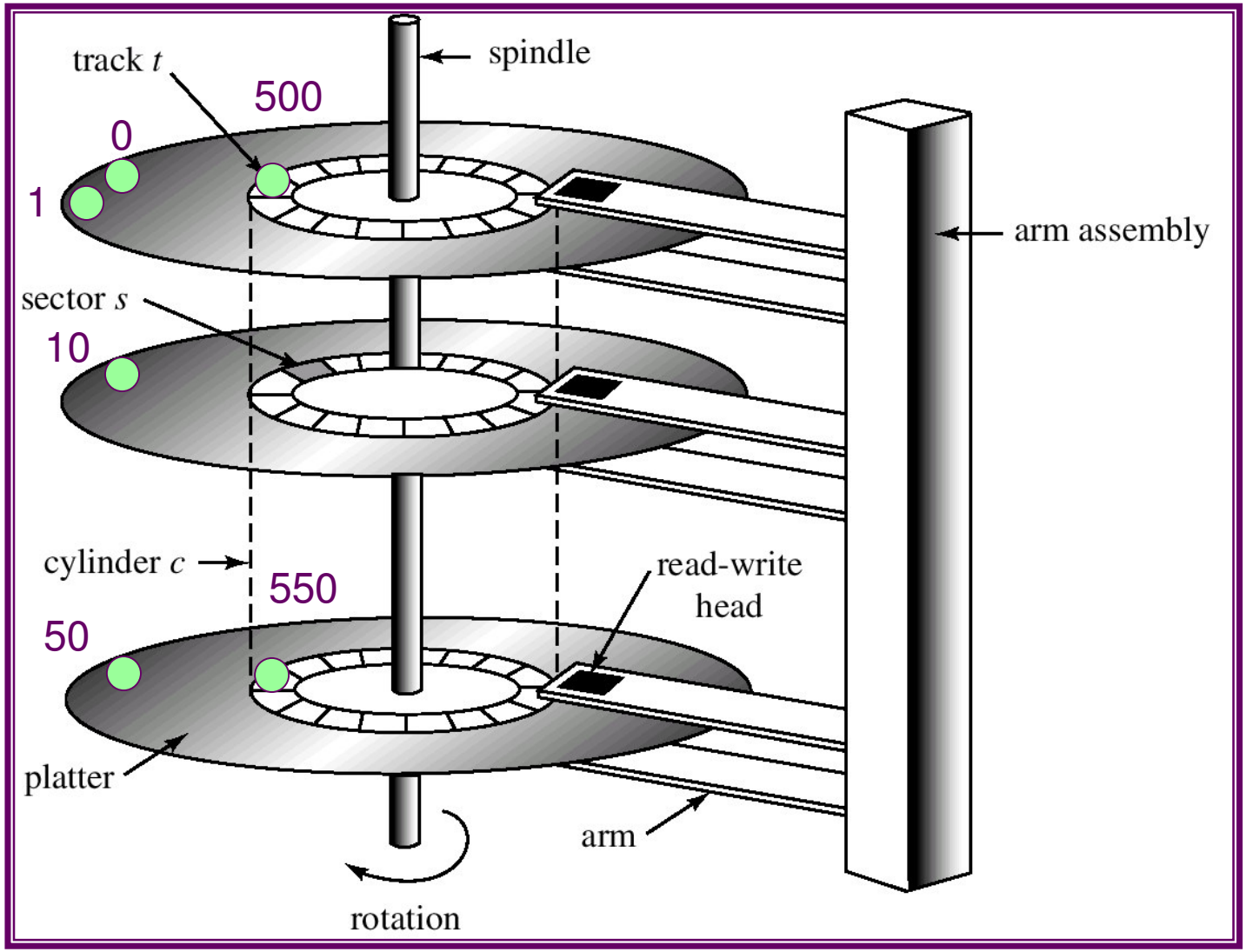
Chapter 14: Mass-Storage Systems

- Disk Structure.
- Disk Scheduling.
- RAID.

Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Disk Structure (Cont.)



