Database Systems
(資料庫系統)

January 5, 2010

Happy New Year

1

Announcement

- Last assignment is due today.
- Final exam covers chapters [8-13,14.4,16,17.1~17.4]
  - How about Chapter 18?
- Place: CSIE 101/103
  - Closed book

2

Crash Recovery

Chapter 18

3

Overview

- Many types of failures:
  - Transaction failure: bad input, data not found, etc.
  - System crash: bugs in OS, DBMS, loss of power, etc.
  - Disk failure: disk head crash
- Recovery manager is called after a system crash to restore DBMS to a consistent state before the crash.
  - Ensure Two transaction properties:
    - Atomicity: undo all actions of uncommitted transactions.
    - Durability: actions of committed transactions survives failures. (redo their update actions if they have not been written to disks).
- ARIES: log-based recovery algorithm.

4

ARIES Overview

- Assume HW support:
  - Log actions on an independent "crash-safe" storage
- What are SW problems?
  - Results of uncommitted transactions may be written to disks → undo them
  - Results of committed transactions may not be written to the disk → redo them
- Can you think about a solution?
- What are the key questions/issues to be addressed?
  - What are the states of transactions (committed vs. uncommitted) at the time of crash?
  - What are the states of page (dirty?) at the time of the crash?
  - Where to start undo & redo?

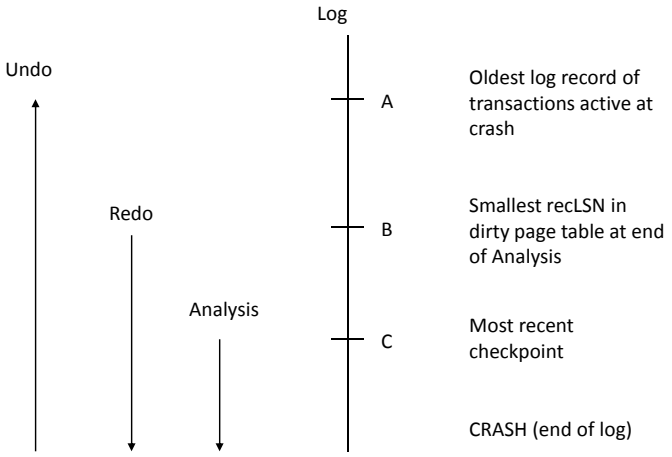
5

ARIES General Approach

- Before crash:
  - Log changes to DB (WAL)
  - Checkpoints
- After crash:
  - Analysis phase
    - Figure out states of [committed vs. uncommitted] transactions & pages [dirty or clean]
  - Redo phase
    - Repeat actions from uncommitted & committed transactions [till the time of the crash]
  - Undo phase
    - Undo actions of uncommitted transactions
- Do we really need "redo" & "undo"? Under what condition they are not needed?
  - Page replacement in buffer pool, e.g., allow only committed transactions to update data in disks.

6

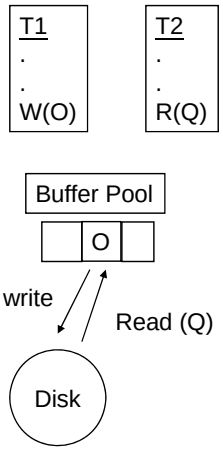
Three Phases of ARIES



7

Steal Policy

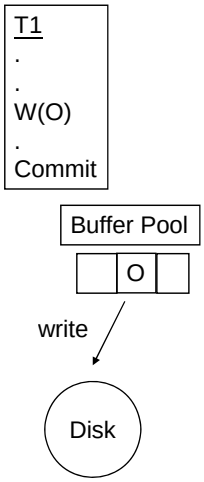
- ARIES is designed to work with a steal, no-force approach.
  - Related to page replacement policies
- Steal property:
  - Can the changes made to an object O in the buffer pool by T1 be written to disk before T1 commits?
  - If yes, we have a steal (T2 steals a frame from T1).
  - Say T2 wants to bring a new page (Q), and buffer pool replace the frame containing O.



8

Force Policy

- When T1 commits, do we ensure that all changes T1 has made are immediately forced to disk?
- If yes, we have a force approach.



9

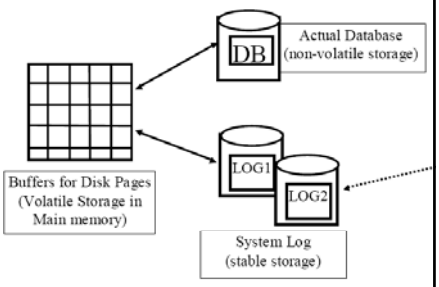
Steal, No-Force Write Policies

- ARIES can recover crashes from DB with steal & no-force write policy:
  - Modified pages may be written to disk before a transaction commits.
  - Modified pages may not be written to disk after a transaction commits.
- "No-steal & Force write policy" makes recovery really easy, but the tradeoff is low DB performance.
  - Why?
  - Adding constraints to an optimal buffer replacement.

