A Recovery Technique Using Client-based Logging in Client/Server Environment

Yong-Mun Park [1] and Chan-Seob Lee , Dong-Hyuk Kim, Eui-In Choi [2]
[1] Department of Network Engineering Technology, ETRI,
161 Gajeong-Dong, Yuseong-Gu, Daejeon 305-350, Korea
Tel. +82-42-860-5149, Fax.: +82-42-860-6858
[2] Dept. of Computer Engineering, Hannam University,
133 Ojeong-Dong, Daeduck-Gu, Daejeon, 300-791, Korea
e-mail : ympark@etri.re.kr, {cslee, dhkim, eichoi}@dblab.hannam.ac.kr

Abstract: The existing recovery technique using the logging technique in the client/server database system only administers the log as a whole in a server. This contains the logging record transmission cost on the transaction that is executed in each client potentially and increases network traffic. In this paper, the logging technique for redo-only log is suggested, which removes the redundant before-image and supports the client-based logging to eliminate the transmission cost of the logging record. Also, in case of a client crash, redo recovery through a backward client analysis log is performed in a self-recovering way. In case of a server crash, the after-image of the pages which needs recovery through simultaneous backward analysis log is only transmitted and redo recovery is done through the received after-image and backward analysis log. Also, we select the comparing model to estimate the performance about the proposed recovery technique. And we analyzed the redo and recovery time about the change of the number of client and the rate of updating operation.

1. Introduction

The existing recovery technique, in a client/server database system, transmits all the logging records to a server when a transaction is executed in each client. This means that the server administers the logging as a whole and it increases network traffic by the logging record transmission. Also, transaction execution can be delayed by the logging record transmission as the time of commit can be possible only after recording of all the logging record for each transaction.

The server workload for recovery action is heavy since it needs to use sever log in case of a system crash and the unnecessary analysis log increases in case of a client crash[6,7].

The proposed recovery technique eliminates the logging record transmission cost by supporting the client-based logging which records the logging record, which is produced from a transaction process in each client, in its own logging file. It does undo immediately after the identification of the lost page through one time of backward client analysis log in case of a client crash. Also, it transmits the after-image of the pages requiring the recovery through simultaneous backward client analysis log in each client to a server in case of a server crash and does Undo to the lost page using the received after-image and backward server analysis log in a server.

In chapter 2, transaction process & logging, execution of checkpoint, logging cut off and system recovery action are suggested to solve the problems of the existing recovery technique. In chapter 3, we select the comparing model to estimate the performance about the proposed recovery technique. And then adapted to the simulation environment of this paper. And we analyzed the redo and recovery time about the change of the number of client and the rate of updating operation. Finally, in chapter 4, the conclusion and the direction for further research are described.

2. Suggested Recovery Technique

This chapter describes transaction process & logging and recovery technique in case of system crash.

2.1 Transaction Process & Logging

Recovery management of the client records redo-only log record that has the same structure as in Table 1 in the client log.

Table 1. Structure of log record

LSN	{nodeID.}trID	type	PgID	AfterImg
log record ID	{node ID.} Transaction ID	log record type	Page ID	value of after update

In the case that the log record of the first updating operation, as top 1 bit is reserved in transaction identification trID among updating operations in Table 1, the top 1 bit of the transaction identification is modified from "0" to "1" as a bit operation (This supports the dynamic structure of the committed transaction list).

Log records are: 'update' - page change from the previous to a new one by a transaction operation, 'commit'- the completion of transaction operation, 'ckp'- removal of unnecessary log record, and additional 'afterImg' log record. This 'afterImg' log record keeps the current figure of the corresponding page in case there is no after-image for a transaction abort.

The server log has the same structure as in the client log, but it adds a recent commit log record. This log records 'recCommit' in log record type, before the Ack transmission, when the server permits the commit of nodeID.trID, then records all the updated pgIDs in the corresponding transaction by sorting them in '#', in the afterImg. Then, it creates a recent commit log record and records it in log.

