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.
Commit Consistent Checkpoint
Commit Consistent Checkpoint
Commit Consistent Checkpoint
Commit Consistent Checkpoint
Commit Consistent Checkpoint

This takes a while
Commit Consistent Checkpoint

This takes a while
This takes a while
This takes a while

This not so much
Commit Consistent Checkpoint

This takes a while

This not so much
Commit Consistent Checkpoint

This takes a while

This not so much
This takes a while

A long, long time

This not so much
Commit Consistent Checkpoint

This takes a while

A long, long time

This not so much
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
Cache Consistent Checkpoint
Cache Consistent Checkpoint

Transactions suspended. No Longer terminate.
Transactions suspended. No Longer terminate.
Cache Consistent Checkpoint

Transactions suspended. No Longer terminate.

This not so much
Cache Consistent Checkpoint

Transactions suspended. No Longer terminate.

This not so much
Cache Consistent Checkpoint

Transactions suspended. No Longer terminate.

This not so much
Cache Consistent Checkpoint

Transactions suspended. No Longer terminate.

Still a long, long time

This not so much
Transactions suspended. No Longer terminate.

Still a long, long time

This not so much
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

CHKPT

CHKPT

CRASH
Cache Consistent Example

T1 → commit

CHKPT → CHKPT → CRASH
Cache Consistent Example

T1

CHKPT

commit

CHKPT

OK -- all data written at CHKPT

CRASH
Cache Consistent Example

T1 → CHKPT

T2 → CHKPT → CRASH

commit

OK -- all data written at CHKPT
Cache Consistent Example

T1 -> CHKPT

commit

OK -- all data written at CHKPT

T2 -> CHKPT

Undo to start of T2

CRASH
Cache Consistent Example

T1

CHKPT

T2

T3

CHKPT

CRASH

commit OK -- all data written at CHKPT

Undo to start of T2
Cache Consistent Example

T1

T2

T3

commit

OK -- all data written at CHKPT

Undo to start of T2

Find T3 in CHKPT active list and undo to start
Cache Consistent Example

T1

CHKPT

T2

commit OK -- all data written at CHKPT

T3

CHKPT

Undo to start of T2

T4

Find T3 in CHKPT active list and undo to start

CRASH

commit
Cache Consistent Example

T1

T2

T3

T4

CHKPT

CHKPT

CRASH

commit

OK -- all data written at CHKPT

Undo to start of T2

Find T3 in CHKPT active list and undo to start

commit

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

Data Buffer

Data Buffer

Data Buffer

Data Disk

Log Buffer

Log Disk
Fuzzy Checkpoint
Fuzzy Checkpoint
Transactions suspended. No Longer terminate.
Fuzzy Checkpoint

Transactions suspended. No Longer terminate.
Transactions suspended. No Longer terminate.

This not so much
Transactions suspended. No Longer terminate.

This not so much
Transactions suspended. No Longer terminate.

This not so much
Transactions suspended. No Longer terminate.

This not so much
Transactions suspended. No Longer terminate.

Promised by next ChkPt

This not so much
Fuzzy Checkpoint

Transactions suspended. No Longer terminate.

Promised by next ChkPt

This not so much
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  CHKPT 2  CHKPT 3  CRASH
Fuzzy Checkpoint Example

CHKPT 1  CHKPT 2  CHKPT 3  CRASH

T1  commit
promised here

T1

CHKPT 1  CHKPT 2  CHKPT 3  CRASH

commit
T1 promised here

CHKPT 1

CHKPT 2

CHKPT 3

CRASH

commit

OK -- all data written by CHKPT 3
promised here

completed here

OK -- all data written by CHKPT 3

commit
OK -- all data written by CHKPT 3

Undo to start of T2

promised here completed here

CHKPT 1 CHKPT 2 CHKPT 3 CRASH

T1 T2
OK -- all data written by CHKPT 3

promised here
completed here

commit

Undo to start of T2
OK -- all data written by CHKPT 3

Undo to start of T2
Find T3 in CHKPT 3/2 active list & undo to start

promised here
completed here
OK -- all data written by CHKPT 3

Undo to start of T2

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

promised here

completed here

CHKPT 1

CHKPT 2

CHKPT 3

CRASH

T1

T2

T3

T4
T1

CHKPT 1

commit

OK -- all data written by CHKPT 3

T2

CHKPT 2

Undo to start of T2

T3

CHKPT 3

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

T4

CRASH
CHKPT 1  CHKPT 2  CHKPT 3  CRASH

T1

T2

T3

T4

commit

promised here

OK -- all data written by CHKPT 3

Undo to start of T2

Find T3 in CHKPT 3/2 active list & undo to start
T1

T2

T3

T4

CHKPT 1

CHKPT 2

CHKPT 3

CRASH

promised here

completed here

commit

OK -- all data written by CHKPT 3

Undo to start of T2

Find T3 in CHKPT 3/2 active list & undo to start
OK -- all data written by CHKPT 3

Undo to start of T2

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

promised here
completed here
CHKPT 1

T1

CHKPT 2

T2

CHKPT 3

T3

CRASH

T4

promised here

completed here

OK -- all data written by CHKPT 3

Undo to start of T2

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

Redo from CHKPT 2 to commit