10

ARIES

- ARIES is a recovery algorithm that can work with a steal, no-force write policy.
- ARIES is invoked after a crash. This process is called restart.
- ARIES maintains a history of actions executed by DBMS in a log.
  - The log is stored on stable storage and must survive crashes. (Use RAID-1 Mirrored)



11

Three Phases in ARIES (1)

- Goal:
  - Restore the DB to the state (buffer pool & disk) before the crash
  - AND without effects from actions of active (uncommitted) transactions.
- Analysis Phase:
  - Identify active transaction at the time of crash
    - T1 and T3.
  - Identify dirty pages in the buffer pool:
    - P1, P3 and P5.

LSN	LOG
10	Update: T1 writes P5
20	Update: T2 writes P3
30	T2 commits
40	T2 ends
50	Update: T3 writes P1
60	Update: T3 writes P3
CRASH, RESTART	

12

Three Phases in ARIES (1)

- Redo Phase:
  - Repeat actions (active & committed), starting from a redo point in the log, and restores the database state to what it was at the time of crash.
  - Where is the redo point?

LSN	LOG
10	Update: T1 writes P5
20	Update: T2 writes P3
30	T2 commits
40	T2 ends
50	Update: T3 writes P1
60	Update: T3 writes P3
CRASH, RESTART	

13

Three Phases in ARIES (2)

- Undo Phase:
  - Undo actions of active transactions in reverse-time order, so that DB reflects only the actions of committed transactions.
    - LSNs 70 -> 60 -> 50 -> 10
- Why in reverse-time order?
  - Consider forward-time order. P5 will be restored to a value written by T1 (in LSN #10), rather than before it.

LSN	LOG
10	Update: T1 writes P5
20	Update: T2 writes P3
30	T2 commits
40	T2 ends
50	Update: T3 writes P1
60	Update: T3 writes P3
70	Update: T3 writes P5
CRASH, RESTART	

14

Three Principles of ARIES

- Write-Ahead Logging (WAL)
  - Update to a DB object is first recorded in the log.
  - The log record must be forced to a stable storage before the writing the DB object to disk.
- How is WAL different from Force-Write?
  - Forcing the log vs. data to disk.
- Repeating History During Redo
  - On restart, redo the actions (recorded in the log) to bring the system back to the exact state at the time of crash. Then undo the actions of active (not committed) transactions.
- Logging Changes During Undo
  - Since undo may change DB, log these changes (and don't repeat them).

15

How is ARIES explained?

- Describe the needed data structure for recovery
  - WAL
  - Data page
  - Transaction table
  - Dirty page table
  - Checkpoints
- Describe the algorithm
  - no crash during recovery
  - crash during recovery

16

Log Structure

- Log contains history of actions executed by the DBMS.
- A DB action is recorded in a log record.
- Log Tail: most recent portion of the log in main memory.
  - It is periodically forced to stable storage.
  - Aren't all records in a log in stable storage? No, only when writes to disk or commits.
- Log Sequence Number (LSN): unique ID for each log record.

LSN	LOG
10	Update: T1 writes P5
20	Update: T2 writes P3
30	T2 commits
40	T2 ends
50	Update: T3 writes P1
60	Update: T3 writes P3

17

Data Page

- PageLSN: the LSN of the most recent log record that made a change to this page.
  - Every page in the DB must have a pageLSN.
  - What is P3's pageLSN?
    - 60 or 20
  - It is used in the Redo phase of the algorithm.

LSN	LOG
10	Update: T1 writes P5
20	Update: T2 writes P3
30	T2 commits
40	T2 ends
50	Update: T3 writes P1
60	Update: T3 writes P3

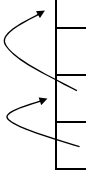
18

What Actions to Record Log?

- A log is written for each of the following actions:
  - Updating a page: when a transaction writes a DB object, it write an update type record. It also updates pageLSN to this record's LSN.
  - Commit: when a transaction commits, it force-writes a commit type log record to stable storage.
  - Abort: when a transaction is aborted, it writes an abort type log record.
  - End: when a transaction is either aborted or committed, it writes an end-type log record.
  - Undoing an update: when a transaction is rolled back (being aborted, or crash recovery), it undoes the updates and it writes a compensation log record (CLR).

