Shortening Recovery

Avoiding Unneeded UNDOs and REDOs

Checkpoints

- Problem: Don’t know state of disk
  - so each recovery starts from last backup
  - same undos and redos over and over
- Solution: Add information to log
  - Information about what is actually on the disk
- Checkpoint may include list of active transactions
Version 1: Commit Consistent Checkpoint

- No new transactions enter DB
- Active transactions complete
  - commit or abort
  - Buffers have only committed data
- Forcewrite the Log
- Forcewrite buffers
  - disk reflects all committed transactions
- Forcewrite CHECKPOINT to log.
Using Commit Consistent Checkpoint

• No information before checkpoint is needed
• Rollback thru log till checkpoint found
• then Roll Forward

Commit Consistent Checkpoint is SLOW

• Wait for transactions to complete
• Wait for all buffers to get to disk
• No new transaction can use DB during checkpointing
Version 2: Cache Consistent Checkpoint

- Avoid wait for transactions to complete
- Suspend all transactions
- Write buffers (cache) to disk
  - Uncommitted data may now be on disk
- Write CHECKPOINT with list of active transactions to log
- Write log to disk
  - Log reflects disk

Cache Consistent Checkpoint

Transactions suspended. No Longer terminate.
Using Cache Consistent Checkpoint

• Committed Transactions before Checkpoint don’t need to be REDONE
• Uncommitted Transactions before Checkpoint DO need to be UNDONE
  – if active at checkpoint

Using Cache Consistent Checkpoint

• Rollback
  – back to last checkpoint as before
    note commits, undo uncommitted
  – then back until uncommitted transactions on active list of checkpoint are all rolled back
• Rollforward
  – redo as before but only from checkpoint forward
Cache Consistent Example

T1

T2

T3

T4

CHKPT

CHKPT

CRASH

OK -- all data written at CHKPT

Undo to start of T2

Find T3 in CHKPT active list and undo to start

Redo from CHKPT to commit

Version 3: Fuzzy Checkpointing

• Reduce delay of forcewriting cache.
• Suspend all transactions
• Write CHECKPOINT with list of active transactions to log
• Write log to disk
  – Log reflects cache, not disk
• Write buffers (cache) to disk BEFORE NEXT CHECKPOINT
Fuzzy Checkpoint

Transactions suspended. No Longer terminate.

This not so much

Promised by next ChkPt

Using Fuzzy Checkpoint

• Basically Like Cache Consistent Checkpoint
• If buffers from last checkpoint were not all written before the crash ...
• Use the previous checkpoint
Fuzzy Checkpoint Example

chkpt 1

T1

chkpt 2

T2

T3

chkpt 3

T4

CRASH

T1

commit

OK -- all data written by chkpt 3

Undo to start of T2

Find T3 in chkpt 3/2 active list & undo to start

commit

Redo from chkpt 2 to commit

promised here

done here

8/6/2015