FILE SYSTEM OPTIMIZATION

Module Number 4. Section 6
COP4600 – Operating Systems
Richard Newman

FILE SYSTEMS TOPICS

- Introduction
- Directories
- File Allocation
- Block Management
- File System Reliability
- File System Optimization

FILE SYSTEM OPTIMIZATION

- Block Size Optimization
- Block Cache
- File Descriptors
- File System Layout
- Solid State Disks
- RAID for speed
- Log Structured File System

BLOCK SIZE

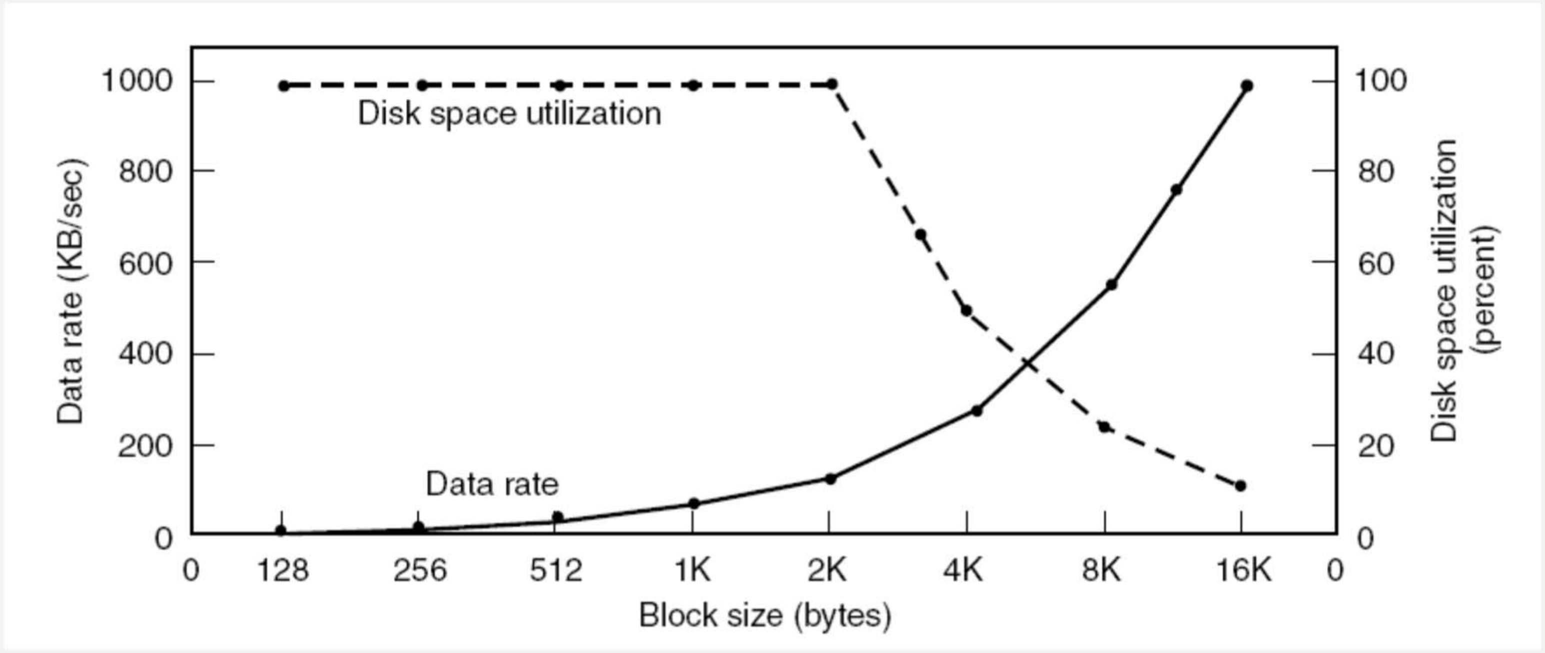
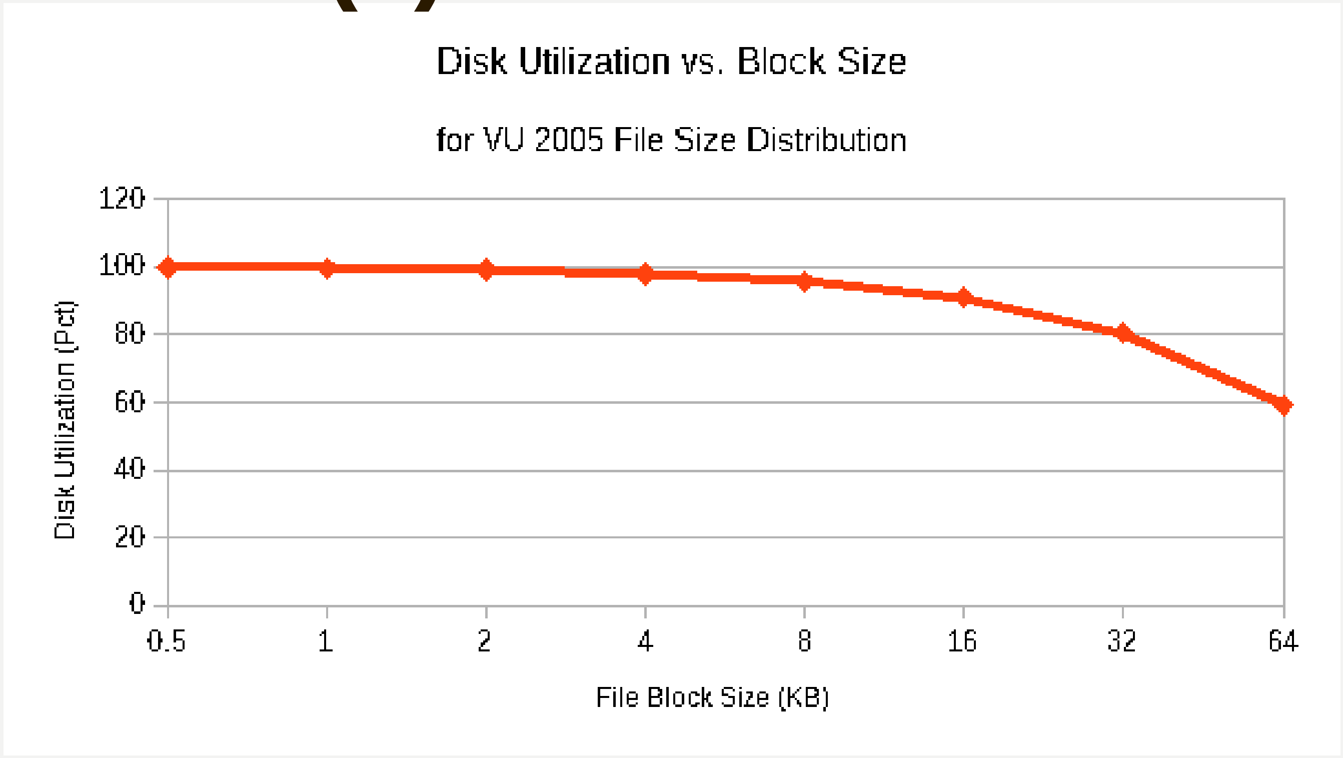


