Transaction Management and Database Recovery for WAKASHI

6G－7

Ge YU‡, Guangyi BAI†, Juanfei GAO†, and Akifumi MAKINOUCHI†

† Department of Computer Science and Communication Engineering, Kyushu University

‡ Department of Computer Science and Engineering, Northeastern University, P.R.China

1 Introduction

Transaction management and database recovery have been studied widely since database systems emerged[2]. With the advance of computer technology, the three key factors(*Database Organization, Transaction Structure and Behavior*, and Computing Environment) have been changed a lot so that the existing methods might not be available for next-generation systems such as WAKASHI which has special features like *Distributed Shared Heap, Nested Transaction* and *Multi-Client-Multi-Server Architecture.*

This paper mainly deals with the recovery method for transaction management of WAKASHI. Although the method is studied based on WAKASHI, the results can also be pertinent to other similar systems. The following techniques are adopted for constructing the recovery system. (1) Using a Two-Level recovery scheme for global and local recovery. (2) Using ARIES paradigm[3] for local recovery since its flexibility and effectivity and also making necessary extensions. (3) Using a Termination-Query mechanism for distributed and nested transaction recovery.

2 Nested Transaction of WAKASHI

WAKASHI is an object storage management system for distributed object-oriented DBMSs by exploiting the *virtual-memory mapping technique*[1]. In WAKASHI, objects are created and operated in memory-heaps, which then mapped to the disks of servers.

A nested transaction in WAKASHI is composed of user-defined transactions (called *Client-Transaction(C-T)*), each of which runs as a client. Each C-T is composed of system-defined transactions (Called *Heap-Transaction(H-T)*), each of which only accesses the objects in one heap. A nested *C-T* is executed at local site or remote site while an *H-T* is executed at the local site. Each nested *C-T* is an autonomous transaction, which is committed independently by *Non-Blocking Commitment Protocol.* However, when its parent is aborted, the *C-T* should be compensated. In Fig. 1, the distributed execution of a nested transaction is illustrated.

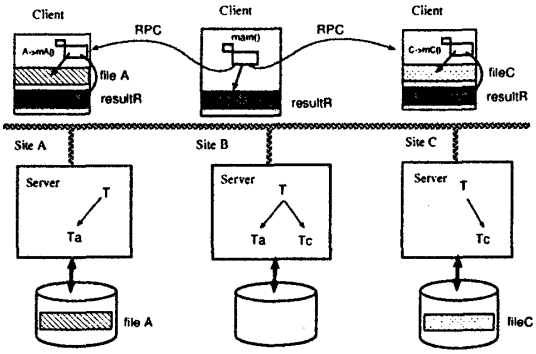

Figure 1: Nested Transaction of WAKASHI

The data in a paged-object server can have three types of copies in *Disk*, *Host Server*(at the same site as Disk), and *User Server*(at the site of page readers and/or page writers). The crash can cause three degrees of data damages: (1) User Heap Loss, (2) Host Heap Loss and (3) Disk Heap Loss. Since the degree 3 can be solved by archival dumping, only degree 1 and 2 are handled in this paper. On the other hand, the crash can cause two types of transaction failure: *Child failure* and *Parent failure.* In the last case, the children without parent become *orphans.*

3 Recovery System and Log

The recovery system is organized in two-levels. At each site, there are two recovery sub-systems: *Transaction-level Recovery Manager(T-RM)* and *Data-level Recovery Manager(D-RM).* Correspondently, there are two Log files: (1) *T-log* to keep the transaction status, the transaction dependency relationship, the distributed information, and pointer to its D-log. (2) *D-log* to keep the operations on databases. Fig. 2 shows an example of the log structures, in which the horizontal arrows express the data operation semantics and the vertical arrows express the dependency semantics.