19

Log Record

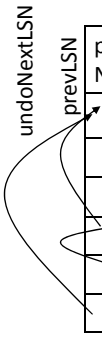


prevLSN	transID	Type	PageID / Length	Offset	Before-image	After-image
Fields common to all log records			Additional fields for update log records			
	T1	update	P500 / 3	21	ABC	DEF
	T2	update	P600 / 3	41	HIJ	KLM
	T1	update	P500 / 3	20	GDE	QRS
	T1	update	P505 / 3	21	TUV	WXY

PrevLSN: LSN of the previous log record in the same transaction. It forms a single linked list of log records going back in time. It will be used for recovery.
Type: update, commit, abort, end, CLR

20

Compensation Log Record (CLR)



prevLSN	LSN	transID	Type	PageID / Length	Offset	Before-image	After-image
	00	T1	update	P500 / 3	21	ABC	DEF
	10	T2	update	P600 / 3	41	HIJ	KLM
	20	T2	update	P500 / 3	20	GDE	QRS
	30	T1	update	P505 / 3	21	TUV	WXY
	40	T1	abort				
	50	T1	CLR / undo 30	P505 / 3	21	TUV	

CLR is written when undoing an update (T1 30) after an abort (or during crash recovery).
CLR records undoNextLSN, which is the LSN of the next log record that is to be undone for T1, which is the prevLSN of log record #30.
undoNextLSN is used for undoing actions in the reverse order.

21

More on CLR

- If a system crash occurs during or after recovery, do we need to undo CLR actions?
  - Why not?
  - Uncommitted T write actions -> Undo
- This bounds the number of CLRs in the log during restart (even if you have repeated crashes).
  - At most one CLR log record per update log record

22

Crash after Abort Example: No Need to Undo An Undo Action

prevLSN	LSN	transID	Type	PageID / Length	Offset	Before-image	After-image
-	00	T1	update	P500 / 3	21	ABC	DEF
00	10	T1	update	P505 / 3	21	TUV	WXY
System Crash							
	20	T1	CLR/undo 10	P505 / 3	21	TUV	
System Crash							
	30	T1	CLR/undo 00	P500 / 3	21	ABC	

Redo 500, continue to undo 20 & 10 & 00

23

Other Recovery-Related Structures

- Transaction Table: one entry for each active (uncommitted) transaction. Each entry has
  - Transaction ID
  - lastLSN: the last LSN log record for this transaction.
  - How is it used? (In Undo)

LSN	Trans ID	Type	PageID / Length
00	T1	update	P500 / 3
10	T2	update	P600 / 3
20	T2	update	P500 / 3
30	T1	update	P505 / 3

transID	lastLSN
T1	30
T2	20

Transaction Table

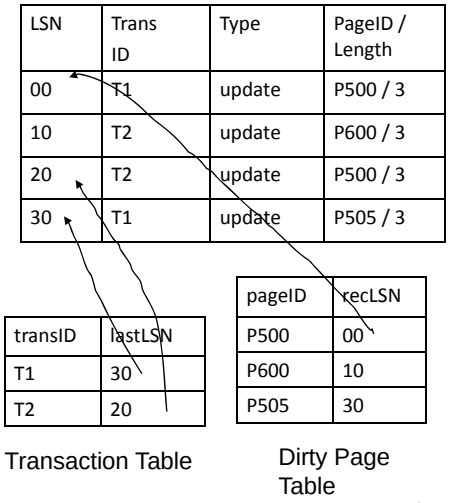
pageID	recLSN
P500	00
P600	10
P505	30

Dirty Page Table

24

Other Recovery-Related Structures

- Dirty Page Table: one entry for each dirty page (not written to disk) in the buffer pool.
  - recLSN: LSN of the first log record that caused this page to become dirty.
  - How is it used? (In Redo)



LSN	Trans ID	Type	PageID / Length
00	T1	update	P500 / 3
10	T2	update	P600 / 3
20	T2	update	P500 / 3
30	T1	update	P505 / 3

transID	lastLSN
T1	30
T2	20

Transaction Table

pageID	recLSN
P500	00
P600	10
P505	30

Dirty Page Table

25

Write-Ahead Log (WAL)

- Before writing a page (P) to disk, every update log record that describes a change to P must be forced to stable storage.
- A committed transaction forces its log records (including the commit record) to stable storage.
- (Non-forced approach + WAL) vs. (Forced approach) at Transaction Commit Time:
  - Non-forced approach + WAL mean log records are written to stable storage, but not data records.
  - Forced approach means data pages are written to disk.
  - Log records are smaller than data pages!