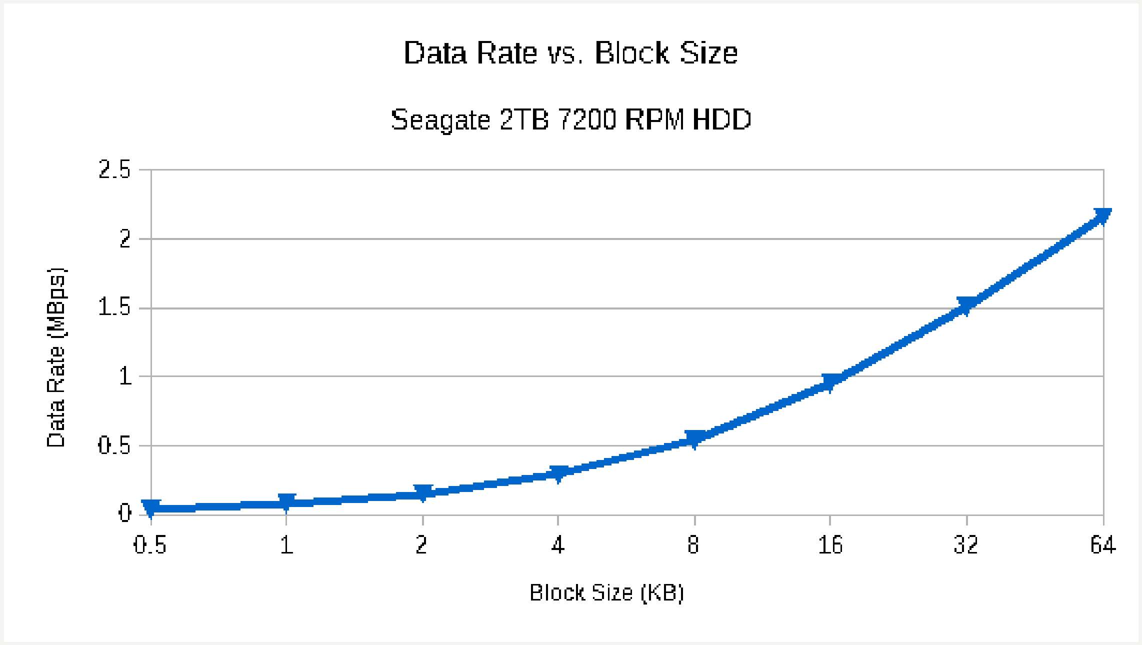
Figure 5-17. The solid curve (left-hand scale) gives the data rate of a disk. The dashed curve (right-hand scale) gives the disk space efficiency. All files are 2 KB.

