

*Introduction to I/O and
Disk Management*

Secondary Storage Management

Disks — just like memory, only different

◆ Why have disks?

- Memory is small. Disks are large.
 - ❖ Short term storage for memory contents (e.g., swap space).
 - ❖ Reduce what must be kept in memory (e.g., code pages).
- Memory is volatile. Disks are forever (?!)
 - ❖ File storage.



	GB/dollar	dollar/GB
RAM	0.013(0.015,0.01)	\$77(\$68,\$95)
Disks	3.3(1.4,1.1)	30¢ (71¢,90¢)

Capacity : 2GB vs. 1TB

2GB vs. 400GB

1GB vs 320GB



How to approach persistent storage

- ◆ Disks first, then file systems.
 - Bottom up.
 - Focus on device characteristics which dominate performance or reliability (they become focus of SW).
- ◆ Disk capacity (along with processor performance) are the crown jewels of computer engineering.
- ◆ File systems have won, but at what cost victory?
 - Ipod, iPhone, TivO, PDAs, laptops, desktops all have file systems.
 - Google is made possible by a file system.
 - File systems rock because they are:
 - ❖ Persistent.
 - ❖ Heirarchical (non-cyclical (mostly)).
 - ❖ Rich in metadata (remember cassette tapes?)
 - ❖ Indexible (hmmm, a weak point?)
- ◆ The price is complexity of implementation.

Different types of disks

- ◆ **Advanced Technology Attachment (ATA)**
 - Standard interface for connecting storage devices (e.g., hard drives and CD-ROM drives)
 - Referred to as IDE (Integrated Drive Electronics), ATAPI, and UDMA.
 - ATA standards only allow cable lengths in the range of 18 to 36 inches. CHEAP.
- ◆ **Small Computer System Interface (SCSI)**
 - Requires controller on computer and on disk.
 - Controller commands are sophisticated, allow reordering.
- ◆ **USB or Firewire connections to ATA disc**
 - These are new bus technologies, not new control.
- ◆ **Microdrive – impressively small motors**

Different types of disks

- ◆ Bandwidth ratings.
 - These are unachievable.
 - 50 MB/s is max off platters.
 - Peak rate refers to transfer from disc device's memory cache.
- ◆ SATA II (serial ATA)
 - 3 Gb/s (still only 50 MB/s off platter, so why do we care?)
 - Cables are smaller and can be longer than pATA.
- ◆ SCSI 320 MB/s
 - Enables multiple drives on same bus

Mode	Speed
UDMA0	16.7 MB/s
UDMA1	25.0 MB/s
UDMA2	33.3 MB/s
UDMA3	44.4 MB/s
UDMA4	66.7 MB/s
UDMA5	100.0 MB/s
UDMA6	133 MB/s

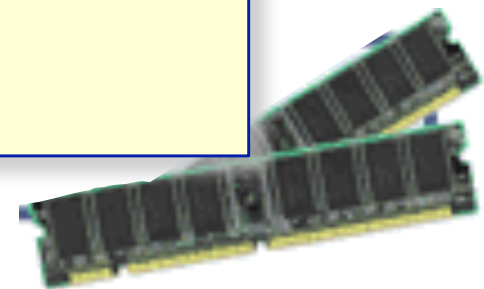
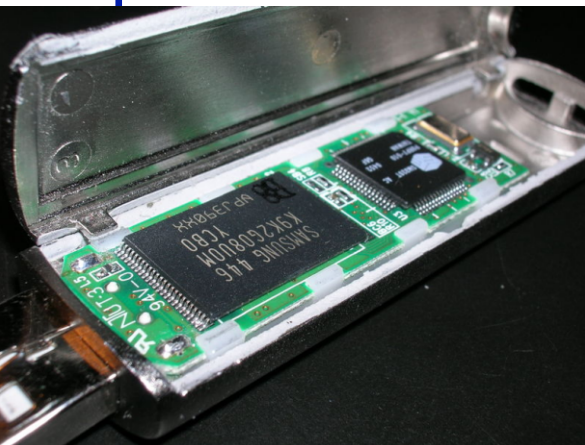
Flash: An upcoming technology

◆ Flash memory gaining popularity

- iPhone has 128GB flash (no disk)
- Windows supports Flash as accelerator
- Future is hybrid flash/disk or just flash?
- Erased a block at a time (100,000 write-erase-cycles)
- Pages are 512 bytes or 2,048 bytes
- Read 256 MB/s, write 128MB/s
- Lower power than (spinning) disk