26

Checkpointing

- A checkpoint is a snapshot of DBMS state stored in stable storage.
- Checkpointing in ARIES has three steps:
  (1) write begin_checkpoint record to log
  (2) write the state of transaction table and dirty page table + end_checkpoint record to log
  (3) write a special master record containing LSN of begin_checkpoint log record.
- Why checkpointing?
  - The restart process will look for the most recent checkpoint & start analysis from there.
  - Shorten the recovery time -> take frequent checkpoints.

27

Recovering from a System Crash

- Recovering will use WAL & the most recent checkpoint
  - Write-ahead log
    - The most recent checkpoint
    - Compensation Log Records
      - undoNextLSN: the LSN of the next log record that is to be undone
  - Transaction table
    - active (not committed) transactions
    - lastLSNs: the LSN of the most recent log record for this transaction. (analysis)
    - Used for undo
  - Dirty page table
    - dirty (not written to disk) pages
    - recLSNs: LSN of the first log record that caused this page to become dirty
    - Used for redo

28

Analysis Phase

- Determine three things:
  - A point in the log to start REDO.
    - Earliest update log that may not have been written to disk.
  - Dirty pages in the buffer pool at the time of crash -> restore the dirty page table to the time of crash.
  - Active transactions at time of crash for UNDO -> restore the transaction table to the time of crash.

29

Analysis Phase: Algorithm

1. Find the most recent begin_checkpoint log record.
2. Initialize transaction & dirty page tables from the ones saved in the most recent checkpoint.
3. Scan forward the records from begin_checkpoint log record to the end of the log. For each log record LSN, update trans_tab and dirty_page_tab as follows:
   - If we see an end log record for T, remove T from trans_tab.
   - If we see a log record for T' not in trans_tab, add T' in trans_tab. If T' is in the trans_tab, then set T's lastLSN field to LSN.
   - If we see an update/CLR log record for page P and P is not in the dirty page table, add P in dirty page table and set its recLSN to LSN.

30

Analysis Phase: Example (1)

- After system crash, both table are lost.
- No previous checkpointing, initialize tables to empty.

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN

Transaction Table

pageID	recLSN

Dirty Page Table

31

Analysis Phase: Example (2)

- Scanning log 00:
  - Add T1 to transaction table.
  - Add P500 to dirty page table.

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN
T1	00

Transaction Table

pageID	recLSN
P500	00

Dirty Page Table

32

Analysis Phase: Example (3)

- Scanning log 10:
  - Add T2 to transaction table.
  - Add P600 to dirty page table.

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN
T1	00
T2	10

Transaction Table

pageID	recLSN
P500	00
P600	10

Dirty Page Table

33

Analysis Phase: Example (4)

- Scanning log 20:
  - Set lastLSN to 20

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN
T1	00
T2	20

Transaction Table

pageID	recLSN
P500	00
P600	10

Dirty Page Table

34

Analysis Phase: Example (5)

- Scanning log 30:
  - Add P505 to dirty page table.

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN
T1	30
T2	20

Transaction Table

pageID	recLSN
P500	00
P600	10
P505	30

Dirty Page Table

35

Analysis Phase: Example (6)

- Scanning log 40:
  - Remove T2 from transaction table.
  - We are done!
- The redo point starts at 00.
- Why?
  - P500 is the earliest log that may not have been written to disk before crash.
- We have restored transaction table & dirty page table.

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600
20	T2	update	P500
30	T1	update	P505
40	T2	Commit	
System Crash			

transID	lastLSN
T1	30
~~T2~~	~~10~~

Transaction Table

pageID	recLSN
P500	00
P600	10
P505	30

Dirty Page Table

36

Redo Phase: Algorithm

- Scan forward from the redo point (LSN 00).
- For each update/CLR-undo log record LSN, perform redo unless one of the conditions holds:
  - The affected page is not in the dirty page table
    - It is not dirty. So no need to redo.
  - The affected page is in the dirty page table, but recLSN > LSN.
    - The page's recLSN (oldest log record causing this page to be dirty) is after LSN.
  - pageLSN >= LSN
    - A later update on this page has been written (pageLSN = the most recent LSN to update the page on disk).

