Database Recovery Techniques

Recovery Concepts

- **Recovery from transaction failures usually means that the DB is restored to the most recent consistent state just before the time of failure.**

- **Log file is used to keep necessary information for recovery.**
- **<u>Typical strategy for recovery is:</u>**
  - **Backup data (from archival storage as tapes) is used with backup log to redo the committed transactions up to the time of failure in case of extensive damage (catastrophic failure).**
  - **Recovery techniques using log files to undo and redo the operations of transactions depend on the recovery (and also concurrency) techniques used.**

Main Recovery Techniques

1. **Deferred Update:**

   - **These techniques do not physically update the DB on disk until after a transaction reaches its commit point.**

   - **The updates are recorded in the local transaction buffer and in the log file for recovery.**

   - **These techniques need only to redo the committed transaction and no-undo is needed in case of failure (No-Undo/Redo).**

(cont)

2. **Immediate Update:**
   - **The DB may be updated by some operations of a transaction before the transaction reaches its commit point.**
   - **The updates are recorded in the log which must contain the old values (BFIM) and the new values (AFIM).**
   - **These techniques need to undo the operations of the uncommitted transactions and redo the operations of the committed transactions (Undo/Redo).**
   - **The Undo/No-Redo may be used in special case where all operations are recorded in the DB before the transaction commits.**

Cashing of Disk Blocks

- **Typically allocation of memory buffers (DBMS cache) is kept under the control of the DBMS (it contains the DB and also the log).**
- **A directory for the cache is used to keep track of the items within the buffers. The entry of the directory may be as: <disk page address, buffer location>.**
- **The buffers are replaced (flushed) to make space for new items.**
- **A dirty bit is used to indicate if the buffer has been modified and must be flushed to disk or not (this bit can included in the buffer entry in the directory).**

Cashing of Disk Blocks (cont)

- **A pin-unpin bit is used to indicate if the buffer can be written back to disk as yet or not (0 means it can be written - this bit can be included in the buffer entry in the directory).**

- **Flushing a page back to disk may be on the same original disk location (in-place updating) or may be in new location (shadowing).**

- **In-place strategy needs a log file to include the old values (BFIM), but shadowing does not.**

Write-Ahead Logging

- **The process of recording the BFIM of the data items in the log and enforcing the log to be recorded on disk before the BFIM is overwritten by the AFIM in the DB on the disk is generally known as Write-Ahead Logging (WAL).**
- **The log contains the appropriate entries for the recovery which are:**
  1. **Undo-type log entries which include the old values (BFIM).**
  2. **Redo-type log entries which include the new values (AFIM).**
  3. **The log may contain the read operations also which may be used for concurrency (as in cascading rollback).**

Write-Ahead Logging (cont)

WAL protocol for recovery algorithms that require both Undo and Redo:

1. **BFIM cannot be overwritten by its AFIM on the disk until all Undo-type log records of the updating transaction up to this point in time have been force-written to disk.**

2. **The commit operation of a transaction cannot be completed until all the Redo-type and Undo-type log recodes for that transaction have been force-written to disk.**

Steal/No-Steal - Force/No-Force

No-Steal approach:

- **In No-Steal approach a cache page updated by a transaction cannot be written to disk before the transaction commits.**
- **The pin-unpin bit is used to realize the approach.**
- **In Steal approach a cache page updated by a transaction can be written to disk before the transaction commits.**

Steal/No-Steal - Force/No-Force (cont)

Force/No-Force approach:

- **In Force approach all cache pages updated by a transaction are immediately written to disk when the transaction commits.**
- **Typical DB systems use a Steal/No-Force strategy because of:**
  - **Steal approach avoids the need for a very large buffer space to store all updated pages in memory.**
  - **No-Force approach provides the advantage of keeping an updated page of a committed transaction in the memory when another transaction needs to update it, thus eliminates the I/O cost of reading that page again from disk.**

Checkpoints

- **Checkpoint record is written into the log periodically at that point when the system writes out to the DB on disk all DBMS buffers that have been modified.**
- **All transactions committed (have [c,T] entries in the log) before a [checkpoint] entry do not need to redo their write operations.**
- **The interval to take a checkpoint may be measured in time or number of committed transactions.**

Checkpoints (cont)

Taking checkpoint consists of the following actions:

1. **Suspend execution of transactions temporarily.**
2. **Force-write all main memory buffers that have been modified to disk.**
3. **Write a [checkpoint] record to the log, and force-write the log to disk.**
4. **Resume executing transactions.**

Checkpoints (cont)