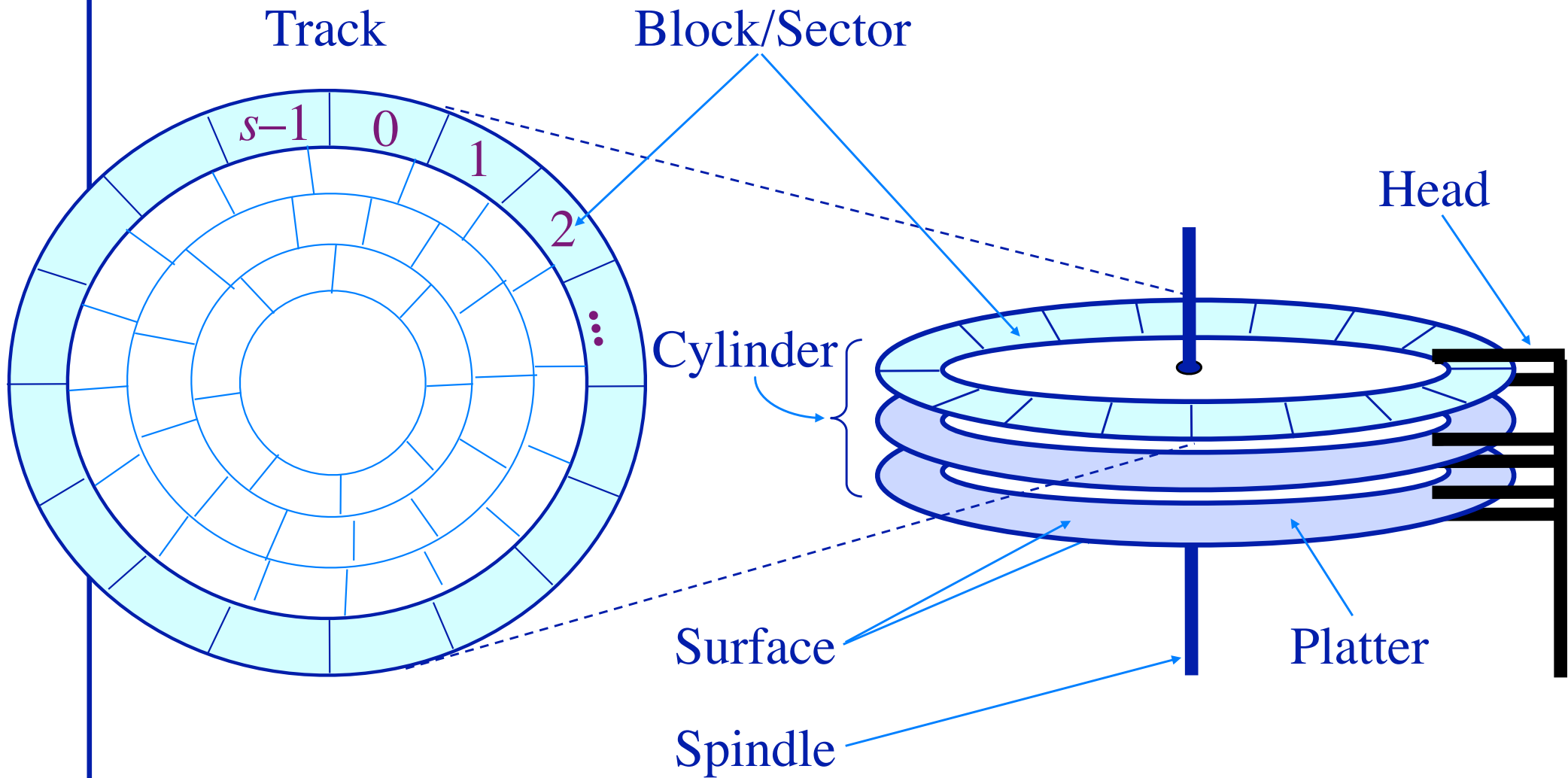


	GB/dollar	dollar/GB
RAM	0.013(0.015,0.01)	\$77(\$68,\$95)
Disks	3.3 (1.4,1.1)	30¢ (71¢,90¢)
Flash	0.1	\$10