Disk Drive Management

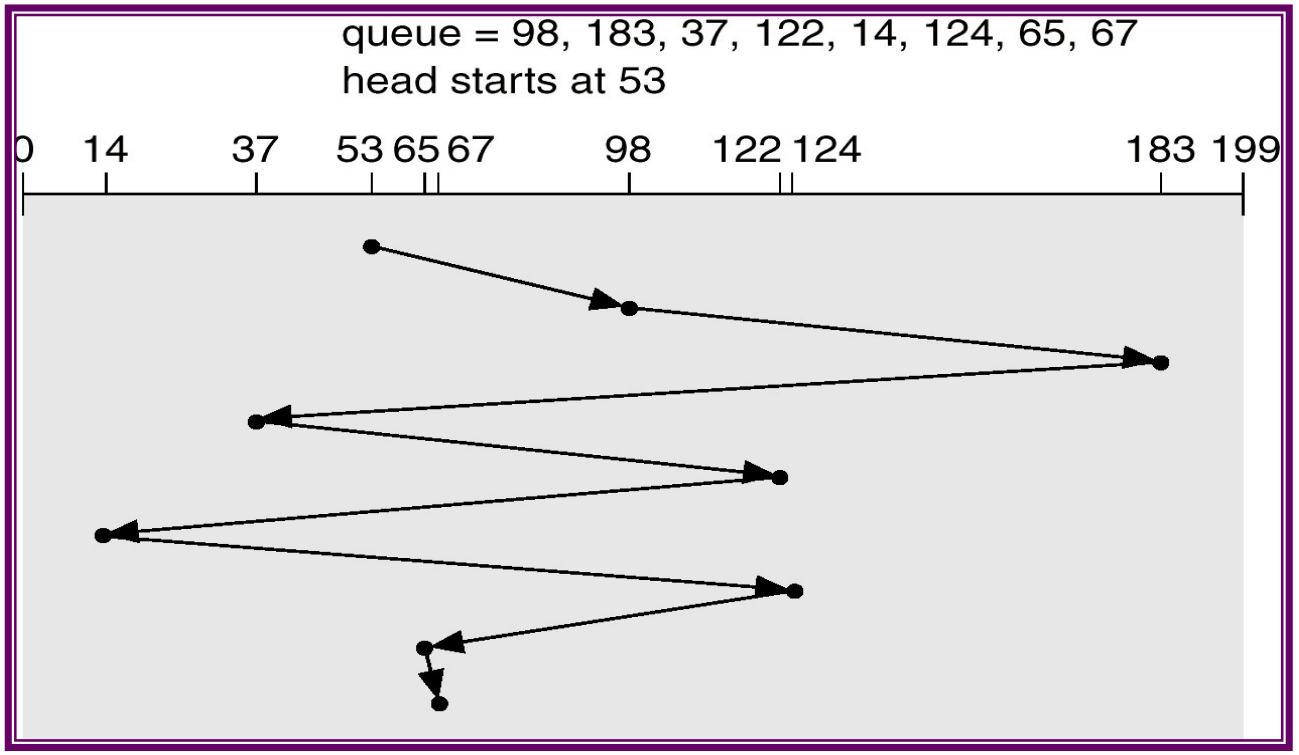
- Disk drive management goal: fast *access time* and high *disk bandwidth*.
- Major components of access time:
  - *Seek time* is the time for the disk motors to move the heads to the cylinder containing the desired sector.
  - *Rotational latency* is the additional time waiting for the disk to rotate the desired sector under the disk head.
  - Seek time far larger than rotational latency.
  - Seek time proportional to seek distance.
  - *Disk scheduling* minimizes seek time.
- Disk bandwidth: the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
  - Better seek time per request improves bandwidth.
  - Beyond this, *parallelism* improves bandwidth.

Disk Scheduling

- How should we schedule the servicing of disk I/O requests?
- Illustrate with queue of disk requests for blocks on listed cylinders (min cylinder: 0; max: 199):