- **The time needed to force-write buffers to disk may delay transactions execution.**
- **Fuzzy checkpoint is used to reduce the delay by resuming the execution of the transactions after the checkpoint record is written to the log without having to wait for force-write step to be completed.**
- **The previous checkpoint is valid until the force-write step is completed (the system has a pointer to the valid checkpoint and changes it to the new checkpoint after the success of the force-write step).**

Transaction Rollback

- **Rollback is required if a transaction fails.**
- **The write operations of rollbacked transactions are undone.**
- **Cascading rollback is possible if the schedule is not cascadless or strict.**
- **The undo-type log entries are used to restore the old values of the rolled back data items.**
- **The write operations of the log are the only operations to undo.**
- **Read operations are used only in checking cascading rollback if it may occur.**

Transaction Rollback (cont)

- Ex:

T1
read-item(A)
read-item(D)
write-item(D)

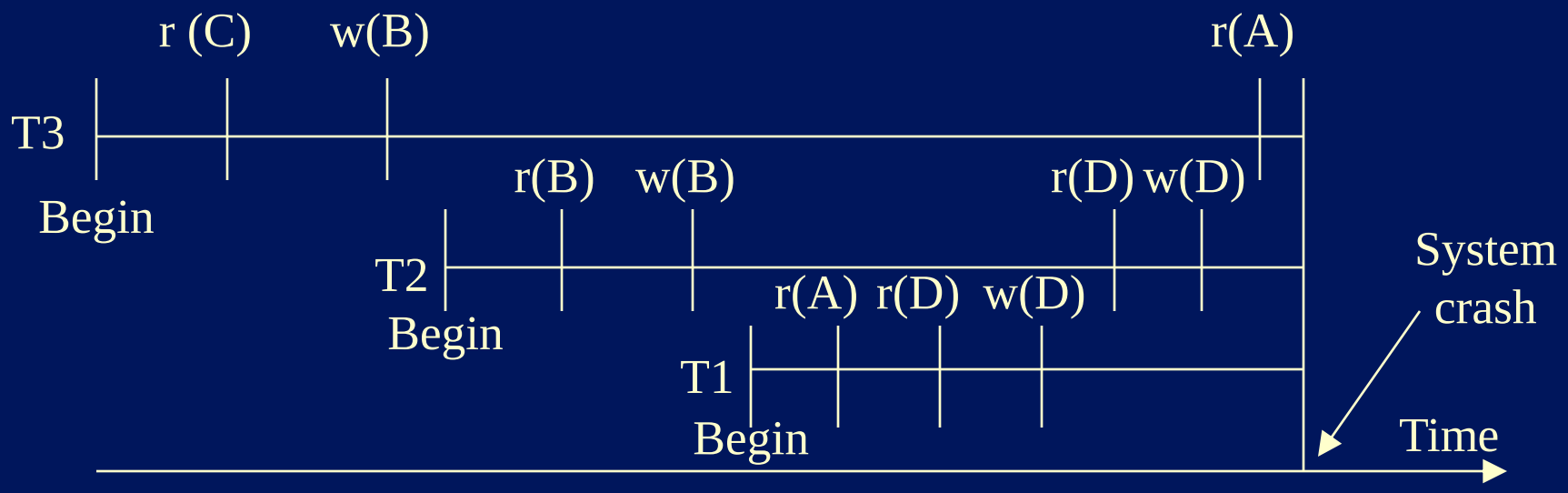
T2
read-item(B)
write-item(B)
read-item(D)
write-item(D)

T3
read-item(C)
write-item(B)
read-item(A)
write-item(A)

Log:	A	B	C	D
	30	15	40	20
[Start,T3]				
[r,T3,C]				
[w,T3,B,15,12]		12		
[Start,T2]				
[r,T2,B]				
[w,T2,B,12,18]		18		

	A	B	C	D
[Start,T1]				
[r,T1,A]				
[r,T1,D]				
[w,T1,D,20,25]				25
[r,T2,D]				
[w,T2,D,25,26]				26
[r,T3,A]				

Transaction Rollback (cont)



Notes:

1. T3 is rolled back because it did not reach its commit point.
2. T2 is rolled back because it reads “B” which has been written by T3.
3. T1 will not be rolled back.
4. The write operations of T2 and T3 are undone.
5. Item “D” is restored to its old value “25”, item “B” is restored to its value “12” then finally to “15”.