BLOCK SIZE (3)



Approximate disk space utilization (due to internal fragmentation) as function of block size for VU 2005 file size distribution

BLOCK SIZE (4)



Data rate (MBps) as function of block size for Seagate 2TB HDD (7200 RPM rotation speed, 63 sectors/track, 8.5 ms mean seek)

BLOCK SIZE (4)

2013 study (Welch & Noer):

•25-90% of files are < 65KB in size

•But these only account for 3% on average (5-15% extreme) of total file space used….

•Average lengths ran from 75KB to over 16 GB for the installations studied, typical was a few MB average.

•Maximum file sizes ran 13 MB to 85 TB.

•They suggest using Solid State Drive (SSD) for small files and HDD for large files….

SPEEDING UP DISK I/O: CACHING

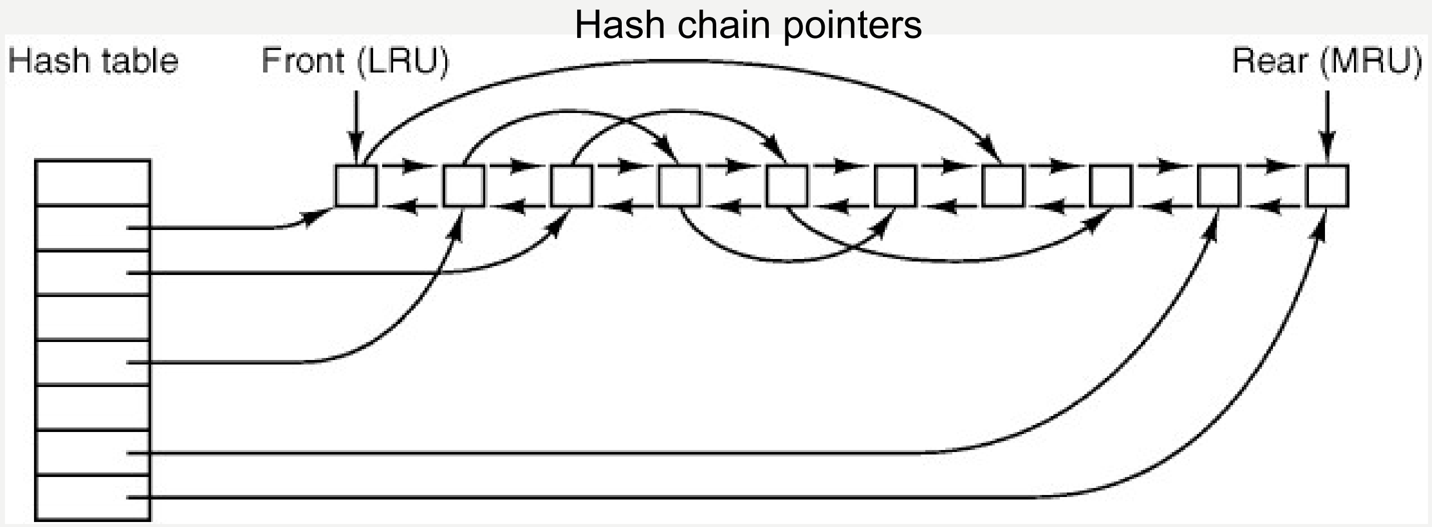


Figure 5-20. The buffer cache data structures.

FILE DESCRIPTORS

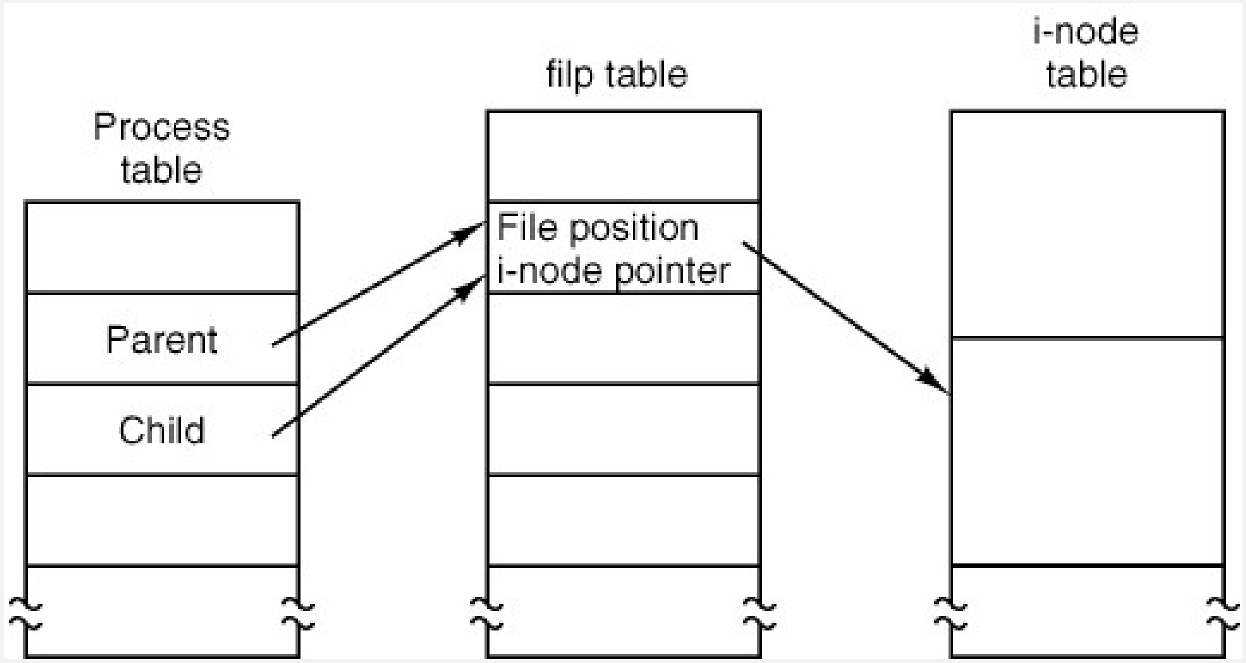
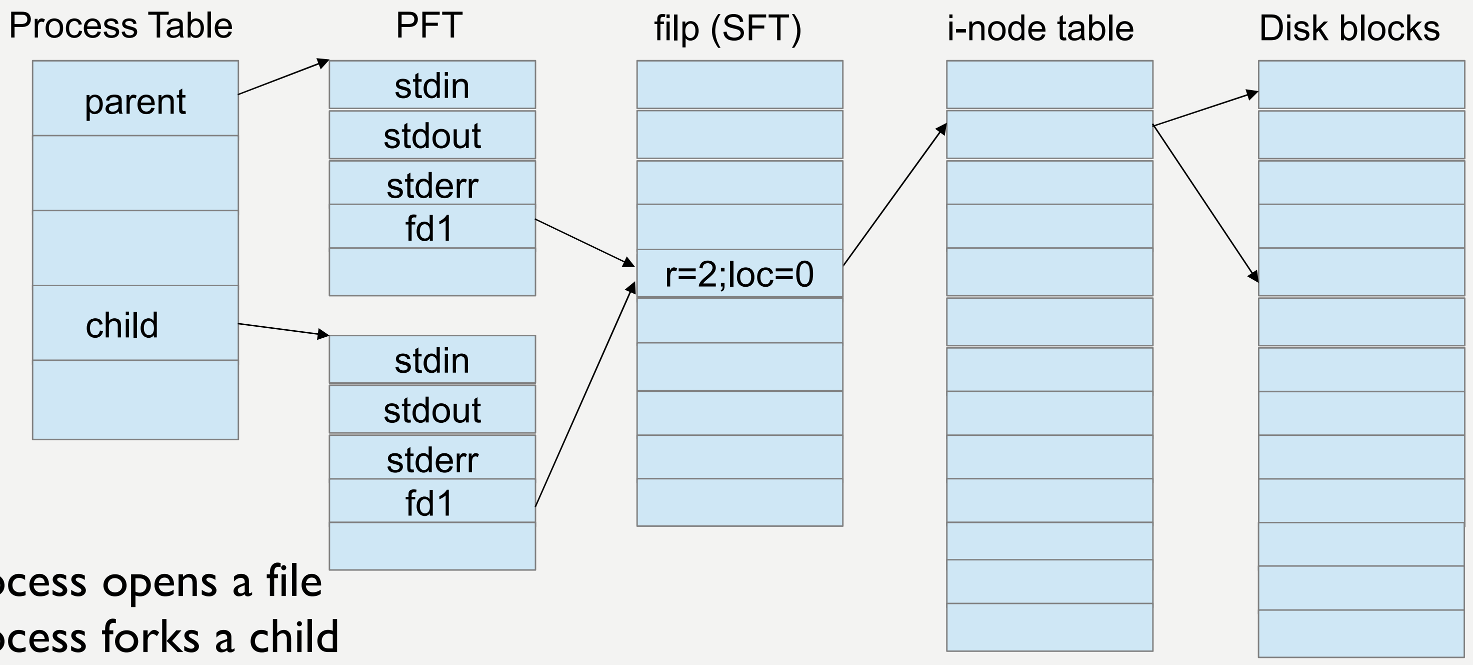