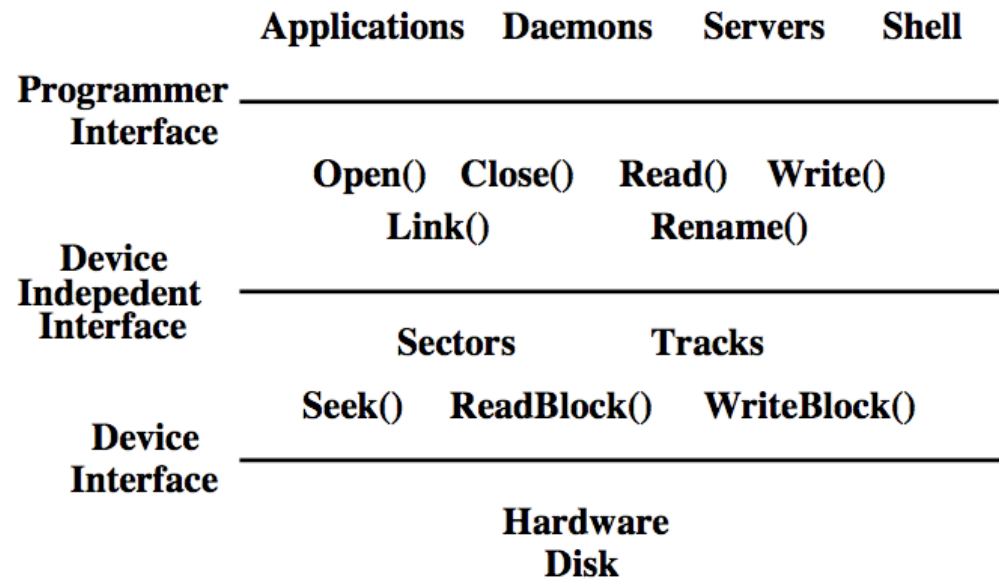


Anatomy of a Disk

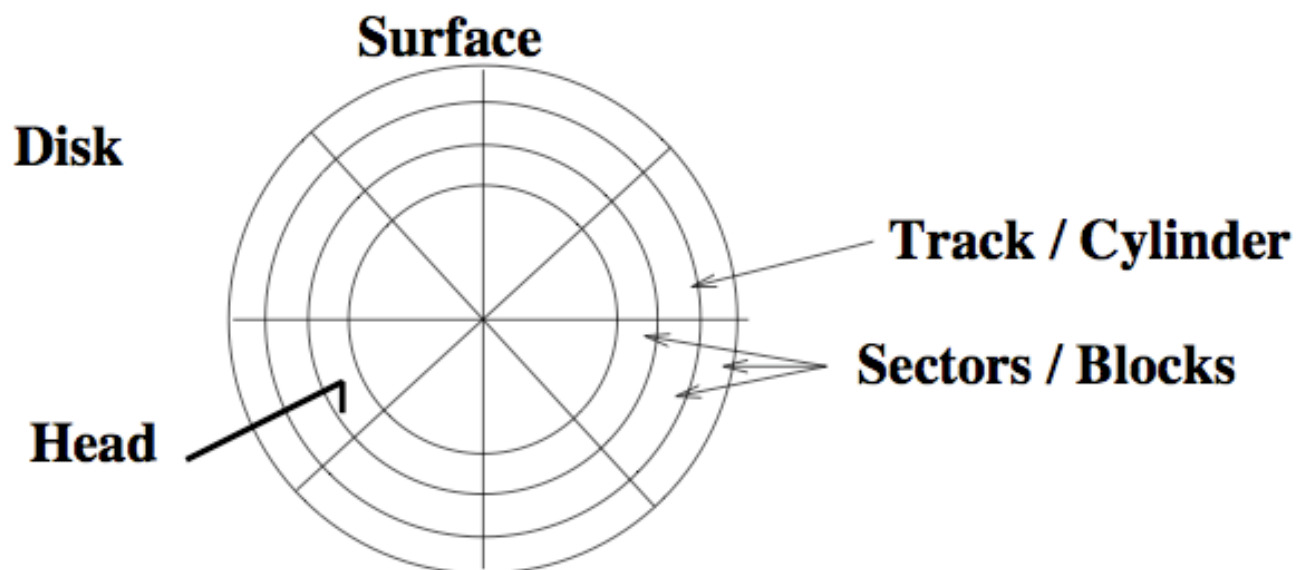
Basic components



Disk structure: the big picture



◆ Physical structure of disks

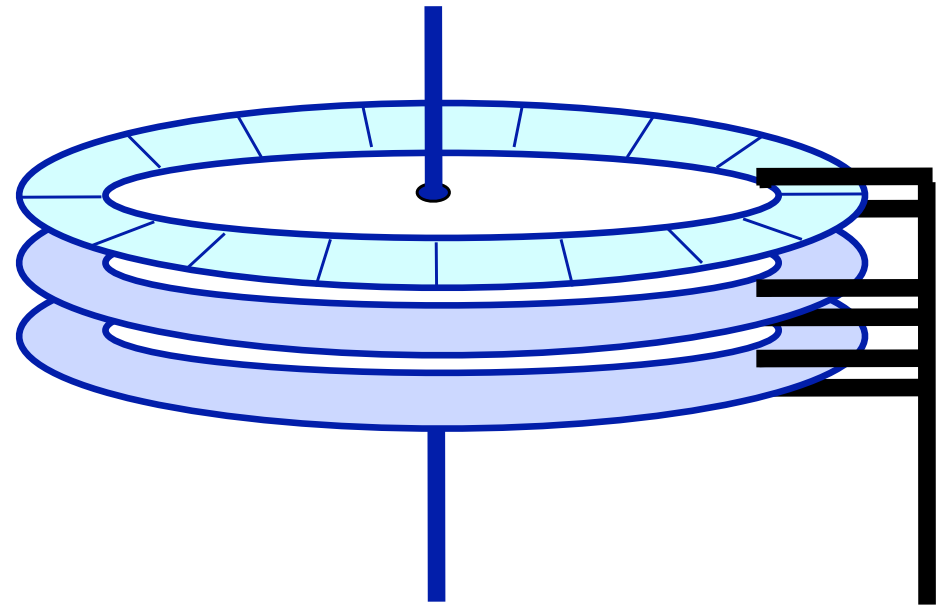


Anatomy of a Disk

Seagate 73.4 GB Fibre Channel Ultra 160 SCSI disk

◆ Specs:

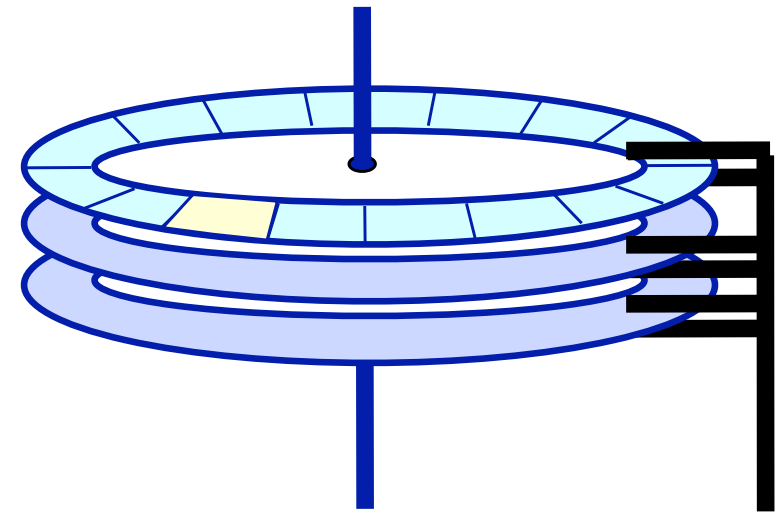
- 12 Platters
- 24 Heads
- Variable # of sectors/track
- 10,000 RPM
 - ❖ Average latency: 2.99 *ms*
- Seek times
 - ❖ Track-to-track: 0.6/0.9 *ms*
 - ❖ Average: 5.6/6.2 *ms*
 - ❖ Includes acceleration and settle time.
- 160-200 MB/s peak transfer rate
