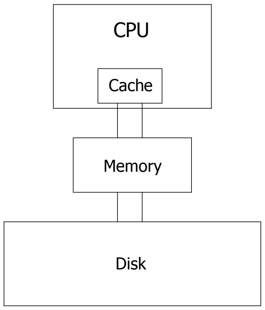
CS422 Principles of Database Systems

Buffer Management

Chengyu Sun
California State University, Los Angeles

Memory Hierarchy



Buffers in a Computer

- Disk cache
- Memory buffer
- L1, L2, and L3 caches

Why OS Memory Buffer Is Not Enough

- DBMS knows its data better
- Database buffer management must be coordinated with failure recovery mechanisms

Data Access Without Buffer Management

- Load a *block* into a *page*
- Access data in the page
- Write the page back to disk if the data is changed
- Release the page

So why do we need buffer management??

Buffer Management – Buffer Manager

- A buffer manager manages a fixed set of pages, called a buffer pool
- Each page in the buffer pool is called a buffer page

Buffer Management – Client Code

- Processes access disk through buffer manager
- These processes are referred to as client code (or just client)

Buffer Management – Pin

- `Buffer pin(Block)`
  - Load a block into a buffer page
  - Mark the buffer page as *pinned*
- A *pinned* buffer page is being used by some client code
- A *unpinned* buffer page is available for reuse

Four Possible Cases for Pin

- The block to be pinned is already buffered in memory
  - The buffer is pinned
  - The buffer is not pinned
- The block to be pinned is not buffered in memory
  - There are unpinned buffers available
  - All buffers are pinned

Buffer Management – Read/Write Data

- If the data in a page is changed, the page is called a dirty page

Buffer Management – Unpin

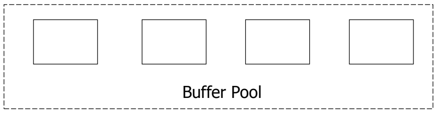
- `unpin(Buffer)`
  - Indicates the page is no longer used by the client

Buffer Management – Flush

- Write the dirty page(s) to disk
- When to flush
  - Before the page is pinned to a different block
  - At the request of the failure recovery mechanism

Example: Buffer Replacement

- Size of buffer pool: 4
- What does the buffer pool looks like after the following requests: `pin(0), pin(1),pin(2),pin(3),unpin(3), unpin(1),unpin(2),pin(5)`



Buffer Replacement Policies

- Naïve
  - Sequentially scan the buffer pool and replace the first unpinned page
- Clock
- FIFO (First In First Out)
- LRU (Least Recently Used)

Problem of the Naïve Policy

- `pin(1), unpin(1), pin(2), unpin(2), pin(1), unpin(1), pin(2), unpin(2)…`

Clock Policy

- Sequentially scan the buffer pool and choose the first unpinned page
- Start the next scan at the page after the previous replacement

Implementing FIFO and LRU

- FIFO
  - For each buffer page, keeps the time when the block is read in
- LRU
  - For each buffer page, keeps the time when the page is unpinned

Readings

- Chapter 13.4 and 13.5 of the textbook