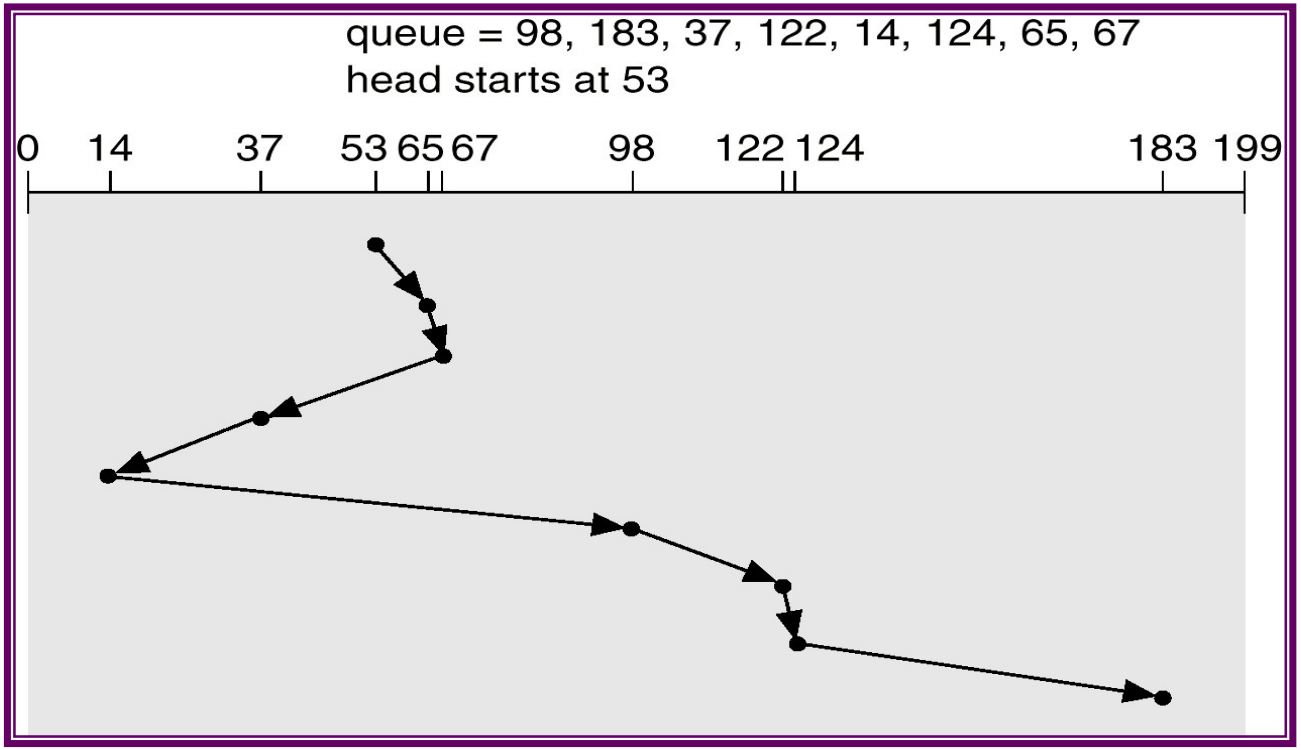
  98, 183, 37, 122, 14, 124, 65, 67

  Initially, the disk head is at cylinder 53.
- First come, first served: total head movement is 640 cylinders.