1. Transaction Logging

During normal processing, all un-terminated transactions in the system are kept track in an *Active Transactions Table (TransTable)* at each site, respectively. For a transaction T, its *T-Log* generation is triggered by *transaction commands.* For example,

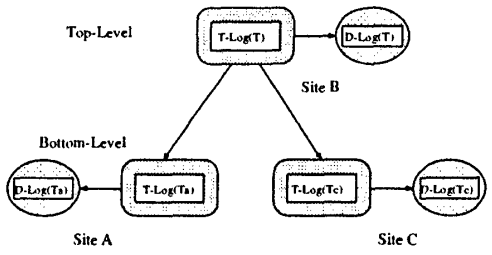

Figure 2: Structure of Nested Transaction Log

when T is started by a *begin_transaction* command, a **begin-record** is logged, when the first update operation of T is invoked, a **D-log-record** is logged, and when a child-transaction is started by a *begin_child* command, a **C-B-record** is logged.

2. Data Operation Logging

To keep track the status of updated disk pages, two status tables *DirtyPageTable* and *pageLSN Table* are used. The second one is newly designed to keep *pageLSN* information of all accessed pages.

In WAKASHI, a *write operation* is started by *Write-fault* and finished by *Page-out*. According to WAL protocol, the UNDO information are recorded before *Write-fault*, and the REDO information are recorded after *Page-out*.

4 Recovery Algorithm

When a site is crashed, all transactions executed at this site need be recovered. The policy for crash restarting is that all local recoverys at crash sites are done first, then failed nested transactions including orphan and un-terminated top-level transactions are found by global recovery system, finally, local recovery systems are commanded to compensate them. The recovery steps are shown in Fig. 3.

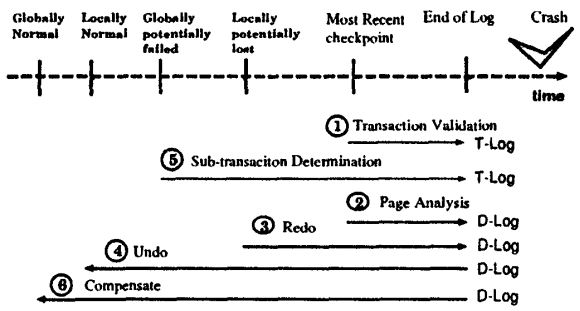

Figure 3: Steps of Recovery Processing

1. Global Recovery Algorithm

In the global level, a two-pass recovery procedure is adopted after a site is restarted. If more than one sites are crashed, the algorithms are executed distributely. (1) **Validation Pass:** Finding the most recent information about transaction executions and log status before crash, and determining which local transaction should be committed or aborted. Besides, finding the *pseudo failure* by *Termination-query* technique. (2) **Determination Pass** : Finding un-terminated nested transactions and orphan transactions, and Undoing them to restore the database into a most recent consistent status before crash.

2. Undo Processing

Because the log-records of a nested transaction are structured as a forest, a nested Undo is carried out by traversing the Log Forest. In the case of transaction aborting, the page in every heap must be purged before the *Write-lock* is released.

3. Local Recovery Algorithm

In the local level, ARIES three-pass recovery procedure is adopted and necessary extensions are made in consistent with global level recovery processing and *pageLSN Table*. (1) **Analysis Pass:** Finding the most recent information about page updates and log status before crash. (2) **REDO Pass:** Restoring the database into a most recent status before the crash. (3) **UNDO Pass:** Restoring the database into a most recent consistent status before crash.

5 Conclusions

The features of our recovery system include: (1) supporting the consistency of distributed object-oriented database system. (2) supporting the properties(atomicity and durability) of advanced transaction, and (3) supporting the performance of the next-generation system. The implementation of the recovery system is in progress. The performance will be further evaluated.

Reference

[1] Bai,G., and Makinouchi,A., "WAKASHI/D: A Distributed Paged-Object Server for Storage Management of New Generation Databases", Proc. of ADTI, Nara, 1994.

[2] Bernstein,P.A.,Hadzilacos,V., and Goodman,N., Concurrency Control and Recovery in Database Systems. Addison-Wesley, MA,1987.

[3] Mohan,C., et al, "ARIES: A Transaction Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging", ACM TODS, March 1992.