Figure 5-39. How file positions are shared between a parent and a child.

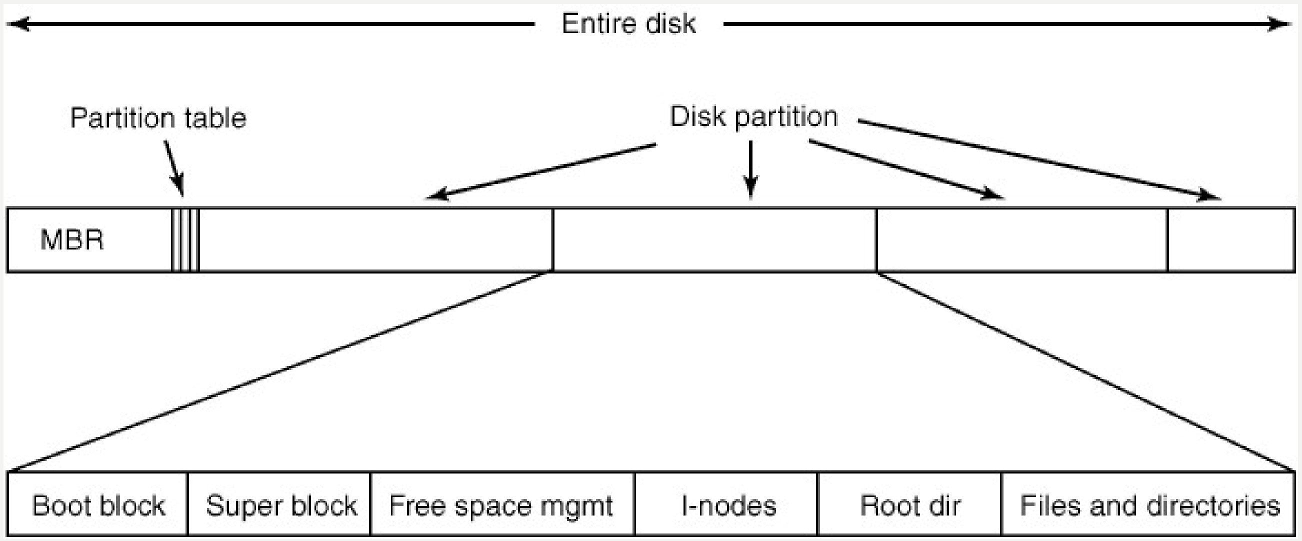
filp table (SFT) also holds access mode and ref count (number of PFT entries pointing to SFT entry) for garbage collection

FILE DESCRIPTORS



Process opens a file
Process forks a child
Child gets copy of parent's PFT
Including pointers into filp (SFT) for any open file
Requiring that the refcount be incremented
Access by either process will change loc value for both
Parent closes file – decrement refcount

SPEEDING UP DISK I/O: RECALL FILE SYSTEM LAYOUT



Typical Unix file system layout. All inodes in inode table near beginning of paritition, but files are all over the disk

=> lots of head movement (costly seeks)

