オペレーティング・システム 26－1
（１９８５．３．１）

ＲＥＳＱによる分散型データベースの性能評価

A DDB Performance Evaluation by RESQ

山村 吉信
Yoshinobu Yamamura
日本アイ・ビー・エム株式会社　サイエンス・インスティチュート
IBM Japan Science Institute

1.0 INTRODUCTION

Data processing requirements have grown so rapidly that no single computer system can offer efficient data accessibility and thus good overall performance. That is, if one host computer contains all of the enterprise databases, then the computer failure will seriously affect the daily work of the enterprise and no one can have an access to that database until it is recovered. Furthermore the growth rate of data processing requirements has outgrown the capacity of any single computer system.

The above circumstances naturally force most companies to install multiple sets of computer systems which are expected to solve the above problems. Usually these computers are coupled together to provide database sharing capability. Without the aids of distributed database management concept, however, this coupling does not seem to solve data accessibility problem but only helps to deal with increased transaction volumes.

Geographically distributed database systems are classified into various categories from the viewpoint of how database copies are held, and the method of controlling of concurrent updating. For instance, a distributed concurrent system with full replication of the database can ensure high data accessibility easily but the many functions to sustain data integrity in inter-node communication tends to lower performance [THOM79, ROTH80]. While centralized concurrent control system with shared single copy of database and distributed concurrent control system with dynamic replication can guarantee both data integrity and performance well but not do excell in data accessibility [STRI82, DANI83].

2.0 SYSTEM

The domain considered here is one of large commercial database systems such as banking applications since the requirements as described in "Introduction" are the most demanding in this area.

2.1 DESIGN APPROACH

The design requires large number of transaction volumes and the improvement of data accessibility. Also required is to be transparent to existing application programs which run on traditional database management systems; to assure acceptable performance to end users; and to guarantee data integrity. Data integrity, data accessibility, and performance are interdependent characteristics with respect to managing DB.

The key words used in this paper are defined as follows:

Data Integrity	Consistency among data entities in a system
Data Accessibility	Ability to continue processing even if some node fails
Performance	Response time. It is equivalent to the time period during which the terminal is dedicated to the transaction.

The design approach taken here is as follows. First, based on a traditional database management system, data accessibility is added by the use of backup copy at backup node. Second, parallel processing of the response to user and the maintenance of the backup copy is carried out by the use of a cluster controller to assure performance.

2.2 SYSTEM COMPONENTS AND FUNCTIONS

A system consists of host nodes, cluster nodes, and networks. Each host node consists of a Transaction Manager with DBTAB, Data Manager with DB, and Backup DM with Backup DB. An example of a system configuration is illustrated in Figure 1. In this example three host nodes are in the system: Tokyo, Nagoya, and Osaka. Every terminal is connected to one primary host node and one backup host node. The leftmost name in Figure 1 shows the primary host node of that terminal. In each host node there are two types of databases, primary and backup. The Tokyo host node has the Tokyo primary database and the Osaka back up database for Osaka. If one of the host node, say Osaka, goes down the Tokyo host node is then to serve every terminal connected to Osaka host node by use of the Osaka backup database.

2.2.1 Host Node

Each host node consists of Transaction Manager with DBTAB, Data Manager with DB, and Backup DM with Backup DB as illustrated in Figure 1.

TM - Transaction Manager: Transaction Manager Manages all of communication within the host node:

- Manages interface between users and host node
- Manages interface between each node
- Recognizes location of requested DB entity by looking at DBTAB
- Sends entity allocation request to DM/BDM
- Include TP access method such as ACF/VTAM

DBTAB - Data Base Table:

- Entity Key Number
- Node-Id of primary DB
- Node-Id of backup DB
- status such as live or dead

