

# Recovery of Transaction Processing by Group Log and Snapshot

Koji Yamada, Tetsuya Egawa, Yuhsuke Tsujiguchi, Motoyasu Nagata  
*Osaka Kyoiku University*  
*Kashiwara, Osaka 582-8582, Japan*  
*TEL/FAX +81-729-78-3672*  
*nagata@cc.osaka-kyoiku.ac.jp*

**Abstract**—The recovery of the transaction processing is time consuming due to operation for respective dirty log. To overcome this problem for restart of the recovery, we will present recovery of the transaction processing, motivated by development of the network storage. We will propose two restart protocols, the first one utilizes group log that is collection of dirty logs during fixed time interval and the second one utilizes snapshot that is a copy of database at some instance. We will evaluate performance of our proposed protocols to compare with the traditional restart protocol.

**Keyword:** recovery, restart, checkpoint, snapshot, performance evaluation

## I. INTRODUCTION

We will study system fault meaning loss of volatile memory. For such a case, recovery manager executes restart processing to restore the normal state. Two roles are given to the restart processing. The first role is synchronization between state of the log and state of persistent database with the aid of periodical checkpointing. The second role is reduction of restart time in the faults of resource or node. Frequency of checkpointing faces tradeoff about performance evaluation. That is, despite the fact that frequent checkpointing makes the restart fast, it causes excessive overhead.

To overcome the tradeoff, we will propose two protocols about the restart processing. The first protocol utilizes group log that is collection of logs during fixed time interval and the second protocol utilizes snapshot that is a copy of database at some instance. We will evaluate performance of our proposed protocols to compare the traditional restart protocol.

Organization of this paper is described. Section 2 describes well-known restart processing using sharp checkpointing. Section 3 presents restart using group log and section 4 presents the restart using snapshot, respectively. Section 5 evaluates performance for our proposed protocols. Section 6 concludes.

## II. RESTART USING SHARP CHECKPOINT

The checkpoint used in ordinary restart protocol is called sharp checkpoint. This checkpoint synchronizes all the volatile memory with persistent memory by periodical flushing. According to memory increase, service is obliged to stop for some seconds. Furthermore, due to the fact that the restart processing must redo the log each by each, the longer the

checkpoint interval becomes, the longer the restoring time becomes. Figure 1 shows recovery model with using log.

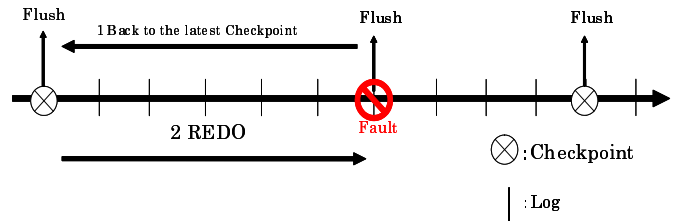


Fig. 1. Recovery using log

In this sharp checkpointing, the longer the time interval between checkpoints, logs by which the recovery manager must redo increase. Along with increase of the logs, time cost becomes large. On the other hand, increase of checkpoints decreases restart time but reduces efficiency of the transaction processing.

Total processing time by the following equation. The part indicates the processing time takes at the recovery.

$T$ : Total processing time

$Tl$ : Total time in processing and logging transaction

$Tc$ : Total time in checkpointing

$Trc$ : Total time to read data of the last checkpoint

$Trec$ : Total time to flush in recovery

$Trl$ : Total time to read log in recovery

$$T = Tl + Tc + (Trc + rl + Trec)$$

## III. RESTART USING GROUP LOG

We consider that the previously described delay of the restart is due to each by each processing of the logs. As one of alternatives to overcome the problem, we will propose restart protocol using group log. The collected logs during assigned time interval is called group log.

The group log is generated by the following procedure.

- 1) The log manager generates log according to each transaction processing.

- 2) When logs are accumulated to some extent in the cache, the log manager generates group log which is collection of these logs.

The checkpoint algorithm using group log is described by the procedure in checkpoint time. This procedure is as the same as that of the sharp checkpoint [Bernstein 1997].

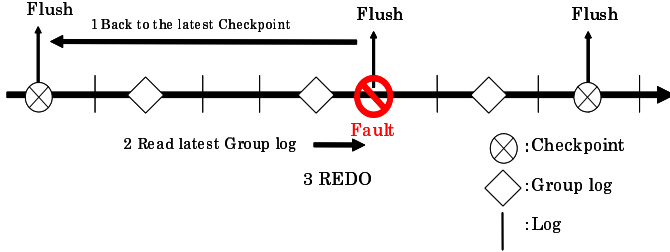


Fig. 2. Recovery using group log

Figure 2 shows recovery model with using group log. Next, the restart algorithm using group log is described by the following procedure in fault occurrence.