SPEEDING UP DISK I/O: REDUCING DISK ARM MOTION

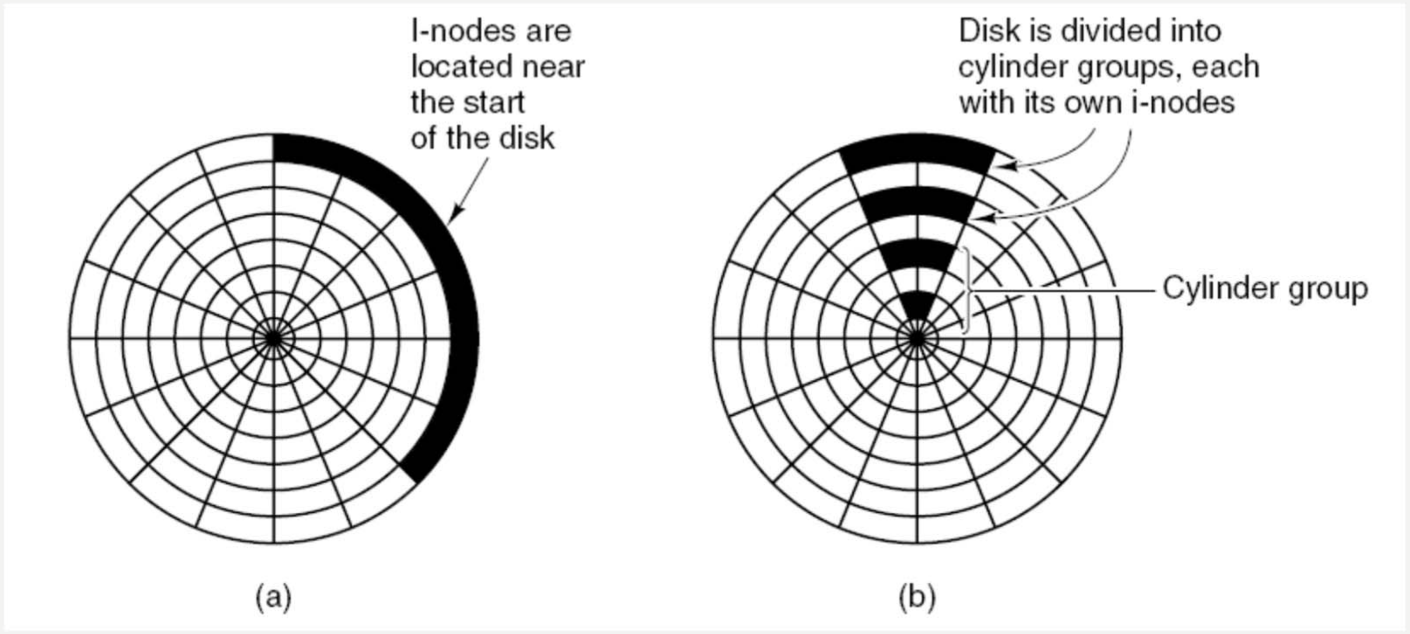

I-nodes are located near the start of the disk
Disk is divided into cylinder groups, each with its own i-nodes
Cylinder group
(a)
(b)

Figure 5-21. (a) I-nodes placed at the start of the disk.

(b) Disk divided into cylinder groups, each

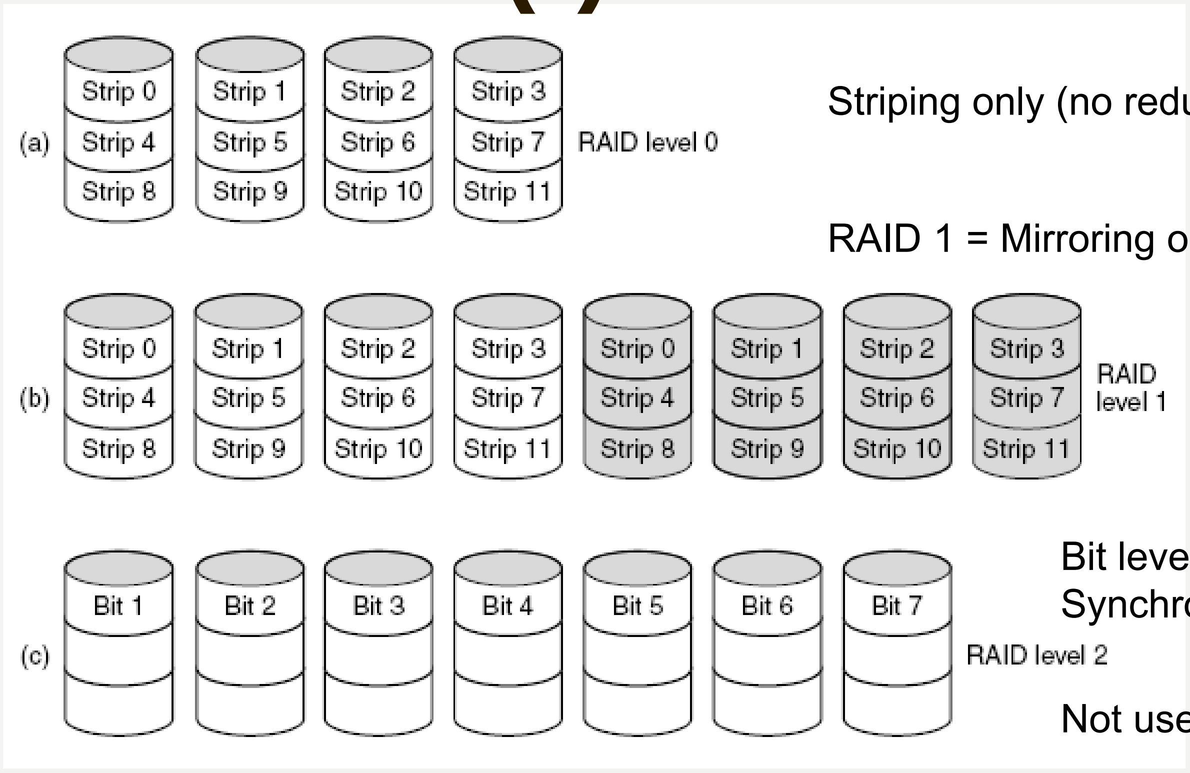
with its own blocks and i-nodes.

