

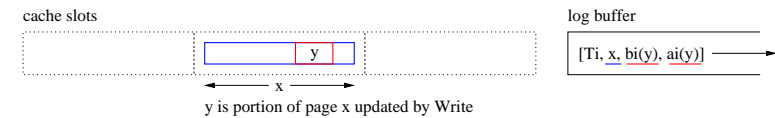
CA420 Databases II

Centralized Recovery, Part II

Partial Data-Item Logging

Log the before image and after image of only the portion of an item x affected by an update (and offset)

Generally requires less space than logging an entire page



Towards A More Realistic Undo/Redo Recovery Algorithm

Checkpointing:

- limiting the portion of the log and the stable database that must be scanned/recovered during restart

Granularity:

- support logging at a finer granularity than the page
- hence also: support locking and updates at a finer granularity than the page

Recovery Information and the Cache – Stable LSN

For each cache slot, its “Stable LSN” is:

- the LSN of the last record in the log buffer at the time this page was last fetched or flushed

Consider:

- LSN_1 , some LSN, and
- LSN_s , the stable LSN associate with some cache slot

If $LSN_1 < LSN_s$:

- then any update reflected in the cache at time LSN_1 must now have been propagated to stable storage

Otherwise:

- this page may have been dirty since before LSN_1

Recovery Information and the Cache – Cache LSN

For each cache slot, its “Cache LSN” is:

- the LSN of the log record generated by the last write to this page

If that log record has been flushed to the stable log:

- then the cache manager can safely overwrite this page in the stable database with uncommitted data
- thus ensuring that the Undo Rule is satisfied

Checkpointing

Unless care is taken, Restart may have to scan entire log and/or stable database

Checkpointing:

- perform a little extra work during normal processing and thereby reduce the amount of work that must be done during Restart

Three variants:

- commit consistent checkpointing
- cache consistent checkpointing
- fuzzy checkpointing

Extended Log Records – Updates, Commits and Aborts

Updates:

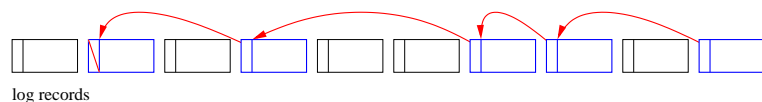
- name of the transaction, name of the item written
- offset and length of portion of the item written
- before image of that portion
- after image of that portion
- LSN of previous update record of this transaction (or null)

Commit:

- name of committing transaction
- LSN of previous update record of this transaction (or null)

Abort:

- name of aborting transaction
- LSN of previous update record of this transaction (or null)



Commit-Consistent Checkpointing

Method:

- stop accepting new *transactions*
wait for active transactions to complete
- flush all dirty cache slots
- write Checkpoint record to log buffer
- resume normal processing

Restart:

- all updates from log records prior to the most recent checkpoint record on the log are already propagated into the stable database

Cache-Consistent Checkpointing

Method:

- stop accepting new *operations*
wait for active operations to complete
- flush all dirty cache slots
- write Checkpoint record to log buffer containing a list of active transactions
- resume normal processing

Advantage over commit consistent checkpointing:

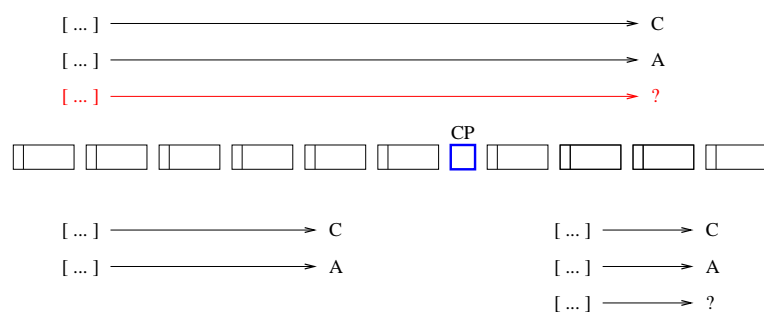
- need only wait for active operations, not transactions, to complete

Fuzzy Checkpointing

Method:

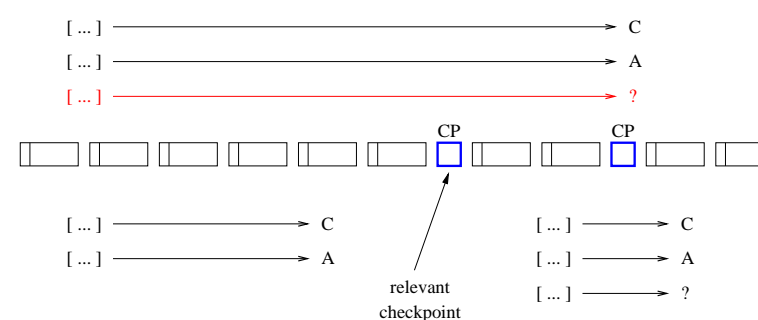
1. stop from accepting new operations
wait for active operations to complete
2. flush each dirty cache slot that has not been flushed since the *previous* checkpoint:
 - that is:
those whose Stable-LSN is less than that of the previous checkpoint
3. create a checkpoint record containing:
 - list of active transactions
4. acknowledge checkpoint completion:
 - RM begins accepting new operations again

Case Analysis



The list of active transactions is necessary to identify cases such as that marked in **red** in the figure

Case Analysis



Case analysis is as previously, but now with respect to the *penultimate* checkpoint record on the log

Recovery Manager Operations

RM-Read, RM-Write, and RM-Commit:

- as previously

RM-Abort:

- scan *backwards* through the chain of update records pertaining to this transaction
- for each update:
 - reinstall the before image

Note:

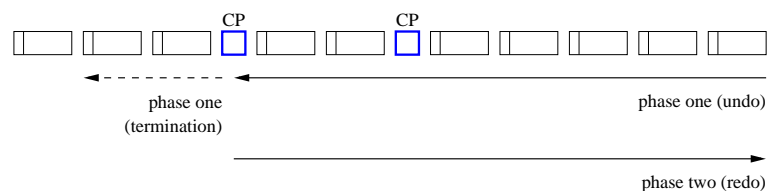
a backwards scan is required because the same item may have been updated more than once, and because ranges may overlap

Restart – Phase One, the Undo Phase

Restart – phase one, the *undo* phase:

1. (commit list) $CL = \{\}$; (abort list) $AL = \{\}$
2. scan backwards through the log:
 - (a) for commit records, add T to CL
 - (b) for abort records, add T_i to AL
 - (c) for update records $[T_i, x, \text{off}, \text{len}, \text{bi}, \text{ai}, \text{prevLSN}]$:
 - i. if T_i is on CL :
skip this record
 - ii. if T_i is on neither CL nor AL :
add T_i to AL
 - iii. if T_i is *now* in AL :
 - A. fetch x , if not already in cache
 - B. restore before image in relevant cache slot
 - C. if this update record has no predecessor:
remove T_i from AL

Restart – Log Scans



Restart – Phase One, Termination

Restart – phase one, termination of the undo phase:

- ignore first checkpoint record
- at the penultimate (next) checkpoint record:
 - consider each transaction listed as active:
 - if it is on AL :
skip it
 - if it is on CL :
skip it
 - if it is on neither AL nor CL :
add it to AL

(these are exactly the transaction that were active at the time of checkpoint, but failed to commit or abort)
- continue backwards scan using only rule 2(c)-iii of previous slide:
- stop when the AL is empty

Restart – Phase Two, the Redo Phase

Restart – phase two, the redo phase:

1. scan *forwards* through the log from the penultimate checkpoint
2. for each update record $[T_i, x, \text{off}, \text{len}, \text{bi}, \text{ai}, \text{prevLSN}]$:
 - if T_i is on CL , then:
 - (a) fetch x , if not already in cache
 - (b) reinstall after image in relevant cache slot
3. stop when end of log is reached

Correctness – Continued

Cases 1 and 2:

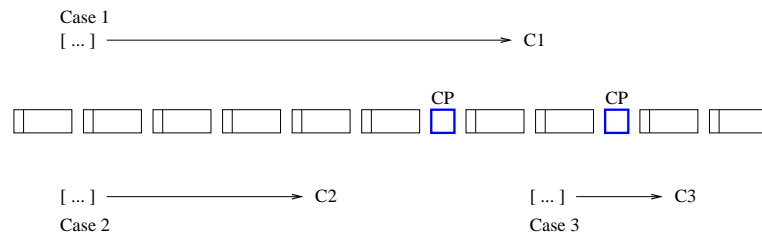
- updated value installed in stable database at the latest by time of final checkpoint

Case 3:

- updated value reinstalled in stable database during phase two (redo) of Restart

Correctness

Consider the *last committed update* to each individual bit:



Correctness – Continued

Consider also intervening operations:



Note: by strictness, only these and similar combinations are possible