As above, as the recovery manager of the server produces and records a recent commit log record, only the after-image of the page requiring recovery can be requested to each client in case of a server crash. The afterImg(pgIDs) of a recent commit log record uses the same server PageInfoTbl as in Table 2.

As the PageInfoTbl has the stored pages and the locked information of the executing transaction in the buffer, the entry of the corresponding page can be deleted in

PageInfoTbl when updating transaction of it is committed or aborted without saving the page - its updated information is taken in a disk - in buffer.

Table 2. Structure of PageInfoTbl

pgID	dirty	Index	lock	nodeID.trID	recUpdateLSN	abortLSN
page ID	update status	Buffer index	lock inf.	nodeID.trans actionID	recent updating log record ID	AfterImg LSN

recUpdateLSN in PageInfoTbl is the LSN of the updated log record which is used in the recent corresponding page and abortLSN takes LSN of the updated log record on the transaction recently updated & committed corresponding page.

The recUpdateLSN of server PageInfoTbl and abortLSN are applied to page LSN in the page header as the LSN of server log record. The recUpdateLSN and abortLSN of the client PageInfoTbl are the information in the client PageInfoTbl as the LSN of each client log record. This information is not applied to the page LSN in the page header. The recUpdateLSN and abortLSN of server PageInfoTbl represents the LSN of a recently committed log record as well as the LSN of an updated log record. So, to differentiate this, top 1 bit of page LSN, recUpdateLSN and abortLSN of PageInfoTbl are reserved and modified from " 0' to '1'.

In the creation of recent commit log record in the server, record pgID - which is under write lock by the corresponding transaction among the pages of PageInfoTbl in the server - in afterImg(pgIDs) of recent commit log record. Then, record the LSN of the recent commit log record, by using a bit operation and modifying the top 1 bit to '1', in abortLSN of Table 2.

If the entry is added in PageInfoTbl, abortLSN, record as '0' when page LSN of the page header in server is smaller than the truncatedLSN of server, and if not, record page LSN in abortLSN. When the entry is added to the client PageInfoTbl by page cache due to missing of the corresponding page in client, record '0' abortLSN of PageInfoTbl.

As the page Pi is updated in the transaction of each system, logging and the modification of PageInfoTbl [Pi].abortLSN is processed as below(i shows page ID).
If the PageInfoTbl[Pi].abortLSN is '0', then step1 is processed. If it is not, then step2 is. When the top 1 bit of page LSN to be updated is 'l', step1 is processed.

Step1: Record 'afterImg' in a log record type and make an additional after-image log record by recording the current page value in the afterImg of the log record and record it in the log. Then, record the LSN of the after-image log record in PageInfoTbl[Pi]. abortLSN.

Step2: Update pages and make updated log records and record it in log. Record the LSN of the updated log record in PageInfoTbl[Pi].recUpdateLSN. Generally, Process step1 can be disregarded since having prevLSN as '0' is rare.

Form a commit log record in each system when the transaction is committed and record it in log. Record the PageInfoTbl[Pi].recUpdateLSN in PageInfoTbl[Pi].abortLSN of the page Pi under 'write' lock by the corresponding transaction, then unlock 'write' lock. In case of transaction abort, abort the transaction with the afterImg of the corresponding log record using PageInfoTbl[Pi].abortLSN of the page Pi under 'write' lock by the corresponding transaction, then unlock 'write' lock. Record the LSN of the corresponding page in the recUpdateLSN of PageInfoTbl when the updated page is transmitted to a server though the committed transaction in each client.

2.2 Recovery action in case of a server crash

The suggested recovery technique in the server crash, in order to recover without combining each client log, requests an after-image of the page to be restored through recent commit log record analysis in a server to the corresponding client.

The rebooted server requests the after-image of page to redo or to be restored through the backward sever analysis log by recovery controller to the corresponding client.

