Lecture #21: Database Recovery

15-445/645 Database Systems (Fall 2018)
https://15445.courses.cs.cmu.edu/fall2018/
Carnegie Mellon University
Prof. Andy Pavlo

1 ARIES

Algorithms for **R**ecovery and **I**solation **E**xploiting **S**emantics. Developed at IBM research in early 1990s. Not all systems implement ARIES exactly as defined in the original paper, but they are similar enough.

Main ideas of the ARIES recovery protocol:

- **Write ahead logging**: Any change is recorded in log on stable storage before the database change is written to disk (STEAL + NO-FORCE).
- **Repeating history during redo**: On restart, retrace actions and restore database to exact state before crash.
- **Logging changes during undo**: Record undo actions to log to ensure action is not repeated in the event of repeated failures.

2 WAL Records

We need to extend log record format to include additional info. Every log record now includes a globally unique *log sequence number* (LSN). Various components in the system keep track of **LSNs** that pertain to them:

- Each data page contains a *pageLSN*: The LSN of the most recent update to that page.
- System keeps track of *flushedLSN*: The max *LSN* flushed so far
- Before page i can be written to disk, we must flush log at least to the point where $pageLSN_i \leq flushedLSN$

3 Normal Execution

Transaction Commit

- Write `COMMIT` record to log.
- All log records to to transaction's `COMMIT` record are flushed to disk. Note that log flushes are sequential, synchronous writes to disk. There can also be multiple log records per log page.
- When the commit succeeds, write a special `TXN-END` record to log. This is used for internal bookkeeping and doesn't need to be flushed immediately.

Transaction Abort

- Aborting a transaction is actually a special case of the ARIES undo operation applied to only one transaction.
- We need to add another field to our log records:
  - **prevLSN**: The previous LSN for the transaction.
  - This maintains a linked-list for each transaction that makes it easy to walk through its records.

- **Compensation Log Record**
  - A *CLR* describes the actions taken to undo the actions of a previous update record. It has all the fields of an update log record plus the undoNext pointer (i.e., the next-to-be-undone LSN).
  - **CLRs** are added to the log like any other record but they never need to be undone.
- Algorithm:
  - First write `ABORT` record to log.
  - Then play back updates in reverse order to remove their effects. For each update, write a **CLR** entry and restore old value.
  - At end, write a `TXN-END` log record.

4 Checkpointing

Blocking Checkpoints

The DBMS halts everything when it takes a checkpoint to ensure that it writes a consistent snapshot of the database to disk. The is the same approach discussed in previous lecture:

- Halt the start of any new transactions.
- Wait until all active transactions finish executing.
- Flush dirty pages on disk.

Slightly Better Blocking Checkpoints

Like previous checkpoint scheme except that you the DBMS does not have to wait for active transactions to finish executing. We have to now record internal system state as of the beginning of the checkpoint.

- Halt the start of any new transactions.
- Pause transactions while the DBMS takes the checkpoint.

Active Transaction Table (ATT)

- One entry per active transaction
- transactionId: Unique transaction identifier
- status: the current "mode" of the transaction (**R**unning, **C**ommitting, **U**ndo candidate)
- lastLSN: Most recent LSN written by transaction
- Entry is removed when transaction commits or aborts

Dirty Page Table (DPT)

- Keep track of pages in the buffer pool contain changes from uncommitted transactions
- One entry per dirty page containing the recLSN, the LSN of the log record that first caused the page to be dirty.

Fuzzy Checkpoints

A *fuzzy checkpoint* is where the DBMS allows other transactions to continue to run. This is what ARIES uses in its protocol.

- Add new log records to track checkpoint boundaries
  - `<CHECKPOINT-BEGIN>`: Indicates the start of the checkpoint
  - `<CHECKPOINT-END>`: Contains the ATT + DPT

5 ARIES Recovery

The ARIES protocol is comprised of three phases. Upon start-up after a crash, the DBMS will execute the following phases:

Analysis: Read the WAL to identify dirty pages in the buffer pool and active transactions at the time of the crash.

Redo: Repeat all actions starting from an appropriate point in the log.

Undo: Reverse the actions of transactions that did not commit before the crash.

Analysis Phase

Start from last checkpoint found via the database's MasterRecord LSN.

- Scan log forward from the checkpoint.
- If you find a **TXN-END** record, remove its transaction from ATT.
- All other records, add transaction to ATT with status **UNDO**, and on commit, change transaction status to **COMMIT**.
- For `UPDATE` records, if page P not in **DPT**, add P to **DPT** and set its recLSN=LSN

Redo Phase

The goal is to repeat history to reconstruct state at the moment of the crash. Reapply all updates (even aborted transactions) and redo **CLRs**:

- Scan forward from log record containing smallest recLSN in **PDT**.
- For each update log record or CLR with a given LSn, redo the action unless:
  - Affected page is not in the **DPT**, or
  - Affected page is in **DPT** but that record's LSN is greater than smallest recLSN, or
  - Affected pageLSN (on disk) $\geq$ LSN.
- To redo an action:
  - Reapply logged action.
  - Set pageLSN to log records **LSN**.
  - At the end of the redo phase, write **TXN-END** log records for all transactions with status "C" and remove them from the ATT.

Undo Phase

- Undo All transactions active at the time of crash
- These are all transactions with "U" status in the **ATT** after the Analysis phase
- Process them in reverse **LSN** order using the lastLSN to speed up traversal
- Write a CLR for every modification