SPEEDING UP DISK I/O: USE SOLID STATE DRIVE

Solid State Drive (SSD)

- No moving parts
- No seek time
- No rotational latency
- 10-20 x faster than HDD
- Can get smaller as chips shrink
- Survives vibration/shock much better
- Costs 5-10 x more than HDD per GB storage
  ($160 for 500 GB SDD, $200 for 3 TB HDD)
- Maximum size HDD is much larger (5-10x)

RAID STORAGE (1)



Striping only (no redundancy)

RAID 1 = Mirroring only (no striping)

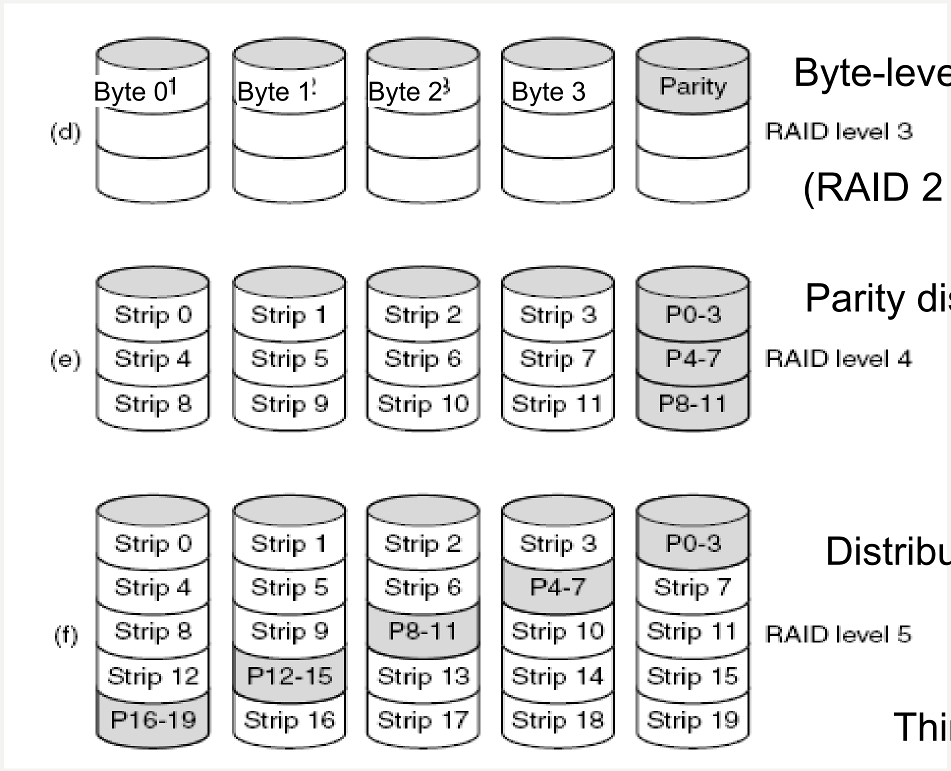
This is RAID 10 (Mirror & Strips)

Bit level striping - Synchronous spin

Not used in practice

RAID levels 0 through 2.
Backup and parity drives are shown shaded

RAID STORAGE (2)



Byte-level striping with parity

(RAID 2 also has parity bit)

Parity disk

Distribute parity strips

Why?

Think about parity disk load

RAID levels 3 through 5.

RAID-6 is like RAID-5 but with two redundant drives

LOG-STRUCTURED FILE SYSTEMS

- Caches are getting large enough so that reads are fast (after first access)
- Writes are often small and slow, and often repeat to same block
- So, structure FS as a log – collect writes (including to i-nodes and directories) into one big segment and write at high speed
- Need more metadata to locate i-nodes
- Cleaner thread to do circular scan and compact log (remove old segments, put still valid blocks into new segment)

SUMMARY

- Block Size Tradeoffs – getting data on/off disk
  - Larger => faster I/O per block, more waste per file
- Block Caching – avoiding disk access
- File Descriptors/Capabilities - check and access quickly
- File System Layout – reduce seek time
- Solid State Disks – eliminate seek time
- RAID – parallel I/O for streaming speed
- Log-Structured File Systems – collect active files for fast access, one big circular buffer