- 1) The recovery manager reads last checkpoint record.
- 2) The recovery manager identifies latest group log after the last checkpoint (group log with maximum group log sequential number (GLSN)).
- 3) The recovery manager reads data of the group log into volatile memory.
- 4) The recovery manager identifies the log with minimum LSN generated after the last group log.
- 5) The recovery manager reads data of the logs from minimum LSN to the latest log into volatile memory.
- 6) The recovery manager writes dirty pages corresponding to these logs into the persistent database.
- 7) The recovery manager starts accepting new update, commit and abort operations again.

In this algorithm, the recovery manager does not respectively need to open, read and close for the logs which have been generated prior to the last group log. The reason is that the last group log includes data of these logs. The recovery manager can reduce time in file manipulation. However, unless the fault occurs, generation of the group log is wasted.

Total processing time by the following equation.

$T_g$ : Total time to create Group Log

$T_{rg}$ : Total time to read Group Log in recovery

$$T = T_l + T_c + T_g + (T_{rc} + T_{rg} + T_{rl} + T_{rec})$$

#### IV. RESTART USING SNAPSHOT

As another one of alternatives to overcome the delay problem in traditional checkpointing, we will propose new restart protocol using snapshot. This restart protocol utilizes logs generated by each transaction and the snapshot that is a copy of database at some time. This protocol aims to reduce time required for the restart in the fault occurrence.

The snapshot is generated by the following procedure.

- 1) The log manager generates log according to each transaction processing.
- 2) When logs are accumulated to some extent in the cache, the log manager takes snapshot that is a copy of database at this time.

The checkpoint algorithm using snapshot is described by the procedure in checkpoint time. This procedure is as the same as that of the sharp checkpoint [Bernstein 1997].

Figure 3 shows recovery model with using snapshot.

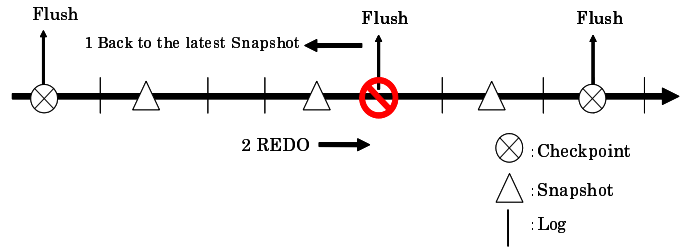


Fig. 3. Recovery using Snapshot

Next, the restart algorithm using snapshot is described by the following procedure in fault occurrence.

- 1) The recovery manager identifies latest snapshot after the last checkpoint (snapshot with maximum snapshot sequential number (SSN) called snapshot anchor).
- 2) The recovery manager reads data of the snapshot into volatile memory.
- 3) The recovery manager identifies the log with minimum LSN that is generated after the last snapshot.
- 4) The recovery manager reads data of logs from minimum LSN to the latest log into volatile memory.
- 5) The recovery manager writes dirty pages corresponding to these logs into the persistent database.
- 6) The recovery manager starts accepting new update, commit and abort operations again.

This procedure doesn't need to read data of last checkpointing, when the recovery manager already have done checkpointing. Because the latest snapshot includes all of data. As this procedure records the snapshot periodically, this protocol does not depend on the time interval of the checkpoint. However, procedure of this checkpoint is the same as that of the sharp checkpointing. There is possibility of resource consumption in obtaining the snapshot.

Total processing time by the following equation.

$T_s$ : Total time to create Snapshot

$T_{rs}$ : Total time to read Snapshot in recovery

$$T = T_l + T_c + T_s + (T_{rc} + T_{rs} + T_{rl} + T_{rec})$$

The total processing time does not depends on time interval of checkpointing because of periodical snapshot. Even if time interval of the checkpointing is long, the recovery manager can redo from the last snapshot.

## V. PERFORMANCE EVALUATION

We will evaluate performance about our proposed checkpointing protocols using the global log and the snapshot. We experimented under various combination of parameters. We obtained valuable experimental results. We will evaluate for these observation.

### A. Experimental Environment

The experiment was carried out using machine at Nagata Laboratory in Faculty of Information Science, Osaka Kyoiku University. Specification of the machine is shown in Table 1.