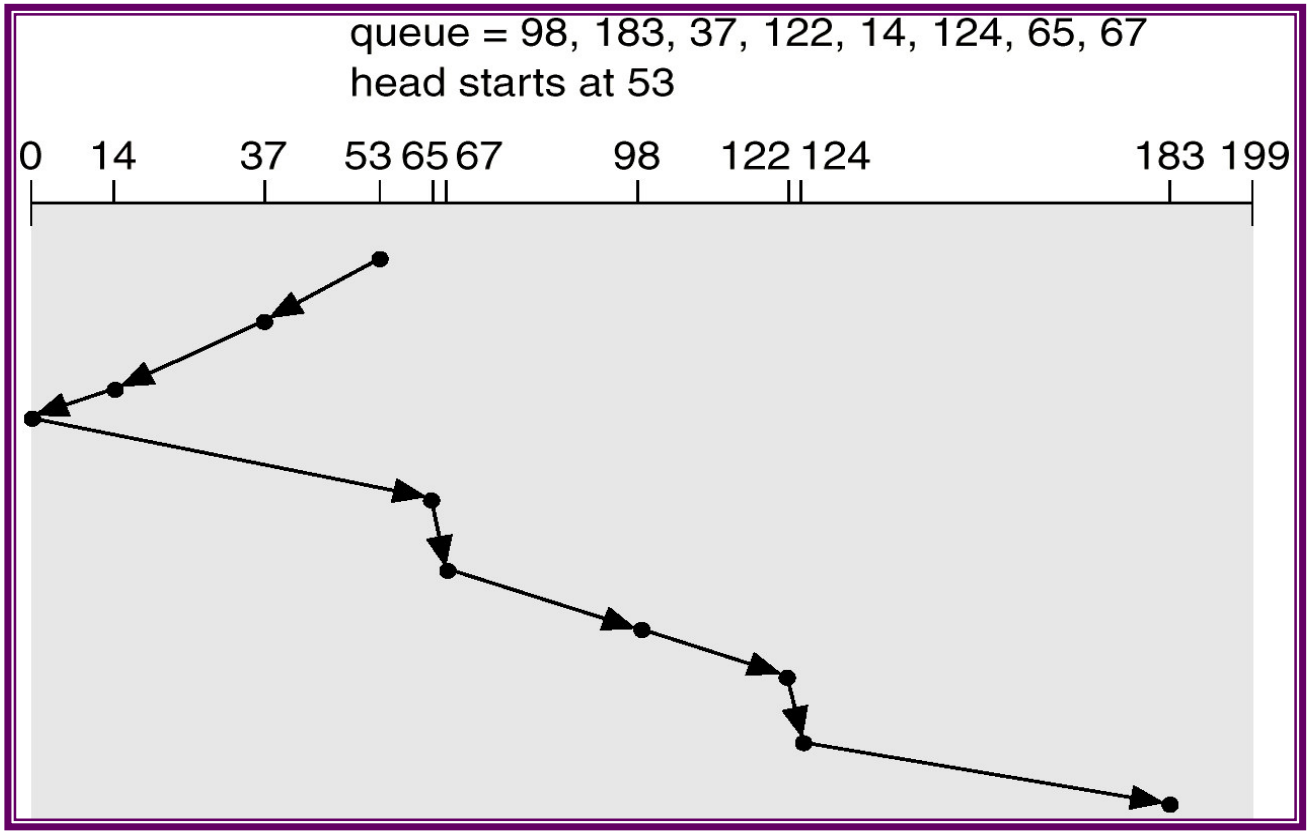
Shortest-Seek-Time-First (SSTF)

- Selects the request with the minimum seek time from the current head position.
- May cause starvation of some requests.
- In example, total head movement is 236 cylinders.
- Like SJF but not optimal (we use different measure here): if we go from 53 to 14, and service 65, 67 later, total is 208.