In the suggested recovery technique in case of a server crash, each client transmits the after-image of page to recover to the server using the backward client analysis log. At the same time, the server executes the redo for the lost page using the backward server analysis log and requests the after-image of the page to recover to the corresponding client. Then, execute the redo using the received page after-image. Therefore, the server workload and required time for analysis log for recovery are reduced through simultaneous analysis log and redo recovery using backward analysis log in each system. Also, only transmitting the after-image of page to recover in a client minimizes network traffic.

2.3 Recovery action in case of a client crash

The pages to recover, in a client crash, are the updated ones by the committed transaction, but not transmitted to a server. After the server senses the client crash, it transmits all the pages - under write lock by the crashed client in its PageInfoTbl - to a rebooted client. The recovery manager of the rebooted client executes a redo recovery process of itself, as shown in Figure 1, through backward analysis log.

In case of additional after-image logging record analysis, it is processed next. If the page ID of the log record is in its PageInfoTbl without updating the corresponding page, redo is done through the after-image of the log record. Then, dirty('1'), recUpdateLSN(logRec.LSN) and abortLSN (logRec.LSN) of the corresponding page is modified in PageInfoTbl for no more redo.

When the updating log record is analyzed, it processes in the order below step1, step2 only in the case of commitLst with trID of the corresponding log record.

Step1: Redo and Modify are executed as in additional after-image log record analysis.

Step2: If the reserved top 1 bit is '1' in log record trID, it means the first updating operation of the corresponding transaction, so the corresponding trID is deleted in the commitLst

Recovery manager in the client can restore the lost page by redo once through the backward client analysis log of

itself and can reduce the time for analysis log as it only uses its log.

3. The performance of the client-based logging

The purpose of the simulation is the comparative analysis to the proposed recovery technique with existing technique about the performance of redo and recovery time in this paper. The simulation was proceed to focus to the recovery of database by suggested recovery technique when the server system has crashed

The simulation of redo algorithm that suggested recovery technique in this paper was run using AweSim simulation package version 3.0. The parameters that used in this simulation were set up depends on the performance estimation parameters, which is suggested in ESM/CS of EXODUS[5].

The simulation of the suggested recovery technique adapted that when the server crashed, concurrently, the suggested recovery technique transmits to the client only after-image that the client has to recover. At the same time, server redoes the crashed page using backward log analysis and the server request the after-image that it has to recover to the client.

3.1 Comparisons as the number of client

In this section, we estimate the redo time according to the number of client and we compare and analysis to the influence of the recovery time when it was changed. Figure 1 represents the change of the average redo time as the number of client increase from 1 to 5.

In this experiment result, EOS technique represents that the average redo time has uniformly according to the number of client has increase. But ESM/CS and suggested technique has increase growingly.

In case of EOS, This was analyzed that because the model has the special and exclusive log buffer in order to construct of the redo only log and we should create transactions to be occur by same probability. Accordingly, we know that the suggested recovery technique estimated lower than the EOS and ESM/CS in case of the number of clients is less about the average redo time but it estimated higher than the comparing models at the point that the number of client is 3.

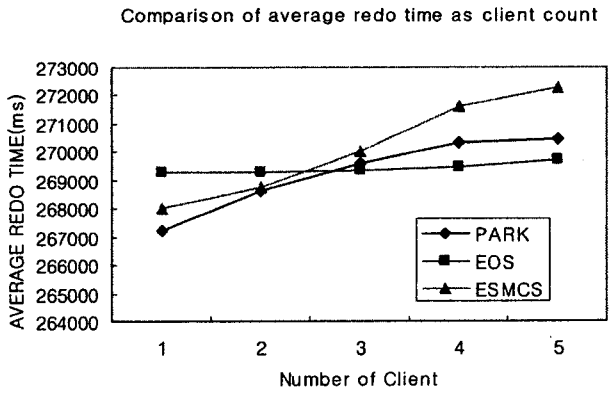

Fig. 1. Comparison of average redo time as the number of clients

Figure 2 represents the comparison result of the average recovery time according to the number of client. In this comparison result, the suggested technique estimated that the average recovery time has increasingly according to the number of client has increase. But ESM/CS technique has increase slowly.