TABLE I  
COMPUTER SPEC

OS	Windows 2000 professional service pack 4
CPU	Intel Pentium 4 1.5GHz
RAM	512MB
Language	C
Compiler	Borland C++ Compiler 5.5

### B. Experiment

We used pseudo database in this experiment. We assume that capacity of buffer area is larger than that of database in the machine. We assumed that pages of database is 1000 and number of issued transactions is 1000. in experiments 1-3. We also assumed that page numbers of database is 10000 and number of issued transactions is 10000 in experiment 4. Generated log is saved in the persistent memory to cope with the fault. Simulated fault is occurred every 100 transaction processings.

We experimented four kinds of experiments characterized in Table 2.

TABLE II  
CHARACTER OF FOUR KINDS OF EXPERIMENTS

ex1	Relation of frequency and processing time in Checkpointing using log and total processing time.
ex2	Relation of frequency and processing time in Checkpointing when faults occur
ex3	Effectiveness of Group Log and Snapshot
ex4	Comparison of three techniques, Log, Snapshot, Snapshot

Experment1 tested relationship between checkpoint frequency and processing time in non-existence of the fault occurrence. Experment2 tested relationship between checkpoint frequency and processing time in the fault occurrence. Experment3 tested relationship between frequencies of group log or snapshot and recovery time or processing time for fixed time interval of checkpointing in the fault occurrence. Experment4 tested comparison between checkpoint frequency and frequencies of group log or snapshot in the fault occurrence.

1) *Experiment1*: For a situation where fault occurrence is not found, an experiment is carried out by varying time interval of checkpoint in the sharp checkpointing. We will investigate whether relationship between frequency of checkpoints and

processing time is tradeoff. Experimental results about relationship between number of checkpoints and processing time is shown in Table 3 and Figure 4.

TABLE III  
RELATION BETWEEN FREQUENCY AND PROCESSING TIME IN CHECKPOINTING USING LOG

Frequency of Checkpoint	5	10	20
Total processing time	11.5	12.2	12.7

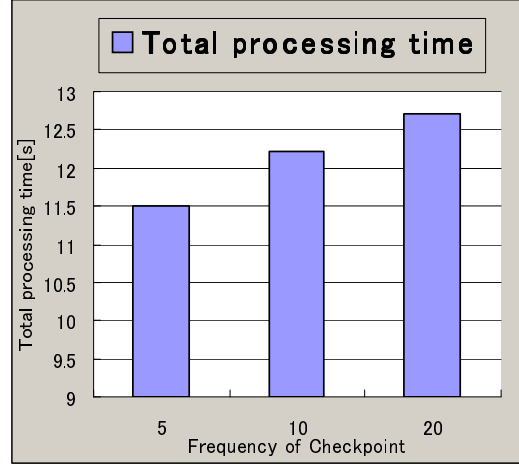


Fig. 4. Processing time using log when fault doesn't occur

From this experimental result, we can observe that increase of checkpoints leads to increase of processing time.

2) *Experiment2*: For a situation where fault occurrence is found, two experiments are carried out by periodically varying time interval of checkpoint in the sharp checkpointing. We will investigate which time interval of group log and which time interval of checkpoint can lead to reduction of recovery time

Case1: Fault Occurrence at Fixed Point Experimental results about relationship between number of checkpoints and its total processing time is shown in Table 4. The recovery processing time and normal processing time constituting the total processing time is also obtained for number of checkpoints, respectively.

From the observations, processing time is almost same for three parameters about number of the checkpoints. For a case of fault occurrence at fixed point, we can observe that decrease of checkpoints leads to increase of the restart time, and increase of check points can reduce restart time.

For five checkpoints, we executed simulation in the following manner.

- 1) Flush logs every 200 transaction processing.
- 2) Read 66 logs in the first recovery.
- 3) Read 100 logs for the second recovery.

TABLE IV  
PROCESSING TIME IN CHECKPOINTING WHEN FAULTS OCCUR AT  
REGULAR POINT

Frequency of Checkpoint	5	10	20
Recovery time	4.28	3.50	1.16
Normal processing time	11.2	11.4	11.8
Total processing time	15.5	14.8	13.0

4) Repeat five times for the above two steps until 1000 transactions are processed.

The recovery manager reads 830 logs after occurrence of ten faults in such a computing  $(66 + 100) * 5 = 830$ .

