Database Buffer Management

Yanlei Diao
UMass Amherst
Feb 20, 2007

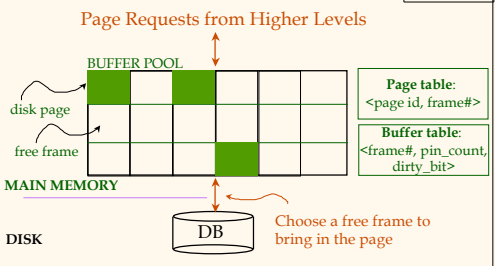
Slides Courtesy of R. Ramakrishnan and J. Gehrke

DBMS vs. OS File System

OS does buffer management: why not let OS manage it?

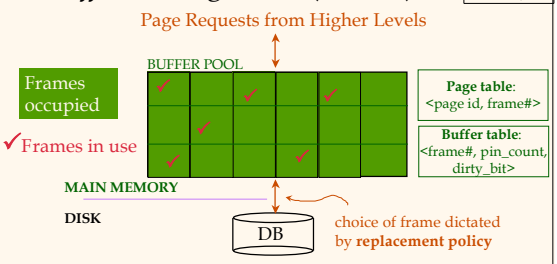
- Differences in OS support: portability issues
- Buffer mgmt in DBMS needs to work with transaction management.
  - *pin* a page in buffer pool, *force* a page to disk (important for implementing CC & recovery)
- Buffer mgmt in DBMS needs to understand query-specific data access patterns.
  - adjust *replacement policy,* and *pre-fetch* pages based on access patterns in typical DB operations.

Buffer Management in a DBMS



- *Data must be in RAM for DBMS to operate on it!*

Buffer Management (Contd.)



- *When all frames are occupied, pick one frame not in use using the replacement policy.*

When a Page is Requested ...

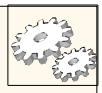
- If the requested page is not in the buffer pool:
  - Choose a frame for *replacement*
  - If the frame is *dirty*, write it to disk
  - Read the requested page into the chosen frame
- *Pin* the page and return its address to the caller.

➨ *If requests can be predicted (e.g., sequential scans) pages can be pre-fetched several pages at a time!*

More on Buffer Management

- Requestor of a page must *unpin* it, and indicate if the page has been modified:
  - *dirty* bit is used for this.
- A page in the pool may be requested many times,
  - a *pin count* is used. A page is a candidate for replacement iff *pin count* = 0.
- Transaction mgmt may entail additional I/O when a frame is chosen for replacement.
  - *Write-Ahead Log* protocol; more later.

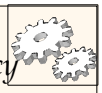
Buffer Replacement Policies



- A frame is chosen for replacement by a *replacement policy:*
  - Least-recently-used (LRU)
  - Clock, a variant of LRU
  - Most-recently-used (MRU)
  - First-in-first-out (FIFO)
  - Last-in-first-out (LIFO)
  - LRU2
  - …

7

Impact of the Replacement Policy



- The policy can have a big impact on # of I/O's; depends on the *access pattern*.
- None of the stochastic policies is uniformly appropriate for typical DBMS access patterns.
- *Sequential flooding*: Nasty situation caused by LRU + repeated sequential scans.
  - # buffer frames < # pages in file means each page request causes an I/O.
  - MRU much better in this situation (but not in all situations, of course).

8

Advanced Buffer Management

- Just a few basic *data access patterns* in relational DBs, with noticeable *locality behaviors*.
- A DB operation can be *decomposed* into a subset of these access patterns.
- To reduce I/Os, expose these patterns to the buffer manager for correct estimation of:
  - *buffer size*: many queries share the buffer pool; need to know how to allocate frames to each query.
  - *replacement policy*: evict unused pages to make room for newly requested pages.