In case of ESM/CS, The logging overhead has decrease because it was adopts to the logical logging, but redo and undo operation has many occurred. And it was analyzed that because the model has occur at 3 times for log access time at recovery.

In the comparison of the average recovery time as the number of client, Low estimation result in the suggested recovery technique, as each clients concurrently executed by backward analysis to the log when the server system has destroyed, can be decided that because the server sends only after-image of the corresponding page to client that it has to recover, and the server request after-image of the page to the client that it has to recover, and then it redoes using after-image of the received page immediately when the server receives the after-image of corresponding page.

It regards the response time of suggested recovery technique as uniform in network process model.

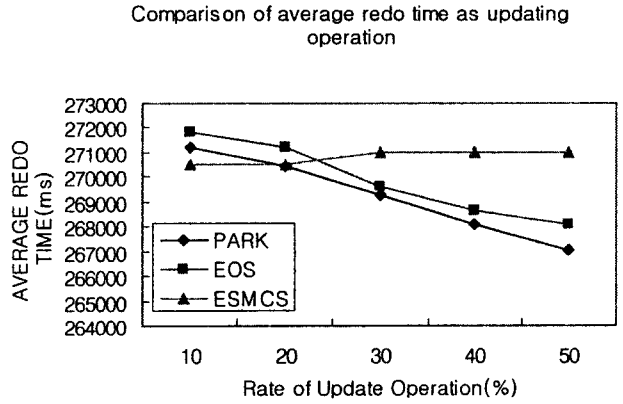

Fig. 2. Comparison of average recovery time as the number of clients

3.2 Comparisons as the rate of updating operation

In this section, we estimate the recovery time as the number of redo and undo operation, and we comparative analysis the suggested recovery technique to existing technique about the result of recovery time estimation.

Figure 3 represents the result of comparison about the average redo time as the change of the updating operation rate. In this comparison, we can aware that the average redo time has largely decrease as the rate of updating operation changes from 10 % to 50% in case of suggested recovery technique and EOS.

We can predict that the unnecessary redo action has decrease as the updating operation is frequently occurred when the number of client is supposed to uniform. But this result can be described that the exact analysis can be made as estimate depend on the real transaction data and similarly experimental data, because transaction is distributed to each node as the probability distribution under the assumption that the transaction has create by uniform rate.

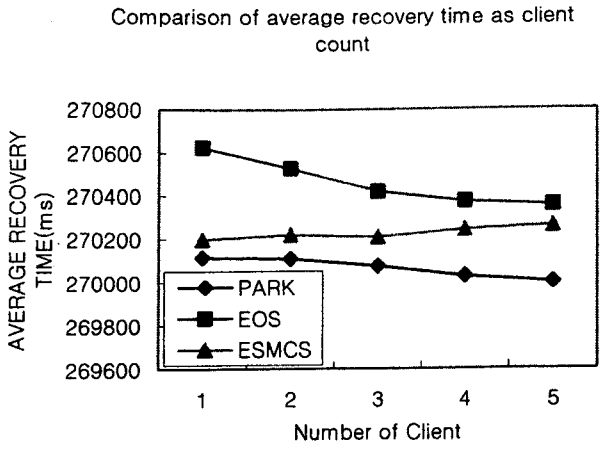

Fig. 3. Comparison of average redo time as the change of updating operation

Figure4 represents the comparison result of the average recovery time as the change of the occurrence rate of updating operation. In this comparison result, as the occurrence rate of updating operation has change from 10% to 50 %, the EOS and ESM/CS technique estimated higher than the suggested technique within the range of 50%, which is the range of estimation in this simulation. That is, if the occurrence of updating operations more frequently does not occurred by 50%, then the suggested recovery technique estimated better in performance.

