Weekly exercises IN3020&4020 (2024)
Week 11: Recovery Protocols

1 Review

1. What do we mean by recovery and why would we want to recover?

   Solution: Recovery is restoring a database to the most recent consistent state before failure. We want to recover from failures to ensure Atomicity and Durability of ACID principles.

2. Discuss at least two types of failures, one where disk is significantly damaged and one where it is not.

   Solution: Many possible answers. Two major groups of failures:

   (a) Disk is significantly damaged

   - natural catastrophe
   - persistent disk failure

   (b) Disk is not damaged

   - Computer failure (hardware, software, network)
   - Transaction failure (division by zero, constraint violation)
   - Local transaction errors (no data found)
   - Concurrency control enforced (serialisability violation, deadlock)

3. What is do we mean by a catastrophic failure? How can we recover from catastrophic failures?

   Solution: This can be disk failure, but also flood, storm, earthquake, fire damage and terrorist attacks. We must restore the entire database from the latest backup (e.g., physical tape or cloud). We can use a backup of the system log to avoid loosing all transactions performed since the last backup.

4. What is database cache and what are the main buffer replacement policies?

   Solution: Some disk blocks are mirrored in cache (main memory) buffers for quick access. Sometimes we have to replace a cache buffer with a new one. If the buffer is dirty (i.e., it was written by some transaction), it should be flushed to the disk before replacement (i.e., the new value should overwrite the old one). A common buffer replacement policy is LRU with domain separation, possibly with hot set and clock sweep.

5. How is the system log stored? What is it used for? What are its main entries?

   Solution: The system log is a sequential append-only file. It is used to keep track of executed operations of transactions. The entries are undo (old value of a written item), redo (new value of a written item), redo/undo, begin transaction, commit transaction, abort transaction, etc.

6. What is and why do we need write-ahead logging?

   Solution: Write-ahead logging means that a transaction cannot commit until all log entries for that transaction have been flushed to disk. This is necessary, for example, to be able to redo transactions that were committed before a failure when we use a deferred-update recovery protocol.

7. What do we mean by checkpoints for recovery and why are they needed?

   Solution: The checkpoint is a recovery subsystem operation, whose behaviour depends on a particular type of a checkpoint. A most basic one flushes relevant buffers and keeps a list of currently active transactions. Checkpoints are important because it makes it possible, in case of a fail, to safely return to a consistent state.

8. What is the difference between a basic (rigid) checkpoint and fuzzy checkpoints?

   Solution: The basic checkpoint suspend execution of all transactions and flush all main non-pinned dirty memory buffers. Fuzzy checkpoints allows us to continue transactions while flushing: it first issues begin checkpoint entry to the log, then flush all buffers while allowing transactions to continue, and finally issue end checkpiont with the list of active transactions. Then, the checkpoint becomes active after the end checkpoint command, but the recovery starts from begin chechpoint.

9. Describe the idea of the shadow recovery protocol. Discuss advantages and disadvantages of this protocol.

   Solution:

   - Flush each modified buffer immediately to a new place on disk, the old (shadow) version is kept on disk
   - If a transaction commits, the current version is used
   - If a transaction aborts, the shadow version is used

   One advantage of this system is that it can be used without a log (in a single user mode) and checkpoints. Disadvantages include lot of overhead, complicated garbage collection.

10. Describe the deferred-update recovery protocol. Discuss advantages and disadvantages of this protocol.

    Solution: The idea is to postpone any actual updates (i.e., flushes of involved dirty buffers) to the database until the transaction reaches the commit point (note that there is no requirement to flush at the commit point, it may happen any time later). In case of a fail, recovery starts from the active (i.e., the latest) checkpoint and redoes all writes of the transactions that are not committed by the checkpoint moment (i.e., active or not yet started) but committed by the end of the log (so, no undo log entries are needed). Its limitation is that we may run out of buffer space. So, it is applicable only if transactions are short with few-enough changes.

11. Describe the immediate-update recovery protocol. Discuss advantages and disadvantages of this protocol.

    Solution: The idea is that we allow flushes of dirty buffers before the transaction that have written this buffer commits (and also after that, so there are no limitations on this regarding the commit point, but everything must be flushed at every checkpoint). At the event of fail, the recovery starts from the active (i.e., the latest) checkpoint; it redoes all writes of the transactions that are not committed (i.e., active or not yet started) by the checkpoint moment but committed by the end of the log, and undoes all writes of the transactions that are not committed by the checkpoint moment and started but not committed by the end of the log. The advantage is that it the flushes do not need to coordinate with commits (only with checkpoints). The disadvantage is that we need to store both redo-type and undo-type entries in the log, as well as expensive checkpoints.

