6.814 - Lecture 15, *ARIES Recovery*

Long Nguyen

November 1, 2017

1 Topics

Last time, we visited the idea that we can build a durable system through write-ahead logging. The log keeps track of states and actions that we can potentially redo or undo.

Do we always need to write undo/redo states?

Undo allows us to write dirty/uncommitted pages to disk (STEAL property). This makes buffer pool management easier.

FORCE policy: forces pages to database before committing.

2 ARIES Goals

1. No-force, steal; support undo and redo.
2. Runtime efficiency, done through lightweight checkpoints.
3. Support for complex operations like indexes.
4. Crash recovery.

2.1 ARIES Phases

The phases are analysis, redo, and undo.

1. Analysis: Going forward; how do we figure out what to redo?
2. Redo: Going forward; repeating history, uses physical log records.
3. Undo: Going backward; "logical".

Physical log record: This kind of record gives physical instructions for how to complete an action. For example, `Write 'SAM' into bytes 3-6 of page 10`.

Logical log record: This kind of record doesn't encode all the technical details. For example, `Insert 'SAM' into table 'emp'`.

Redo: We redo everything, even the loser transactions. When scanning through, we write everything we see in the log, even for transactions that may not have committed. The logic is similar to reapplying everything until the point during which the fatal crash happened. When we repeat history, we need to use physical log records because we want to maintain consistency. If we just go with the logical route, perhaps some modification already happened during the crash point. We have to write/update what we did exactly to the physical storage.

Undo: We need to perform logical undos, because we need to consider the state of the database at the time we undo.

2.2 Log/Page Format

A log contains LSNs, sequential numbers to keep track of actions and states.

Disk pages contain page LSNs, which are the last LSNs reflected in them.

For redos, start at the minimum recovery LSN value, because anything prior to that recovery LSN is guaranteed to have been written and reflected on disk. On a high level, we pretty much redo everything. We redo an update *unless*: page is not in the dirty page table (flushed prior to checkpoint), current LSN $<$ recovery LSN (page flushed and redirtied prior to checkpoint), current LSN $\leq$ page LSN.

2.3 Compensation Log Records

These log records are written after each undo operation. They tell us what we undid, and if we happen to crash during an undo phase, we can go back to these from the log. They redo the undos done during the redo phase of a crash recovery during the crash.

You can think of CLRs as physical information logged about the logical undo operations done before, such that we can refer to this physical information to avoid repeating undo work.