Recovery Techniques Based on Deferred Update

- **The idea is to postpone any actual updates to the DB until the transaction completes its execution and commits (it follows the No-Steal approach).**
- **The updates are recorded only in the log and in the buffer.**
- **After transaction reaches its commit point and the log is force-written to disk, the updates are recorded in the disk.**
- **If the transaction fails before commit, no need to undo any operations.**

Recovery Techniques Based on Deferred Update (cont)

- **It is a simple recovery technique, but it may not be practical unless the transactions are short and change few items.**
- **The typical deferred update protocol can be described as follows:**
  1. **A transaction cannot change the DB on disk until it reaches its commit point.**
  2. **A transaction does not reach its commit point until its update operations are recorded in the log and the log is force-written to disk (WAL protocol).**
- **The technique is known as No-Undo/Redo recovery algorithm (redo for committed transactions before failure and all its updates are recorded in the log and in the buffer).**

Recovery Using Deferred Update in a Single-User Environment

- **The technique is simple because we have only one active transaction (no concurrency).**
- **<u>The algorithm RDU_S:</u>**
  1. **Use a list for committed transaction and another for active transaction since the last checkpoint.**
  2. **Apply the REDO operation to all write-item of committed transactions in the order of recording them in log.**
  3. **Restart the active transaction.**

  – **<u>REDO(write-op):</u>**
  1. **Examine the log entry of the operation to get the item and its new value (AFIM).**
  2. **Set the value of the item in the DB to its new value (AFIM).**

Deferred Update (cont)

- **The REDO operation is idempotent (executing it many times is equivalent to executing it just once).**
- **Thus, if a failure occurs during recovery, the process is started again and will be correct.**

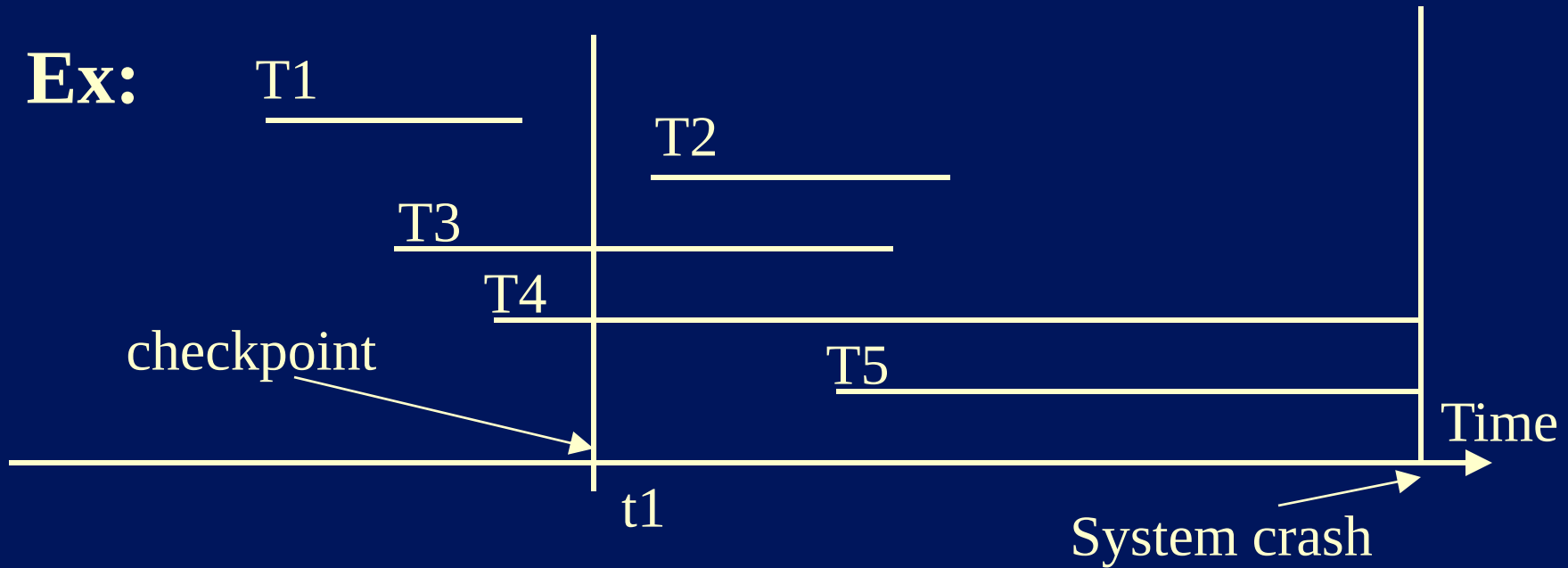
Recovery Using Deferred Update in a Multi-User Environment

- **The techniques may be more complex depending on the used concurrency control.**
- **Assuming strict 2PL concurrency protocol and a checkpoint are used, the recovery technique may be as:**

