Lecture 07: Buffer Pools

15-445/645 Database Systems (Fall 2017)
Carnegie Mellon University
Prof. Andy Pavlo

"DBMS can manage memory far more better than OS"

Workloads

1. OLTP: On-line transaction processing
   (a) Short lived txns
   (b) Small footprint
   (c) Repetitive operations
   (d) Usually the kind of application that people build first
2. OLAP: On-line analyitical processing
   (a) Long running queries
   (b) Complex joins
   (c) Exploratory queries

Storage Models

1. There are different ways to store tuples
2. We have assumed the **n-ary storage model** so far. In the n-ary storage model, tuples have n attributes
   (a) Advantages
      i. Fast inserts, updates, deletes
      ii. Good for queries that need entire relation
   (b) Disadvantages
      i. N-ary storage model suffers when you bring into memory data for attributes you wont use

   ↳ stores n attrs for a single tuple contiguously in a block.
3. There is also the decomposition model or **column store**. In the decomposition model, the DBMS stores the data for a single attribute in its own block
   (a) Advantages
      i. Reduces the amount wasted I/O because DBMS only reads data in needs
      ii. Better query processing and data compression
   (b) Disadvantages

i. Slower for OLTP operations because of tuple splitting/stitching

4. **Spatial Control:** How and where to write pages on disk
5. **Temporal Control:** When to read pages into memory and when to write them to disk

Not using the OS

1. There exists some solutions to using OS in a DBMS
2. `madvise`: Tell the OS how you expect to read certain pages
3. `mlock`: Tell the OS that memory ranges cannot be paged out
4. `msync`: Tell the OS to flush memory ranges out to disk

- Not portable
- Thread block by OS not preventable.
- Correct write ordering is tricky.

Locks vs Latches

1. Locks : high-level constructs. exposed to the app.
   (a) Protect the database logical contents from other transactions
   (b) Held for transaction duration
   (c) Need to be able to rollback changes
   (d) Protect tuples, tables, indexes
2. Latches Mutex (OS). Not exposed to the app.
   (a) Protects the critical sections of the DBMS internal data structure from other threads ex) page table.
   (b) Held for operation duration
   (c) Do not need to be able to rollback changes

Buffer Pool

1. The DBMS always knows better so we want to manage memory and pages ourselves
2. **The buffer pool is an in-memory cache of pages read from disk**
   (a) It is a region of memory organized as an array of fixed size pages. Each array entry is called a **frame**
   (b) When the DBMS requests a page, an exact copy is placed into one of these frames
3. Metadata maintained by the buffer pool: ~> nearly the same compared to OS's VM system.
   (a) Page table: keeps track of pages that are currently in memory
   (b) Dirty-flag: Gets set when a thread modifies a page (will need to be written back)

→ maintained by page table.

(c) Pin counter: Number of threads touching that page

4. Optimizations:

   (a) Multiple buffer pools: The DBMS can also have multiple buffer pools for different purposes. This helps reduce latch contention and improves locality ex) per-db or per-page type.

   (b) Pre Fetching: The DBMS can also optimize by pre fetching pages based on the query plan. Commonly done when accessing pages sequentially → this is why prefetching is possible.

   (c) Scan Sharing: Query cursors can attach to other cursors on mid-scan and scan pages together
   ex) query could be different, but scanning could be the same.

Buffer Pool Replacement Policies

1. A replacement policy is an algorithm that the DBMS implements that makes a decision on which pages to evict from buffer pool when it needs space

2. **Goals**

   (a) Correctness

   (b) Accuracy

   (c) Speed

   (d) Meta-data overhead

3. LRU: Least Recently used

   (a) Maintain a timestamp of when each page was last accessed

   (b) DBMS picks to evict the page with the oldest timestamp

4. CLOCK

   (a) Approximation of LRU without needing a separate timestamp per page

      i. Each page has a reference bit

      ii. When a page is accessed, set to 1

   (b) Organize the pages in a circular buffer with a clock hand

      i. Upon sweeping check if a pages bit is set to 1

      ii. If yes, set to zero, if no, then evict

   (c) Clock hand remembers position between evictions

5. Problems with LRU and Clock replacement policies

   (a) LRU and Clock are susceptible to **sequential flooding**

   (b) Sequential flooding: Buffer is entirely overwritten by sequential pages

   (c) It may be that the LRU page is actually important due to not tracking meta-data of how a page is used

6. Better solutions → highly developed on commercial DBMSs.
   (a) LRU-K: Take into account history of the last K references
   (b) Priority hints: Allow txns to tell the buffer pool whether page is important or not → quite common on enterprise apps.
   (c) Localization: Choose pages to evict on a per txn/query basis

Allocation Policies

1. Global policies: How a DBMS should make decisions for all active txns
2. Local policies: Allocate frames to a specific txn without considering the behavior of concurrent txns

Other Memory Pools

1. The DBMS needs memory for things other than tuples and indexes
2. These other memory pools are not always backed by disk
   (a) Sorting and Join buffers
   (b) Query caches
   (c) Maintenance buffers
   (d) Log buffers
   (e) Dictionary caches