In the suggested recovery technique, because the decreasing rate of average recovery time according to the change of occurrence rate of updating operation is very slow curve, it can be decided that the ESM/CS technique, which is the decreasing rate of recovery time is large, has more profitable when the occurrence rate of updating operation is larger than 50%.

Also, the suggested recovery technique is more profitable to redo algorithm in case of that the updating operation of same page has more often occurred, which is not considered in this simulation model. But as shown in the result of the Figure 4, which is the transaction was created by probability; it is need to additional restraint in case of updating to each other different pages.

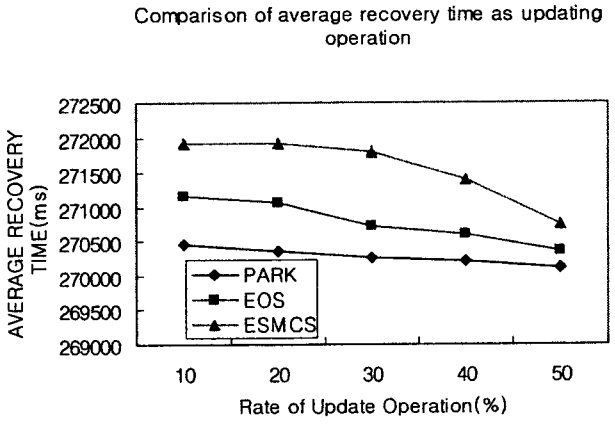

Fig. 4. Comparison of average recovery time as the change of updating operation

4. Conclusion

The suggested recovery technique, with the elimination of the unnecessary before-image, has reduced logging overhead by recording & administering redo-only record on log and has eliminated logging record transmission cost by supporting client-based logging. In case of a client crash, it identifies the after-image of the lost page by using a one time of backward client analysis log of itself and executes redo promptly.

In case of a server crash, to recover without combining the log of each client, the server requests the identification of the page after-image to recover in recent commit log record and executes redo using the received page after-image. At the same time, each client transmits the requested page after-image from the server by using simultaneous backward client analysis log.

Thus, the suggested recovery technique supports self-recovering of the client and in case of a server recovery, helps fast recovery by reducing required time for analysis log. Also, network traffic has been minimized in the case of recovery since the page after-image to recover in each client is transmitted to the server. In the suggested recovery technique, access overhead of information structure which needs maintain-ing & administering has been minimized by dynamic commitLst add or delete.

The result of performance analysis is estimated better than the EOS and ESM/CS, which is comparing model, in the comparison of redo and recovery time according to the change about the number of client and the occurrence rate of updating operation. Also, it was analyzed that total recovery time has can be completed in the case of the updating operation of each different transactions are often create more than 50%.

References

[1] P. A. Bernstein, et al., "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1987.

[2] A. Delis and N. Roussopouls, "Modern Client-Server DBMS Architectures," Proc. ACM SIGMOD RECORD, pp.52-61, 1991.

[3] David Lomet, Gerhard Weikum, "Efficient Transparent Application Recovery in Client-Server Information Systems," Proc. ACM SIGMOD, pp.460-471, 1998.

[4] C. Mohan & Don Haderle, et al, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," Proc. ACM TODS, pp.94-162, 1992.

[5] M. Franklin, M. Carey, "Crash Recovery in Client-Server EXODUS," Proc. ACM SIGMOD, pp.165-174, 1992.

[6] Jim Gray, Andreas Reuter, "Transaction Processing: Concepts and Techniques," Morgan Kaufmann, 1993.

[7] Theo Haerder, Andreas Reuter, "Principles of Transaction-Oriented Database Recovery," Computing Surveys, pp.287-317, 1983.

[8] Tobin J. Lehman, Michael J. Carey, "A Recovery Algorithm for A High-Performance Memory-Resident Database System", ACM, pp.104-117, 1987.