Case2 : Fault Occurrence at Varied Point Experimental results about relationship between number of checkpoints and its total processing time is shown in Table 5 and Figure 5. The recovery processing time and normal processing time constituting the total processing time is also obtained for number of checkpoints, respectively. From the observations,

TABLE V  
AVERAGE PROCESSING TIME IN CHECKPOINTING WHEN FAULTS OCCUR 10  
TIMES

Frequency of Checkpoint	5	10	20
Recovery time	3.84	2.78	1.55
Normal processing time	11.4	11.6	11.8
Total processing time	15.2	14.2	13.3

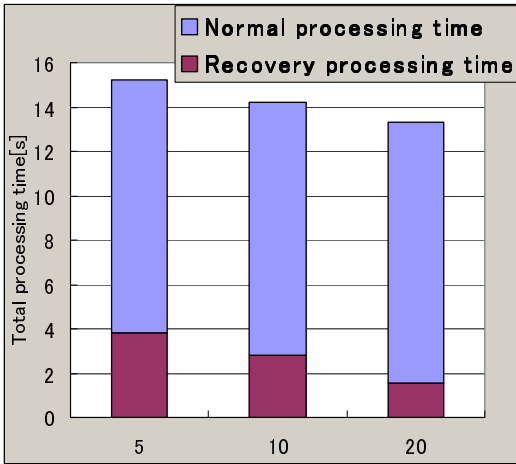


Fig. 5. Processing time using log when fault occur

normal processing time becomes long a little according to increase of number of the checkpoint. This result is considered due to flush time. For a case of fault occurrence at varied point, we can observe that decrease of checkpoints leads to increase of the restart time, and increase of check points can reduce restart time. This influences the total processing time.

3) *Experiment3*: For a situation where fault occurrence is found, an experiment is carried out by fixing time interval of checkpoint in the checkpointing using group log and snapshot. Frequencies of the group log and the snapshot are varied. We assumed that number of checkpoints is five. Experiments are carried out for one, three, ten and twenty group logs and snapshots, respectively.

TABLE VI  
PROCESSING RATIO, RECOVERY RATIO USING GROUP LOG OR SNAPSHOT

Frequency of Group Log or Snapshot	log	1	3	10	20
Total time using Snapshot	100%	96%	90%	105%	126%
Total time using group log	100%	79%	79%	94%	110%
Recovery time using Snapshot	100%	69%	39%	14%	11%
Recovery time using group log	100%	69%	41%	17%	13%

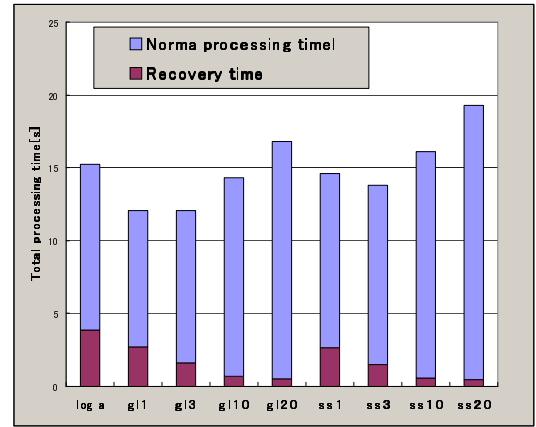


Fig. 6. Processing time using Log, Group Log and Snapshot

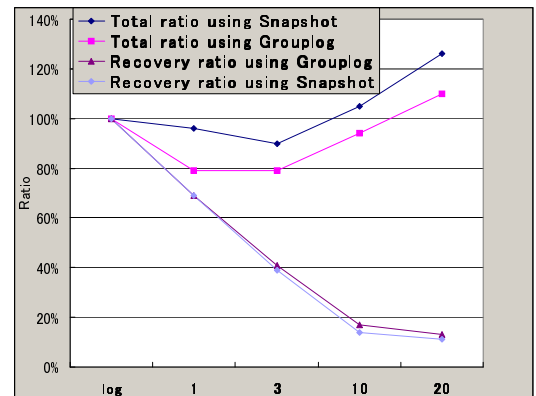


Fig. 7. Processing ratio, Recovery ratio

From the observations, recovery time becomes fast according to increase of number of the group logs and the snapshots. However, increase of the group logs and the snapshots lead to increase of the total processing time due to unused group

logs and the snapshots. Table 6 shows ratios (percentages) of recovery time and total processing time using the group log and snapshot when the recovery time and total processing time using log a are standard values of 100 percentage. The optimal result was obtained for three group logs and three snapshots, respectively. The group log protocol reduced recovery time of 59 From this result, many frequency or little frequency of the group logs and the snapshots is not good. We also observe that the result using the group log is better than that using the snapshot. This result is due to the consideration about time cost that the group log protocol flushes only dirty pages but the snapshot protocol flushes all the pages.