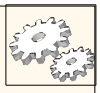
9

DBMIN: QLSM

- Query Locality Set Model (QLSM)

Sequential data access	*Straight Sequential (SS)*	*Clustered Sequential (CS)*	*Looping Sequential (LS)*
Behavior	One access w/o repetition	Local rescan	Repetitive sequential scan of a file
Examples	File scan	Merge join with point of plurality	Nested loops Join: inner
Locality set	1	Largest cluster (# pages with the same key value)	# pages of the file
Replacement policy	Any; pre-fetching is good	If enough memory, FIFO/LRU (evict old data)	If file doesn't fit, MRU

DBMIN: QLSM (Contd.)



Random data access	*Independent Random*	*Clustered Random*
Behavior	A series of independent accesses	A series of random accesses
Examples	Indexed scan: access to data pages through a non-clustered index	Indexed join: outer is a clustered file with duplicate join values, inner is through a non-clustered index.
Locality set	1 or b (# of pages that will be re-visited)	Largest cluster (maximum # records in the cluster)
Replacement policy	Any	Any

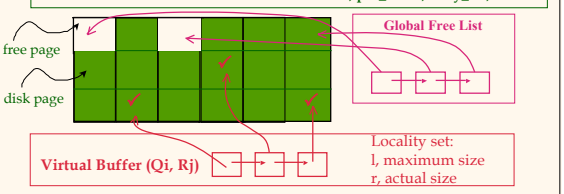
DBMIN: QLSM (Contd.)

Hierarchical Index Access	*Straight Hierarchical*	*Hierarchical w. SS*	*Hierarchical w. CS*	*Looping Hierarchical*
Behavior	Index is traversed once	Tree traversal+ scan of leaves	Local rescan in page access	Repeated access to the index structure
Examples	Equality search using an index	Range search using the index	Inner of indexed nested loops join, w. duplicate values from the outer	Inner of indexed nested loops, w.o. duplicate values from the outer
Locality set	1	1	= CS	Closer to root, often only sure about the root for high F, 3~4.
Replacement policy	Any	Any	If enough memory, FIFO or LRU	LIFO

DBMIN Buffer Management

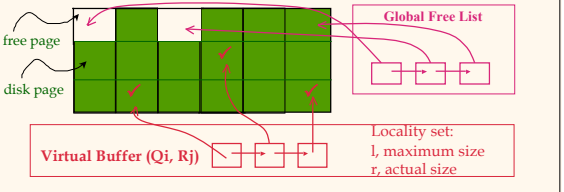
- A buffer manager that uses QLSM
  - e.g. all those access patterns, sizes of locality sets, suitable replacement policies
- To reduce I/O: a "virtual buffer" per query-file (Q, R)
  - Can allocate # frames as the locality set requires
  - Use the right replacement policy
- Data sharing via the use of a global buffer pool
  - Two queries accessing the same page: one copy of the page in the global buffer, two pointers.
  - Fairness?
- Load control
  - *Overload*: can not take more queries
  - *Underload*: a query can use the entire global pool!

DBMIN Algorithm



- Upon page request for (Q, R)
  - Page found in both global table and (Q, R) locality set: just use it
  - Page found in global table but not (Q, R) locality set: grab the page
    - Page has owner or not?
    - r>l: pick a page in locality set using replacement policy, put it back to global free list
  - Page not in memory

DBMIN Algorithm (Contd.)

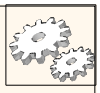


- Overload? suspend a new query if maximum sizes of existing locality sets sum up to the size of global buffer pool
- Underload? (Q, R) keeps updating its locality set, but those pages stay in memory. It can use the entire global buffer pool!

Summary

- Buffer manager brings pages into RAM.
  - Page stays in RAM until released by requestor.
  - Written to disk when frame chosen for replacement (which is sometime after requestor releases the page).
  - Choice of frame to replace based on *replacement policy*.
  - Tries to *pre-fetch* several pages at a time.
- Advanced buffer management
  - Understand access patterns to allocate enough buffer and choose the right replacement policy
  - Load control for overload and underload

Summary (Contd.)



- DBMS vs. OS File Support: DBMS needs features not found in many OS's, e.g.,
  - forcing a page to disk (transaction mgmt)
  - controlling the order of page writes to disk (WAL for recovery)
  - ability to control pre-fetching
  - page replacement policy based on predictable access patterns, etc.

Questions