12. What are the main ideas of the ARIES recovery protocol? Discuss its advantages and disadvantages.

Solution: Similar to the immediate-update protocol, but the checkpoint logs the table of currently dirty buffers, including the pointers to the log entries of the earliest corresponding writes (which is a significant improvement, because it does not force any flushing, even at checkpoints). Then, recovery essentially starts not at the checkpoint, but at the moment of the earliest write of a dirty buffer in the table logged at the checkpoint.

2 Recovery protocols

1. Consider the following log of two transactions, T_1 and T_2, with undo/redo entries in the form

   [Action, Transaction, DataItem, UndoValue, RedoValue]

   and rigid checkpoints:

[checkpoint]
$[\text{begin}, T_1]$
$[\text{write}, T_1, X, 10, 11]$
$[\text{begin}, T_2]$
$[\text{write}, T_2, Y, 20, 21]$
$[\text{write}, T_1, Z, 30, 31]$
$[\text{write}, T_2, U, 40, 41]$
$[\text{commit}, T_2]$
$[\text{write}, T_1, V, 50, 51]$
$[\text{commit}, T_1]$

   Describe the action of the recovery manager under the immediate-update protocol if there is a crash and the last log record to appear on the disk is

   (a) $[\text{begin}, T_2]$

   (b) $[\text{commit}, T_2]$

   (c) $[\text{commit}, T_1]$

   Solution:

   (a) The recovery manager should undo the write action of transaction T_1—that is write 10 to X.

   (b) The actions of T_2, which is committed by the end of the log, should be redone—that is, the manager writes 21 to Y and 41 to V; at the same time, the actions of T_1 must be undone—that is, the manager writes 30 to Z and 10 to X.

Table 1: Schedule

T_1	T_2	T_3	T_4
$\mathtt{b}_1$			
$\mathtt{r}_1(X)$			
$\mathtt{w}_1(X)$			
	$\mathtt{b}_2$		
	$\mathtt{r}_2(Y)$		
	$\mathtt{r}_2(Z)$		
	$\mathtt{w}_2(Y)$		
$\mathtt{c}_1$			
Checkpoint t_1			
		$\mathtt{b}_3$	
		$\mathtt{r}_3(U)$	
			$\mathtt{s}_4$
			$\mathtt{r}_4(V)$
			$\mathtt{w}_4(W)$
	$\mathtt{c}_2$		
		$\mathtt{c}_3$	
System crash t_2			

(c) The manager should redo all actions of both transactions.

2. (*) For each of the following recovery protocols, discuss how the log must look like and discuss what the system would have to do to recover after the system crash in the schedule in Table 1 (you may write *NewValue* and *OldValue* instead of the non-given concrete values):

   (a) A deferred-update recovery protocol

   (b) An immediate-update recovery protocol

   Solution:

   - For deferred-update, the log is as follows (assuming that all log buffers are flushed by the crash moment; if it is not the case, then the log is a prefix of the following):

[begin, T_1]
[write, $T_1, X, \mathit{NewValue}$]
[begin, T_2]
[write, $T_2, Y, \mathit{NewValue}$]
[checkpoint, $\{T_2\}$]
[begin, T_3]
[begin, T_4]
[write, $T_4, W, \mathit{NewValue}$]
[commit, T_2]
[commit, T_3]

At the crash point, t_2, T_1, T_2 and T_3 had committed and T_4 had not. There is no need to deal with the operations of T_1 because it had committed before the checkpoint (i.e., not active by the checkpoint, which flushed all writes of T_1). The write operations of T_2 and T_3 must be redone according to the log, but all writes of T_4 are ignored. Since nothing is written to the disk before committing, this will effectively cancel (roll back) all effects of T_4.

- For the immediate-update, the log is the same (under the same assumptions), except that the write entries are undo/redo; for example, the second entry is as follows:

...
[write, $T_1, X, \mathit{OldValue}, \mathit{NewValue}$]
...

We don't have to care about T_1 since it is before the checkpoint. The committed transaction are T_2 and T_3. These have to be redone with the log entries on the disk. Transaction T_4 is not committed, so we have to undo the known operations of this transaction.

3. Assume a single-user environment and a shadowing recovery protocol. Describe how the log should be like and how this system can recover from a system crash.

Solution: No log is required. We have both the old versions and the new versions on the disk. We have two directories, the new one pointing to the new, modified block, and a shadow directory with pointers to the unmodified blocks. To recover after the system failure, we free the modified database pages and discard the new directory and use the shadow directory as the new one.

4. Let us consider the ARIES recovery protocol. Which log entries does it undo in the corresponding step? In particular, does it need to look before the checkpoint?

Solution: In the undo step, ARIES goes through the log from the end and backwards all the way to the earliest write of a dirty buffer at the checkpoint step, and undo all writes of non-committed (i.e., aborted) transactions that are not overwritten be committed ones (note that the latter do not exist for strict schedules and blocks as data items). It may have to look before the checkpoint, because a transaction with a dirty buffer by the checkpoint may be aborted at the crash.