37

Redo Phase: Example (1)

- Scan forward from the redo point (LSN 00).
- Assume that P600 has been written to disk.
  - But it can still be in the dirty page table.
- Scanning 00:
  - P500 is in the dirty page table.
  - 00(recLSN) = 00 (LSN)
  - -10 (pageLSN) < 00 (LSN)
  - Redo 00
- Scanning 10:

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600 (disk)
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN
T1	30

Transaction Table

pageID	recLSN
P500	00
P600	10
P505	30

Dirty Page Table

38

Redo Phase: Example (2)

- Scanning 10:
  - 10 (pageLSN) == 10 (LSN)
  - Do not redo 10

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600 (disk)
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN
T1	30

Transaction Table

pageID	recLSN
P500	00
P600	10
P505	30

Dirty Page Table

39

Undo Phase: Algorithm

- It scans backward in time from the end of the log.
- It needs to undo all actions from active (not committed) transactions. They are also called loser transactions.
  - Same as aborting them.
- Analysis phase gives the set of loser transactions, called ToUndo set.
- Repeatedly choose the record with the largest LSN value in this set and processes it, until ToUndo is empty.
  - If it is a CLR and undoNextLSN value is not null, use undoNextLSN value in ToUndo. If undoNextLSN is null, this transaction is completely undo.
  - If it is an update record, a CLR is written and restore the data record value to before-image. Use prevLSN value in ToUndo.

40

Undo Phase: Example (1)

- The only loser transaction is T1.
- ToUndo set is {T1:30}

LSN	TransID	Type	PageID
00	T1	update	P500
10	T2	update	P600 (disk)
20	T2	update	P500
30	T1	update	P505
40	T2	commit	
System Crash			

transID	lastLSN
T1	30

Transaction Table

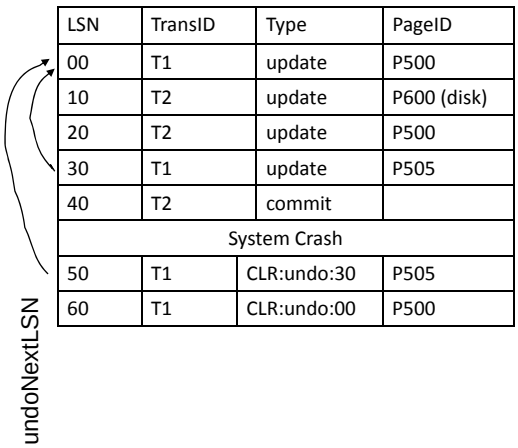
pageID	recLSN
P500	00
P600	10
P505	30

Dirty Page Table

41

Undo Phase: Example (2)

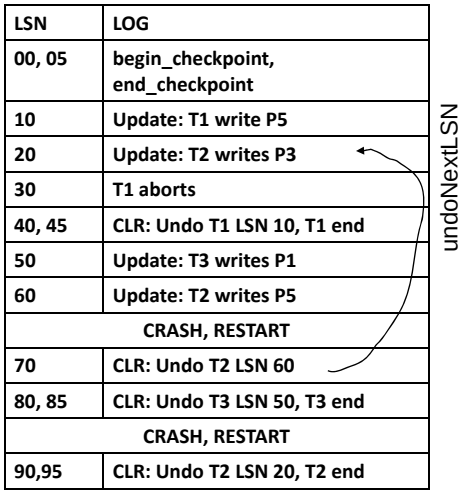
- The only loser transaction is T1000.
- ToUndo set is {T1:30}
- Undoing LSN:30
  - Write CLR:undo record log.
  - ToUndo becomes {T1:00}
- Undoing LSN:00
  - Write CLR:undo record log.
  - ToUndo becomes null.
  - We are done.



42

Crashes During Restart (1)

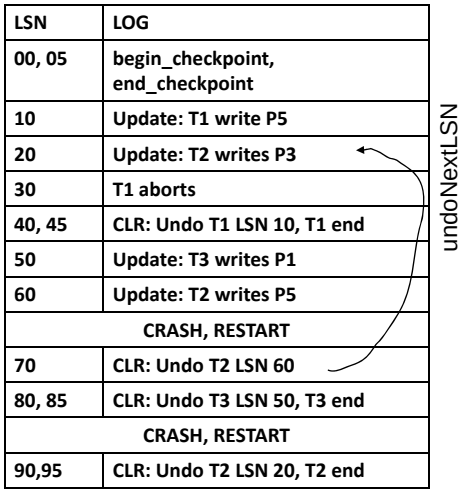
- After T1 aborts, undo actions from T1.
- Undo LSN #10: write CLR:undo record log for LSN #10.
- Dirty pages:
  - P1 (recLSN=50)
  - P3(20)
  - P5(10).
- Loser transaction:
  - T2(lastLSN=60)
  - T3(50)
- Redo phases starts at 10.
- Undo LSN #60.



43

Crashes During Restart (2)

- Undo LSN #50: write CLR: undo record log.
  - T3 is completely undone.
  - LSN #85,80,70 are written to stable storage.
- Crash occurs after restart.
  - Loser transaction is T2.
  - Read LSN #70, set ToUndo to #20.
  - Undo #20: write another CLR.
  - Done



44