File-System Performance

Dr Andrew Scott

a.scott@lancaster.ac.uk

1

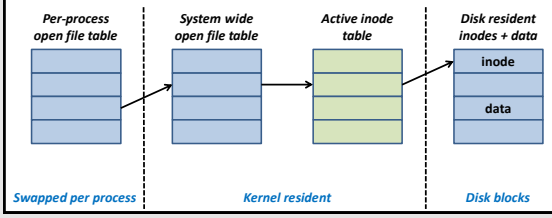
Speeding up access…

FILE-SYSTEM CACHES

2

Speeding up access: Kernel Tables

- Kernel maintains cache of recently used inodes
  - *Known as Active index node (inode) table*
  - Accessed using hash of inode + device (filesystem) ID
    - Remember: inode number only valid within same filesystem



3

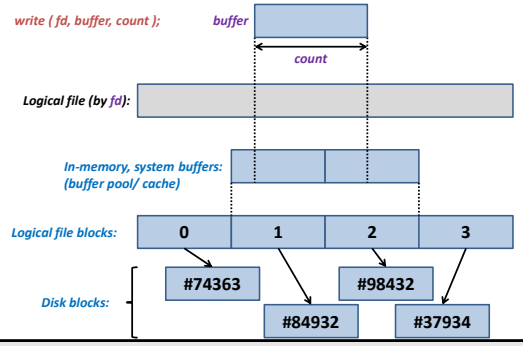
Buffering and Block Caches

- Processes read and write bytes
  - Disks handle blocks
- High volume of reads and writes
  - Naïve per byte or word: read, modify, write
  - Easy to spend all time moving disk blocks
- Buffers allow
  - Decoupled byte and block reads
  - Caching* of blocks to reduce disk activity
  - Use *read-ahead* to speed up access (leverage locality)
    - Don't just read requested block, read next *n* blocks as well

* For removable media often use *write-through caches* to ensure updates written immediately -- slow as, for example, individual character writes would result in one disk write per byte

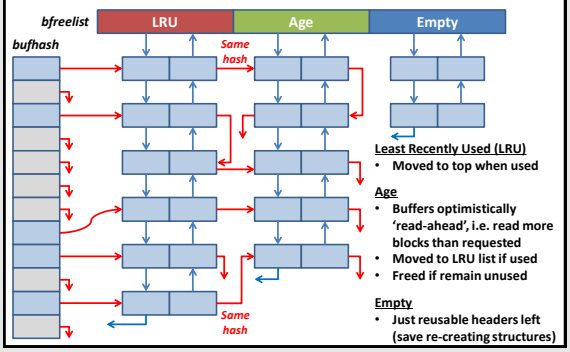
5

File Input/ Output on Block Devices



6

Efficient buffering: *Buffer Pool*



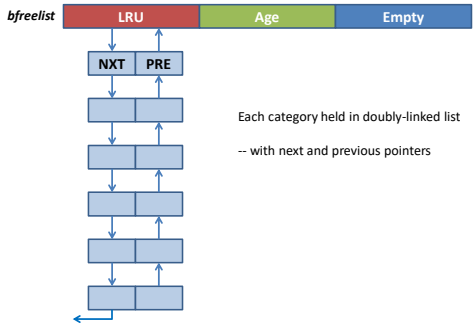
Least Recently Used (LRU)
- Moved to top when used

Age
- Buffers optimistically 'read-ahead', i.e. read more blocks than requested
- Moved to LRU list if used
- Freed if remain unused

Empty
- Just reusable headers left (save re-creating structures)

7

Efficient buffering: *Buffer Pool II*

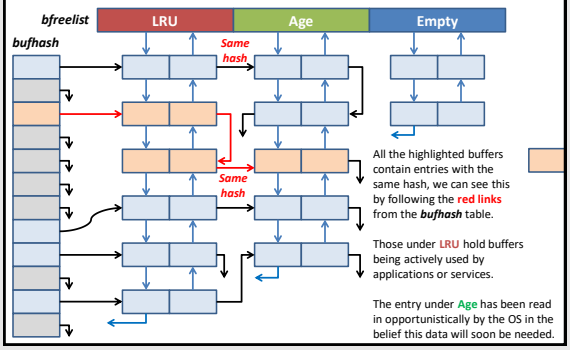


Each category held in doubly-linked list

-- with next and previous pointers

8

Efficient buffering: *Buffer Pool III*



All the highlighted buffers contain entries with the same hash, we can see this by following the **red links** from the ***bufhash*** table.

Those under LRU hold buffers being actively used by applications or services.

The entry under Age has been read in opportunistically by the OS in the belief this data will soon be needed.

9

Filesystem Flush/ Sync (-hronise)

- Internal buffers must be regularly flushed/ sync'd to disk
  - Protect against lost data in event of system failure
  - Speed-up reuse of Least Recently Used (LRU) buffers
    - Less chance of cache victim being dirty (out of sync with on-disk copy)
    - Any modified buffers must be written out before they are reused
- Use a dirty flag to track updates
  - Regular/ periodic *sync* to flush 'dirty' buffers to disk
  - Sync command/ *system-call* can force this
    - Usually called, for example, as part of system shutdown process

10

Delayed Allocation/ Allocate on Flush

- Using buffers allows files to grow in memory
  - Reduces frequency of disk block allocation requests
- Sync forces in-memory data to disk
  - May demand multiple block allocations
  - As requests now come as a batch, blocks can be allocated as contiguous run on disk
    - Limits fragmentation
    - More efficient file access due to contiguous blocks

Used in ext4, HFS+

11

Extents: *building on delayed writes*

- Portions of file stored as contiguous set of disk blocks
  - Makes file reads more efficient
    - Less meta-data (block pointers and indexes, etc.) to process
    - Less fragmentation
    - Less head movement
  - Applications can use *pre-allocation* with *fallocate( )*
- File information contains list of extents:

Logical Block address (start of extent in file)
Number of contiguous disk blocks in extent
Block address of first disk block in extent

Used in ext4 and similar *Cluster Run* scheme used in NTFS

12

Other Improvements

- Database style indexing
  - Use H or B+ based indexes for
    - Directories
    - Accessing tree of *extents*
- Fine grained timestamps
  - Granularity of 1 sec no longer acceptable… use *ns*
- Hash based data deduplication
  - Don't store multiple copies of same data
    - Many users with same email attachments, for example
  - Look for repeated block runs, replace with links

13