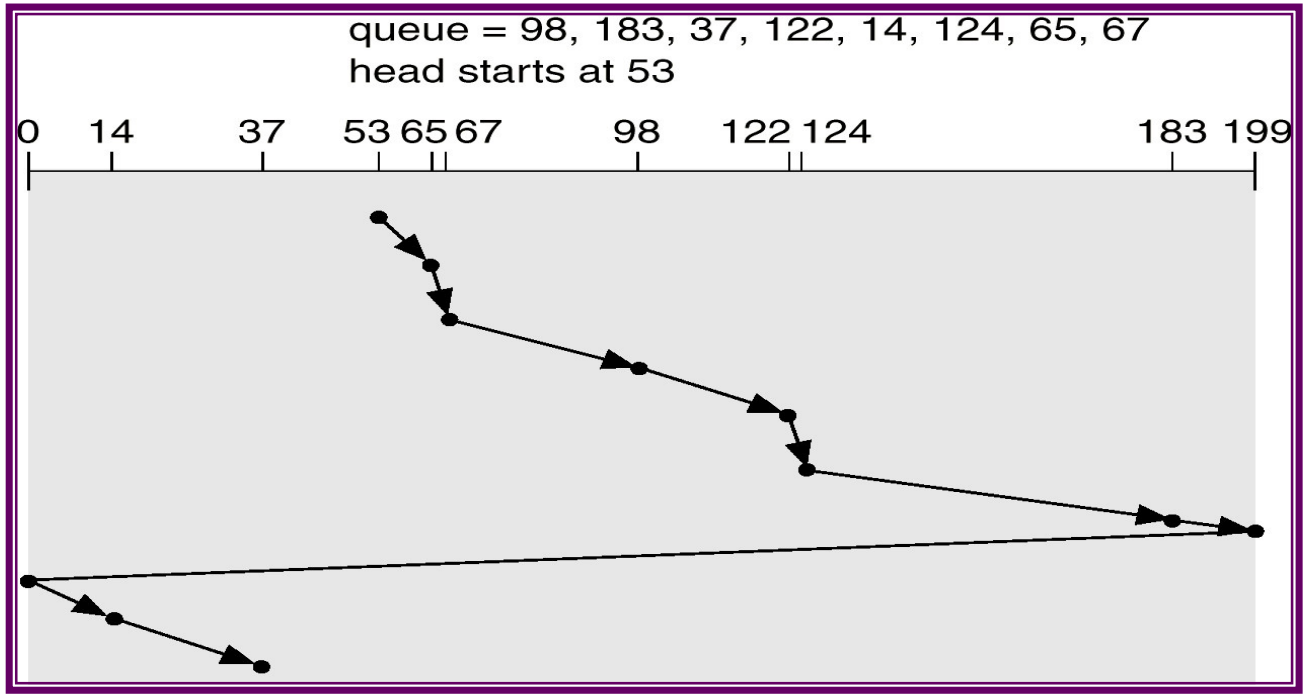
SCAN

- Head starts at one end of the disk, and moves toward the other end, servicing requests until it gets there. Then, the head starts moving in the opposite direction and servicing continues.
- Sometimes called the *elevator algorithm*.
- Easier on the hardware (less wear).
- In example, total head movement is 236 cylinders.