4) *Experiment4*: Finally, experiments are carried out to evaluate performance for almost same frequencies of the group logs and the snapshots.

Case 1 : Non-Existence of Fault Occurrence

Test of the transaction Processing are executed for 50 checkpoints (Log a), 50 checkpoints and 150 group logs (group log), 50 checkpoints and 150 snapshots (Snapshot) and 200 checkpoints (Log b), respectively. The result is shown in Figure 8.

For a situation where fault occurrence is not found, generation of the group log and snapshot leads to performance reduction. Because of small file size, the group log protocol is better than the snapshot. The snapshot protocol was the slowest. From this result, the snapshot protocol is not effective.

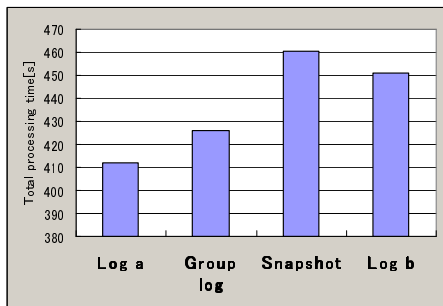


Fig. 8. Processing time in non-Existence of fault occurrence

Case 2 : Fault Occurrence

Tests of the transaction processing are executed for fault occurrence of 20 times in 10000 transactions. Table 7 and Figure 9 show comparison of total processing time and recovery time for Log a, group log, Snapshot and Log b, respectively. Test of Log a resulted in short processing time but long recovery time due to long time interval of the checkpoint. The recovery times were almost same for group log, Snapshot and Log b, respectively. Among these tree tests, group log test resulted in the shortest total processing time. This is due to the observation that generated file size is small and flush number is small in group log case. From this result, for the situation where checkpoint frequency using only logs and group log

snapshot frequency are almost equal, the group log is effective protocol.

TABLE VII  
COMPARISON OF THREE TECHNIQUES, LOG, GROUP LOG, SNAPSHOT

		Recovery time	Total time
Checkpointing : 50 times	Log a	8.0	418
Checkpointing : 50 times	Group Log 150 times	3.0	415
Checkpointing : 50 times	Snapshot 150 times	3.0	444
Checkpointing : 200 times	Log b	2.9	441

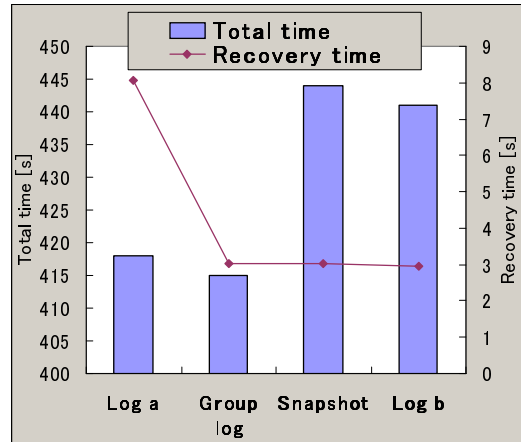


Fig. 9. Processing time and Recovery time when Checkpointing, Group Log, Snapshot are done at the same frequency

VI. CONCLUSION

To realize rapid recovery for occurrence of system fault in the transaction processing, we presented two protocols, that is, group log protocol gathering logs of dirty pages in fixed time interval and snapshot protocol gathering all the logs in fixed time interval. We also evaluated performance of our proposed protocol. From the performance evaluation, the recovery time can be reduced using the group log and snapshot for long checkpoint interval. On the other hand, the group log protocol shows effectiveness for the case where frequencies of the group log and checkpoint are almost same as the frequency of checkpointing.

REFERENCES

- [1] Philip A. Bernstein, Eric Newcomer, "Principle of Transaction Processing", Morgan Kaufmann Publishers, Inc,1997.
- [2] Jim Gray, and Andreas Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann Publishers, Inc,1993.
- [3] Andrew S. Tanenbaum, Maarten van Steen, "Distributed Systems: Principles and Paradigms, 1st Edition", Pearson Education Japan, Inc, 2003.
- [4] Ravi Sethi, "Programing Languages Concepts and Constructs" Bell Telephone Laboratories, Inc 1995.
- [5] S. Tokuda, T. Nakamura, S. Fujiwara, "Development of Expansion Differential Volume Feature for Embedded NAS Snapshot", Technical Report of IEICE, CPSY2004-52, pp.13-18, 2004
- [6] Jeffrey D. Ullman, "Principles of Database Systems Second Edition", Computer Science Press, Inc, 1982.