RDU_M(with checkpoint):

1. **Use a list for the committed transactions and another for the active transactions since the last checkpoint.**
2. **Redo the operations of the committed transactions according to their order in the log.**
3. **Cancel the active the transactions and resubmit them again.**

Deferred Update in a Multi-User Environment (cont)



Notes:

- **T1 is not redone because it commits before the checkpoint.**
- **T2 and T3 are redone because they commit after the checkpoint.**
- **T4 and T5 are ignored (canceled) and restarted again.**

Deferred Update in a Multi-User Environment (cont)

Notes:

- **The No-Undo/Redo algorithms can be enhanced by scanning the log in reverse order and ignore the write operations of any item already scanned.**
- **The techniques suffer from these drawbacks:**
  - **It limits the degree of concurrency (strict 2PL).**
  - **It requires excessive buffer space to hold all updates until the transaction commits.**
- **The techniques have the following advantages:**
  - **No updates are recorded on disk until commit, so no rollback is needed.**
  - **No cascading rollback is allowed.**

Transaction Actions That Do Not Affect The DB

- **Some actions as generating reports or messages are using the DB but do not update it.**
- **If these transactions fail, it is required to notify the user that he got uncompleted and wrong output.**
- **A common solution for these cases is to issue the transactions as batch jobs, which are executed only if the transaction reaches its commit point. The jobs are canceled if the transaction fails.**

Recovery Techniques Bases on Immediate Update

- **In these techniques the DB can be immediately updated by the transactions (the updates are also recorded in the log for recovery using WAL).**
- **When a transaction fails, undo must be used for rollback and redo must be used if we allow a transaction to commit before all its changes are written to the DB (Undo/Redo algorithms).**
- **If the recovery technique ensures that all updates of a transaction are recorded in the DB before the transaction commits, the Undo/No-Redo algorithms are used.**

Recovery Using Immediate Update in a Single-User Environment

RIU_S algorithm:

1. **Use a list for the committed transactions and another for the active transaction since the last checkpoint.**
2. **Undo all the write operations of the active transaction using the UNDO procedure described below.**
3. **Redo the write operations of the committed transactions in the order in which they were written in the log using the REDO procedure described earlier (during discussing RDU_S).**

Recovery Using Immediate Update in a Single-User Environment(cont)

Procedure UNDO(write-op):

1. **Examine the log entry of the operation to get the item and its old value.**
2. **Set the value of the item in the DB to its old value.**

[The log must be processed in the reverse order]

Recovery Using Immediate Update in a Multi-User Environment

- **The process depends on the used concurrency control protocol.**
- **Assuming strict 2PL and checkpoint, the technique will be as:**

Procedure RIU_M:

1. **Use a list for the committed transactions and another for the active transaction since the last checkpoint.**
2. **Undo all write operations of the active transactions using the UNDO procedure (undo in reverse order).**
3. **Redo all write operations of committed transactions in their log order (this step can be enhanced as RDU_M)**

Recovery in Multidatabase Systems

- **If a transaction accesses multiple DB, it is called a multidatabase transaction.**
- **To maintain the multidatabase transactions, it necessary to use a *Global Recovery Manager* (coordinator) besides the local recovery control.**
- **The coordinator usually follows a protocol called the two-phase commit protocol.**

Recovery in MDB Systems (cont)

Two-phase commit protocol:

Phase 1:

1. **When all participating DBs signal the coordinator that the part of the MDB transaction has concluded, the coordinator sends a message “prepare for commit” to each participant to get ready for committing.**

2. **Each participant receiving the message will force-write all log records and needed information for local recovery to disk and send “ok” message to the coordinator or”not ok” if it fails.**

3. **If the coordinator does not receive a reply from the DB within a certain time out interval, it assumes a “not ok” response.**

Recovery in MDB Systems (cont)

Phase 2:

1. **If all participating DBs reply “ok” and also the coordinator:**
   - **A “commit” signal is sent to all participating DB.**
   - **Each participating DB writes a commit record in the log and permanently updating the DB if needed.**
2. **If one or more participating DBs or the coordinator sends “not ok” message:**
   - **A “rollback” message is sent to each participating DB.**
   - **Each participating DB executes a rollback for the local of the transaction on its site.**

DB Backup and Recovery From Catastrophic Failures

- **Backup is used to handle the catastrophic failures.**
- **The whole DB and the log are periodically copied onto a cheap media as tapes.**
- **It is usual to backup the log more frequent than the full backup because the log is small and can be used to recover from failure by using the last backup and the log.**
- **A new log is started after each DB backup.**