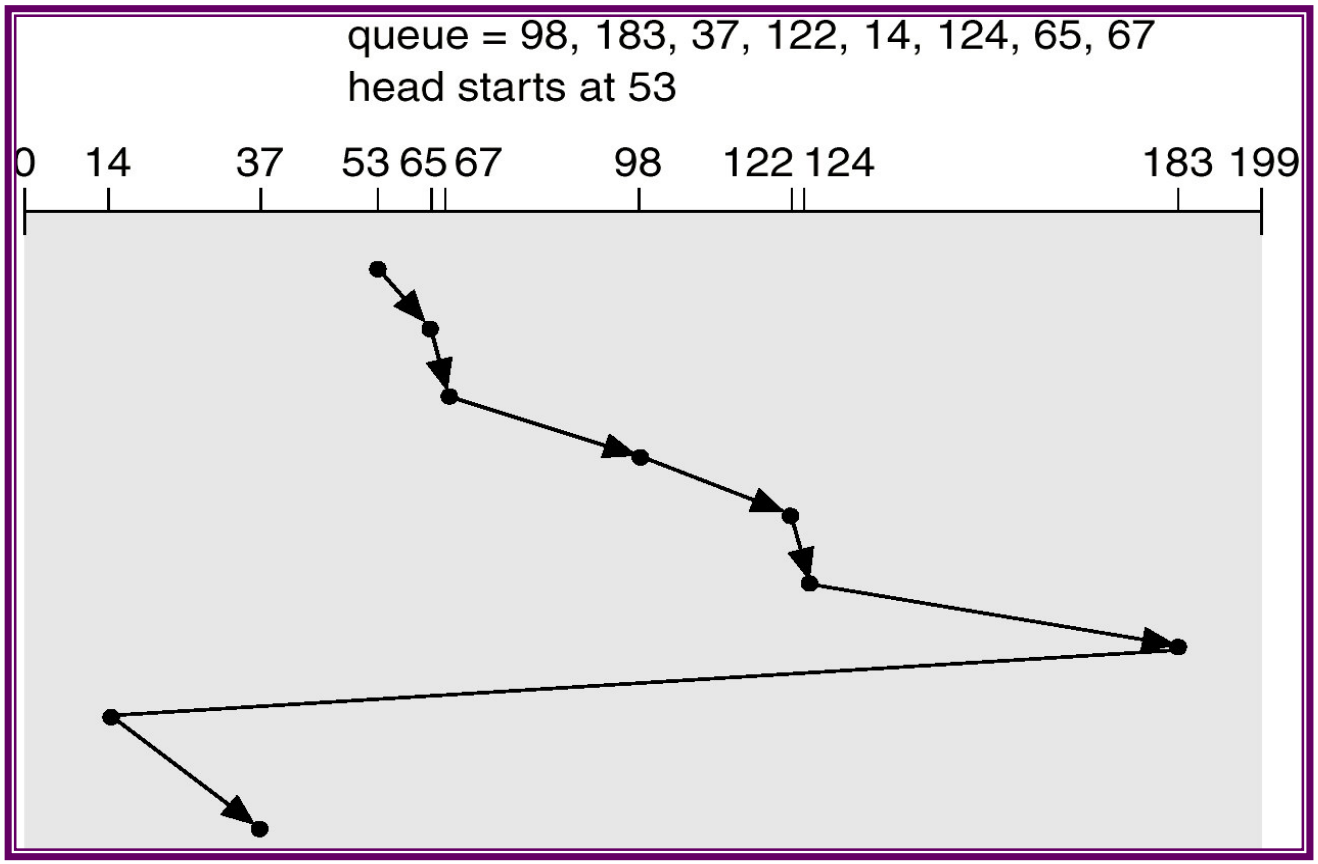
C-SCAN

- Head starts at first end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk. Then, the head returns to first end, without servicing requests on the return trip.
- Provides a more uniform wait time than SCAN: if going right and head is at cylinder 1 when request for cylinder 0 arrives, we'll travel 399 cylinders before we service it.
- "C" comes from thinking of cylinders as if in *circular* list.



C-LOOK

- Like C-SCAN, but head only goes as far as the last request in each direction.
- LOOK is like SCAN: service both ways, but turn if nothing appears if we *look* ahead.



Selecting a Disk-Scheduling Algorithm

- Either SSTF (most common) or LOOK is a reasonable choice for a default system algorithm.
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk (they are more fair).
- Performance depends on the number and types of requests. So the disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary, possibly while system is running.
- Requests for disk service can be influenced by the file-allocation method.

RAID Structure

- RAID: Redundant Array of Inexpensive/Independent Disks.
- With multiple disks, how can we store our data onto them? Want to balance:
  - Performance: fast I/O.
  - Reliability: avoid data loss if (part of) a drive fails.
- Ideal balance depends on system need. Hence many RAID levels.
- Software RAID (Windows 2000): unusably slow.
- Hardware RAID: <$300 for a 6-drive controller (must pay extra for memory).
- Assignment 4.

RAID Performance

- Given 3 identical disks, each with 3 blocks, provide abstraction of single disk with 9 blocks.
  - Basic block organization:

Disk 0	Disk 1	Disk 2
Block 0	Block 3	Block 6
Block 1	Block 4	Block 7
Block 2	Block 5	Block 8

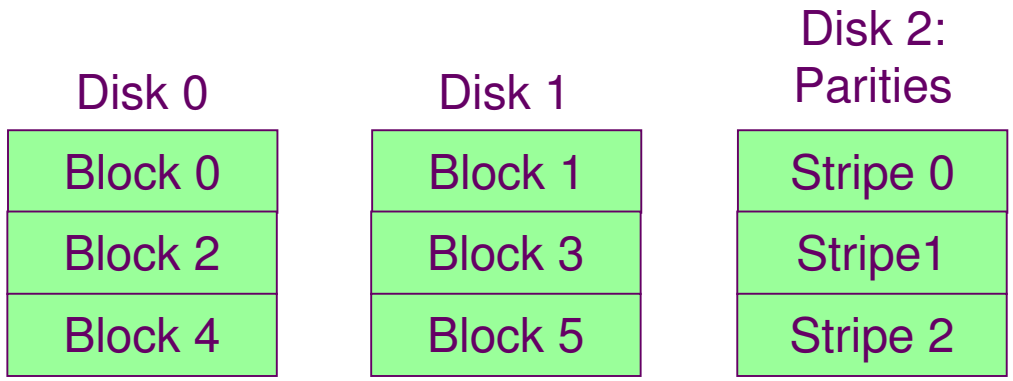
  - *Striped* block organization:

	Disk 0	Disk 1	Disk 2
Stripe 0 →	Block 0	Block 1	Block 2
Stripe 1 →	Block 3	Block 4	Block 5
Stripe 2 →	Block 6	Block 7	Block 8

  - If a file spans blocks 0-2, then the time to read file is
    - Basic: 3 x block I/O time.
    - Striped: block I/O time since all disks do I/O simultaneously.

RAID Reliability

- *Mirroring* or *shadowing*: keep a duplicate of each block somewhere else. Simplest is to copy each disk onto another.
- *Parity*: steal one block from each stripe. Store XOR (^) sum of all other stripe blocks in it.



