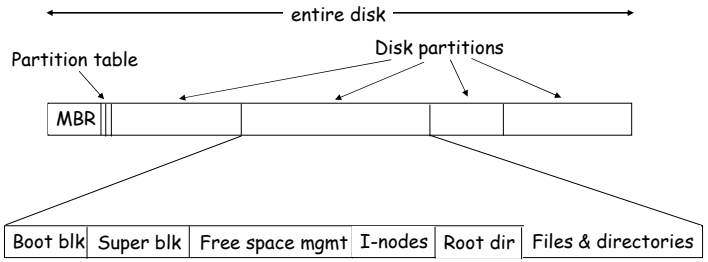
More on File System

CS 256/456

Dept. of Computer Science, University of Rochester

File System Layout

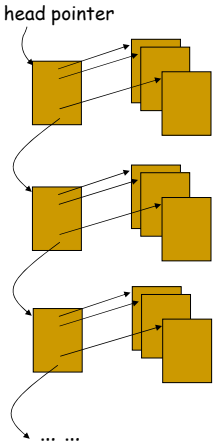


File Space Allocation on the Disk

- File allocation methods
  - contiguous allocation
  - linked allocation
  - indexed allocation
- Metrics:
  - access speed (sequential & random)
  - space utilization

Free-Space Management

- Free-space management for memory?
- Bit map
  - requires some extra space
  - efficiency?
    - getting the address of one free block
    - getting the addresses of a number of free blocks
- Linked list (free list)
  - no waste of space
  - efficiency?
- Grouping



File System Caching

- File content is cached in memory buffer for later reuse
  - what is the basic unit of such caching?
- Replacement policy for file system buffer cache
  - LRU replacement is one possibility; but sequential access is very likely in file system I/O
  - MRU or free-behind

File System Prefetching

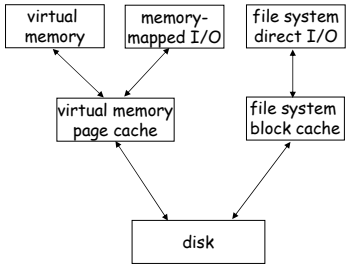
- File content is read ahead of time for anticipated use in the near future
  - often sequential (based on past access history on the file)
  - what is the advantage of file prefetching?
  - what is the danger of file prefetching?
- Linux 2.4 file prefetching
  - first prefetch 16KB
  - then gradually increase the prefetch size when the OS senses a sequential access pattern
  - the prefetch size does not go beyond 128KB
  - it falls back to 4KB read-ahead when the access is considered random

Beyond Past Access History

- Problems with prefetching based on "past access history"
  - not always accurate
  - can rarely recognize non-sequential patterns
- Ideas beyond past access history?
  - application I/O hints
  - automatic I/O hints based on speculative execution [Chang&Gibson 2000], [Fraser&Chang 2003]

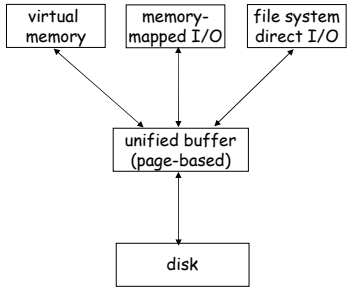
Buffer Cache in Main Memory

- Memory-mapped I/O naturally share page cache with the virtual memory system
- Problems:
  - double buffering
  - inconsistencies



Unified Buffer Cache & Unified Virtual Memory

- A unified buffer cache uses the same page cache to store
  - virtual memory pages
  - memory-mapped pages
  - file system direct I/O data



Multi-level I/O Buffer

- buffer cache in the main memory
- track cache on the disk controller

File Systems You Heard Of

- FAT
- NTFS
- ext2
- ext3
- ReiserFS

Recovery from Machine Crash

- File system operations are not atomic; a sudden machine crash may leave the file system in an inconsistent state
- Consistency checking and fix
  - it takes too long
- Journaling file system: Ext3, ReiserFS
  - maintain a dedicated journal that logs all operations
  - the logging happens before the real operation
  - each logging is made to be atomic
  - after the completion of an operation, its entry is removed from the journal
  - at the recovery time, only journal entries need to be examined ⇒ fast recovery
  - similar to transactions in database systems
  - performance impact?

Log-Structured File Systems

- With CPUs faster, memory larger
  - buffer caches can also be larger
  - most of read requests can come from the memory cache
  - thus, most disk accesses will be writes
  - poor disk performance when most writes are small
- LFS Strategy [Rosenblum & Ousterhout SOSP1991]
  - structures entire disk as a log
  - always write to the end of the disk log
  - when updates are needed, simply add new copies with updated content; old copies of the blocks are still in the earlier portion of the log
  - periodically purge out useless blocks

Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).