DM - Primary Database Manager

- Manages local primary database

DB - Primary Database

BDM - Backup Database Manager

- Manages local backup database
- Simulates failed DM

BDB - Backup database

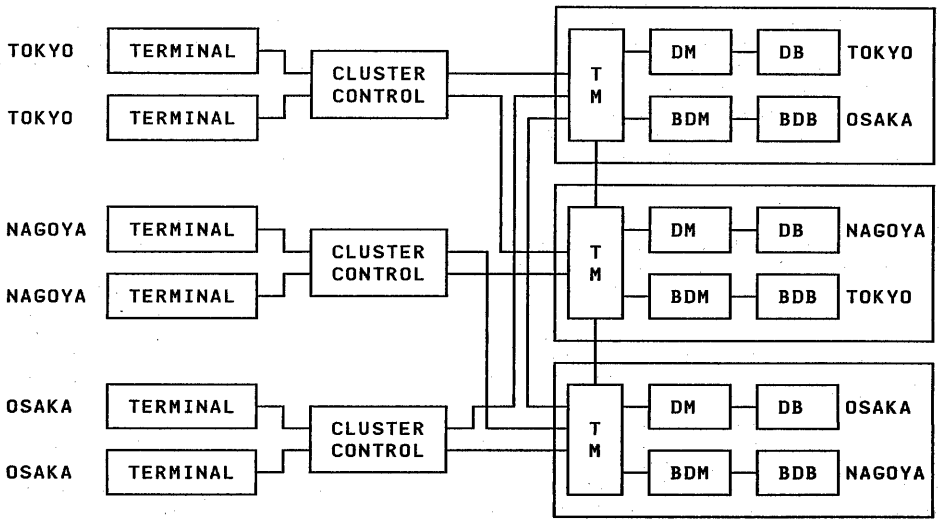

Figure 1. System Configuration: A system consists of host nodes, cluster nodes with several terminals, and communication networks. Each host node consists of Transaction Manager with DBTAB, Data Manager with DB, and Backup DM with Backup DB.

2.2.2 Cluster

The cluster consists of cluster controller and terminals. The cluster controller is a special purpose small processor which contains a hard disk drive and a sufficient processor power. It manages receiving transactions from terminals, communicating with host node, and storing sufficient information of each transaction profile which is necessary for recovery procedure as described in "Recovery".

2.2.3 Network

The network consists of communication controllers, modems, and communication links.

2.3 SYSTEM BEHAVIOR

Take next transaction as an example a system behavior is explained.

2.3.1 Description of Example

- Normal processing mode is assumed.
- The primary database is located at the host node 1 named HOST.
- The backup database is located at the host node 2 named BACKUP.
- Each system component is suffixed to identity the host eg., TM2 means TM at host node 2, HOST.

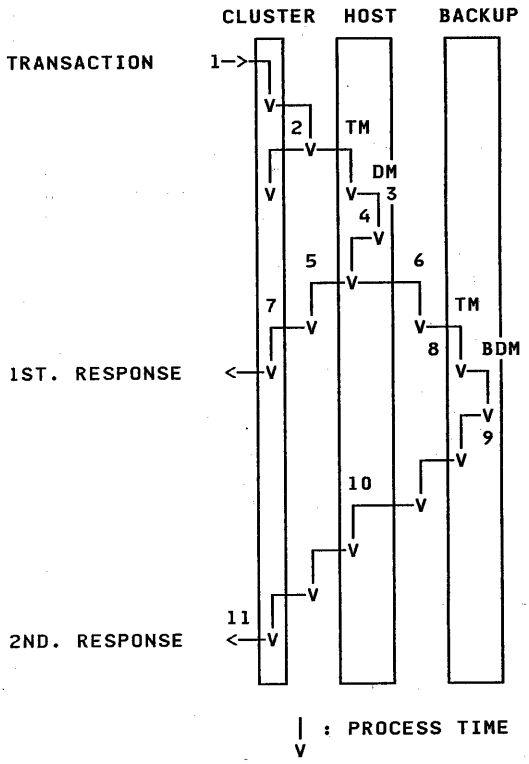

Figure 2. A Transaction Life Example: This is an example of single entity request. A message received node has an acquired production entity.

2.3.2 Behavior

The illustration of this example is in Figure 2, and the description of this example is as follows,

1. A user generates a transaction. The cluster controller records the transaction profile and send the transaction itself to the host node via network.
2. TM1 sends back the acknowledgement message to the cluster,
3. and consult the DBTAB to identify the location of requested database entity. In this example, since DM1 is the owner TM1 sends a processing request to DM1.
4. DM1 allocates all resources to execute and does update processing for the primary database. At the same time it updates the sequence number of the target entity. After processing it releases resources and pass the control to TM1.
5. TM1 sends the first response to the cluster, and
6. consults the DBTAB to identify the location of the corresponding backup database entity. In this example since the host node 2, BACKUP, is the owner TM1 generates a transaction for backup maintenance and sends it to TM2.

7. The cluster controller recognizes that the update completed. It records the status and sends a response to the user. At this time it releases the terminal for that transaction. It means that terminal is ready for the next transaction.
8. When TM2 receives a backup maintenance transaction it sends a processing request to BDM2. DM2 then updates the corresponding backup database entity as indicated update sequence number.
9. After completion of the update BDM2 notifies TM2 and TM2 sends a response to TM1
10. After receiving the response from TM2, TM1 completes the backup maintenance transaction and sends the second response to the cluster.
11. When the cluster receives the second response, ie., the final response, it stores all of status information of the transaction to its hard disk, and completes the transaction.