A	B	P=A^B	P^A=B!	P^B=A!
0	0	0	0	0
0	1	1	1	0
1	0	1	0	1
1	1	0	1	1

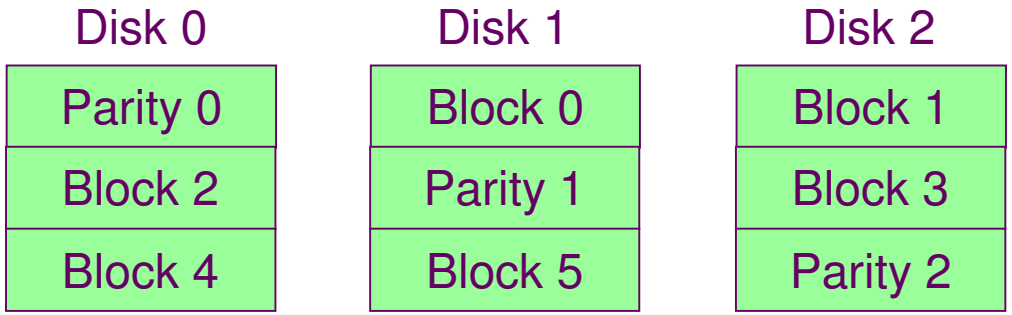
Hence if Disk 1 (B) fails, I can rebuild its blocks from Disks 0 (A) and 2 (P). Similarly for Disk 2 failing.

RAID Reliability (Cont.)

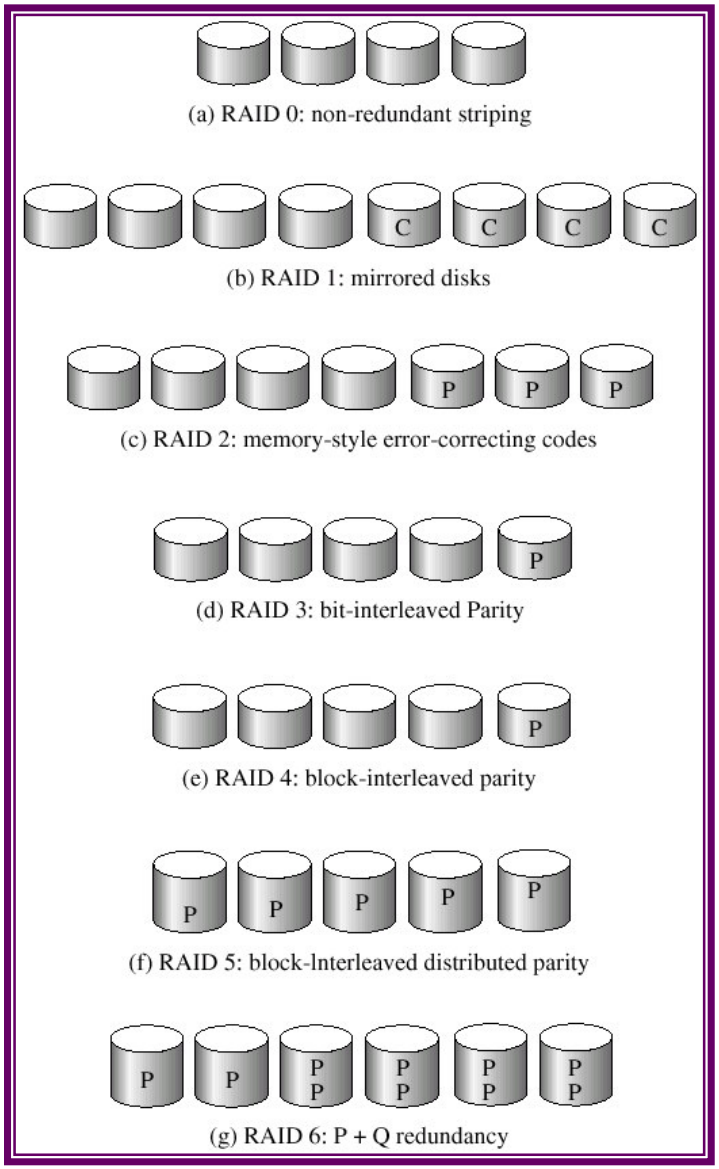
- Parity updates:
  - If Block 0 changes (A becomes A'), how do I update parity?
    - Slow: P' = A' ^ B ^ C ^ D...
    - Fast: P' = A' ^ B ^ C ^ D...
      = A' ^ (A ^ A) ^ B ^ C ^ D...
      = A' ^ A ^ P. ← Controller needs on-board memory to perform this computation.

