama, M. (1990). The Implementation of POSTGRES. *IEEE Transaction on Knowledge and Data Engineering*, 2(1):125-141.

[Stonebraker et al., 1994] Stonebraker, M., Aoki, P.M., Devine, R., Litwin, W. and Olson, M. (1994). Mariposa: A New Architecture for Distributed Data. *Proceedings of IEEE Data Engineering*, 54–65.

[Stonebraker et al., 1996] Stonebraker, M., Aoki, P.M., Pfeffer, A., Sah, A., Sidell, J., Staelin, C. and Yu, A. (1996). Mariposa: A Wide-area Distributed Database System. *VLDB Journal*, 5(1):48–63.

[Thomas, 1979] Thomas, R.H. (1979). A majority consensus approach to concurrency control for multiple copy databases. *ACM Transaction on Database Systems*, 4(2):180–209.