3.0 RECOVERY

"Two Phase Lock (2PL)" protocol by Eswaran [ESWA76] is a starting point of this discussion. All transactions in this system are supposed to take the 2PL protocol. "Two Phase Commit (2PC)" protocol by Gray [GRAY79] and "Majority Consensus Approach" by Thomas [THOM79] look very elegant. However, communication delay due to voting may become unacceptable in a commercial DB application since the above approaches do not permit any responses to the transaction originator (a user) unless the DBMS decides OK or not. "Fault and atomic actions" by Randell [RAND79] and "Atomic action's unitary property" by Lampson [LAMP83] are good milestones for reliability issues. This paper extends those two by use of Gray's logging architecture [GRAY79].

3.1 THE RECOVERY MODEL

The model simply takes a logging strategy to redo or undo some failed transactions. General terms used in this paper are defined as follows:

node: host nodes and cluster nodes [YAMA83 and YAMA84A]
cluster log: done when a response is sent back to the user [YAMA84A]
information in hands: cluster log, host log, and pending transactions in live nodes [YAMA83 and YAMA84A]
recovery action: REDO and UNDO [GRAY79]

3.2 SINGLE PROCESSOR SINGLE DB (SPSD) RECOVERY

In a traditional SPSD system, the recovery strategy becomes simple [YAMA84B]. The transaction and the host log are defined as:

transaction: a series of actions: READ, WRITE, LOCK, and UNLOCK [LAMP83]
host log: commit point by log-write-ahead [GRAY79]

As a transaction goes from start to end, each status corresponds to this transaction phase as illustrated in Figure 3. In Figure 3, "START" phase transaction, which corresponds a period between generation of transaction in a cluster node and before locking, produces "PENDING" status in the cluster node. "LOCK" or "UNLOCK" phase transaction produces nothing. "UPDATE" phase transaction produces one of the two states: "PRE_COMMIT LOG" in the host log or "POST_COMMIT LOG" in one. "END" phase transaction, which corresponds a period between "UNLOCK" and a response to the user, produces "CLUSTER_LOG" in the cluster log and "RELEASE" the previously set "PENDING" status in the cluster.

The vital information to guarantee consistency in the SPSD environment is matching between user's view and the DB entity. Since our assumption about the available information comprises "CLUSTER", "HOST", and "PENDING" status, we can easily transform the combination of those into the transaction phase, then we can take appropriate recovery action.

TRANS	DB	CLUST	HOST	PEND
START				CLUST
LOCK				
UPDATE			PRE	
	UPDATE			
			POST	
UNLOCK				
END		CL_LOG		REL

Figure 3. Transaction Phases in a SPSD

3.3 MULTIPLE PROCESSOR MULTIPLE DB (MPMD) RECOVERY

The treatment for a MPMD recovery becomes more complicated [YAMA84B]. The definitions of the transaction and the host log are changed as follows:

transaction: parallel actions: READ, WRITE, LOCK, and UNLOCK, defined as in SPSD (primary transaction) Non-primary ones are kicked by the primary one (mirror transactions). [YAMA83 and YAMA84A]

host log: commit point by log-write-ahead in each DB, i.e., primary and backups [YAMA83 and YAMA84A]

By analogy of the SPSD recovery, same kind of transformation can be carried out for MPMD recovery. Important situations related directly to consistency among user's view, primary DB entity, and backup DB entities are summarized in Figure 4. Again, similar discussion can be applied here for the MPMD environment. The important thing is that, by the aid of the available information, we can easily take an appropriate recovery action.

TRANS	CLUSTER	HOST LOG
START LOCK	NONE	NONE
UPDATE OR UNLOCK OR END	NONE	PRIMARY DB LOG & 0 TO N BDB LOG
	CL_LOG	PRIMARY DB LOG & 0 TO N BDB LOG
	CL_LOG	ALL LOG

Figure 4. Transaction Phases in a MPMD

4.0 PERFORMANCE MODEL

The objective of this evaluation is to assess whether this system's performance is acceptable or not. We take response time as acceptability measure because it represents a performance measure from the viewpoint of users. We need to construct a model to reflect the transaction life exactly since we do not know dominant effect of each componet to the response time at this point of time. This enforces us to use a simulation model.

4.1 SIMULATION

The reasons why simulation is used instead of analytic models are as follows,

Fork process at network
Process synchronization
Passive queue such as lock

4.1.1 Fork process at network

A transaction (a job) generates several processes in the system.