Read-modify-write:
2 reads, 2 writes

- So parity disk participates in *every* write! To reduce overuse and premature failure, *distribute* parity:



RAID Levels



Striping only.

For each primary disk, a secondary disk is an exact copy of it (mirror). No striping.

Replace parity with memory style error-correcting codes: rarely used.

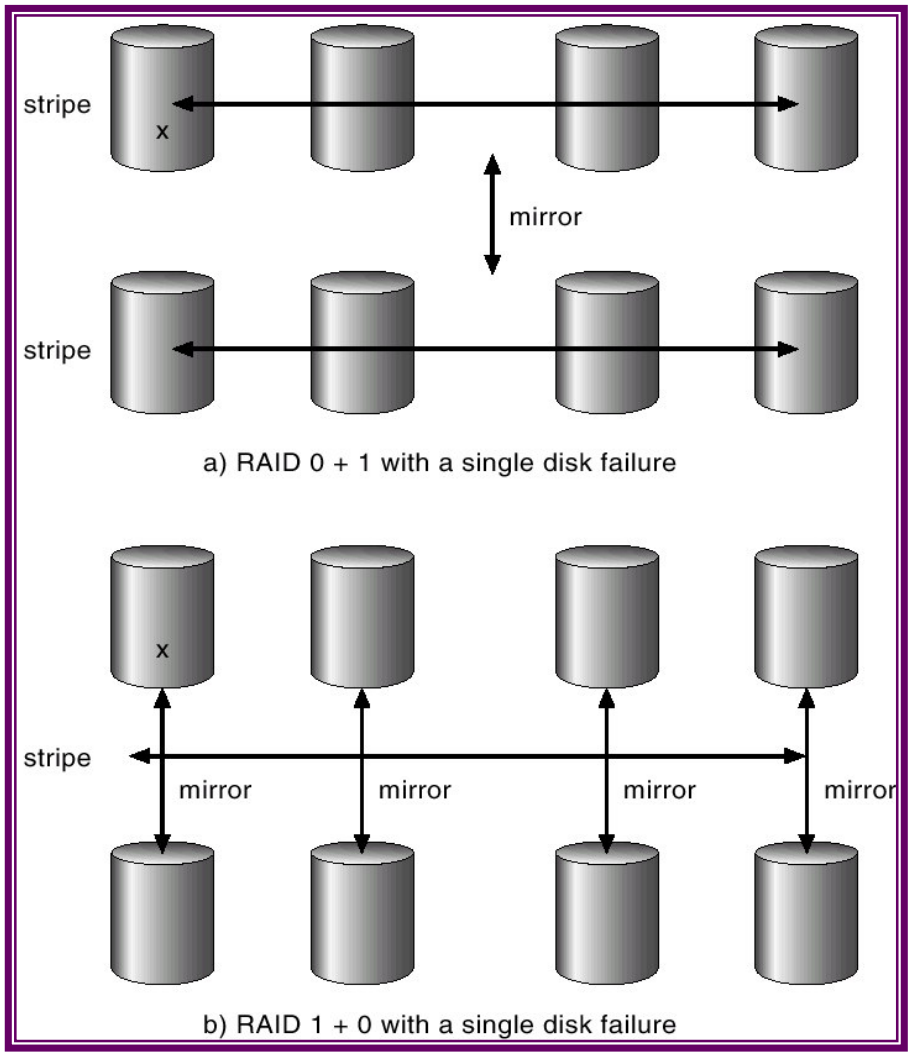
Parity with striping of bits, not blocks: rarely used.

Striping and parity. Parity on one disk.

Striping and parity. Distributed parity.

Striping and parity-like Reed-Solomon codes: protect against multiple disk failures. Uncommon.

RAID Levels (Cont.)



0+1: a stripe is mirrored, but individual disks may be different, e.g.

Disk 0A	Disk 1A	Disk 2A
Block 0	Block 1	Block 2
Block 5	Block 4	Block 3
Block 6	Block 7	Block 8

Disk 0B	Disk 1B	Disk 2B
Block 0	Block 1	Block 2
Block 3	Block 4	Block 5
Block 6	Block 7	Block 8

0A is not duplicate of 0B. If 0A fails, 0B cannot take its place: 1A, 2A become useless and no longer enhance RAID reliability.