4.1.2 Process synchronization

The way of handling a transaction by a DM (Primary Database Manager) is as follows:

1. Schedule an application region for the transaction.
2. Process the transaction in the application region. This includes DBs read activity.
3. Send response to the user.
4. Process the backup activity (logging) in a DM region.
5. Process the OTHREAD activity in another DM region. This is to write the DBs.

Activity 3, 4, and 5 run concurrently, and activity 4 and 5 must be synchronized.

4.1.3 Passive queue such as lock

The number of the backup process is limited since the physical number of logging device is limited. This means that more than one transaction use the same backup process. To ensure mutual exclusion, we have to provide the locking for the backup process.

4.2 RUNNING ENVIRONMENT

We use the Research Queueing Package (RESQ) [SAUE82] as a simulation langauge since RESQ well copes with the above problem. The parameters used in the run are listed in Figure 5.

5.0 RESULT

The result is obtained in Figure 6.

configuration:
3 host nodes
10 comm. ctl. per node
5 comm. links per comm. ctl.
48 K BPS full duplex line
parameters:
10 MIPS per processor
25 mili sec for each DB I/O
5 DB I/Os per each appl. pgm.
25 mili sec for comm. ctl. service time

Figure 5. Modeling Parameters

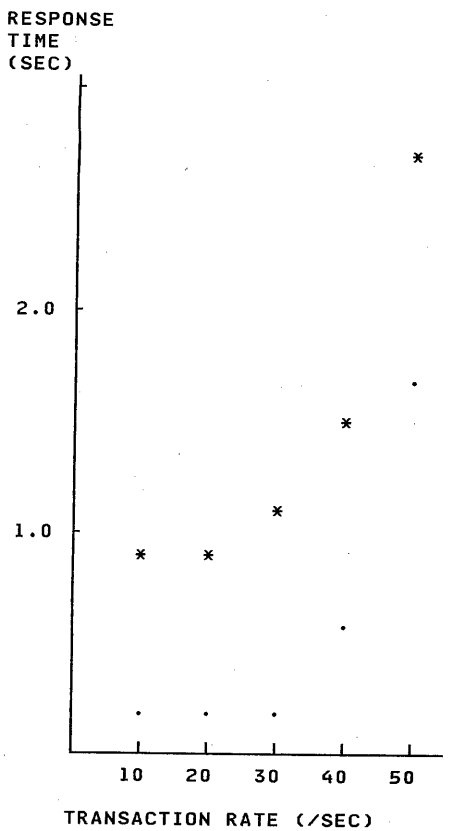

Figure 6. First Response Time VS Final One: '.': First, '*': Final.

6.0 CONCLUSION

Data integrity, data accessibility, and acceptable performance are mandatory for a large commercial database system. In this paper, a system which satisfies the above requirements are proposed.

7.0 REFERENCES

DANI83 Daniel R.C. et al., "Dynamic Replication, an overview", Proceedings of NCC, (1983), 219-227.

ESWA76 Eswaran, K. P., et. al., "The Notions of Consistency and Predicate Locks in a Database System", Comm. ACM 19, 11, (Nov. 1976), 624-633.

GRAY79 Gray, J. N., "Notes on Data Base Operating Systems", Operating Systems - An Advanced Course, Springer, (1979), 393-481.

LAMP83 Lampson, B. W., "Atomic Transactions", Distributed Systems - Architecture and Implementation, Springer, (1983), 246-265.

RAND79 Randell, B., "Reliable Computing Systems", Operating Systems - An Advanced Course, Springer, (1979), 282-391.

ROTH80 Rothnie J.B. et al., "Introduction to a System for Distributed Databases (SDD-1)", ACM TODS, 5, 1, (March 1980), 1-17.

SAUE82 Sauer C.H., et al., "The Research Queueing Package Version 2: CMS Users Guide", IBM Research Center, Research Report, RA-139, April, (1982).

STRI82 Strickland J.P. et al., "IMS/VS: An Evolving System", IBM Systems Journal, 21, 4, (1982), 490-510.

THOM79 Thomas R.H., "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases", ACM TODS, 4, 2, (June 1979), 180-209.

YAMA83 Yamamura Y., "A Parallel Processing in a DDB System", Proceedings of the IPSJ Semi-Annual Conf. 27, (1983), 6K3.

YAMA84A Yamamura Y., "A Multiple Backup Strategy for a DDB Syatem", Proceedings of the IPSJ Semi-Annual Conf. 28, (1984), 6E6.

YAMA84B Yamamura Y., "A Note on Data Integrity for a DDB Syatem", Proceedings of the IPSJ Semi-Annual Conf. 29, (1984), 6F3.