 - ❖ 1-8K cache
- 12 Arms
- 14,100 Tracks
- 512 bytes/sector



Disk Operations

Read/Write operations

- ◆ Present disk with a sector address
 - Old: $DA = (drive, surface, track, sector)$
 - New: Logical block address (LBA)
- ◆ Heads moved to appropriate track
 - seek time
 - settle time
- ◆ The appropriate head is enabled
- ◆ Wait for the sector to appear under the head
 - “rotational latency”
- ◆ Read/write the sector
 - “transfer time”



Read time:
 $seek\ time + latency + transfer\ time$
 $(5.6\ ms + 2.99\ ms + 0.014\ ms)$

Disk access latency

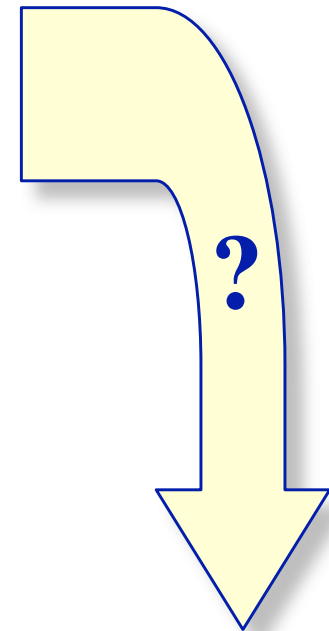
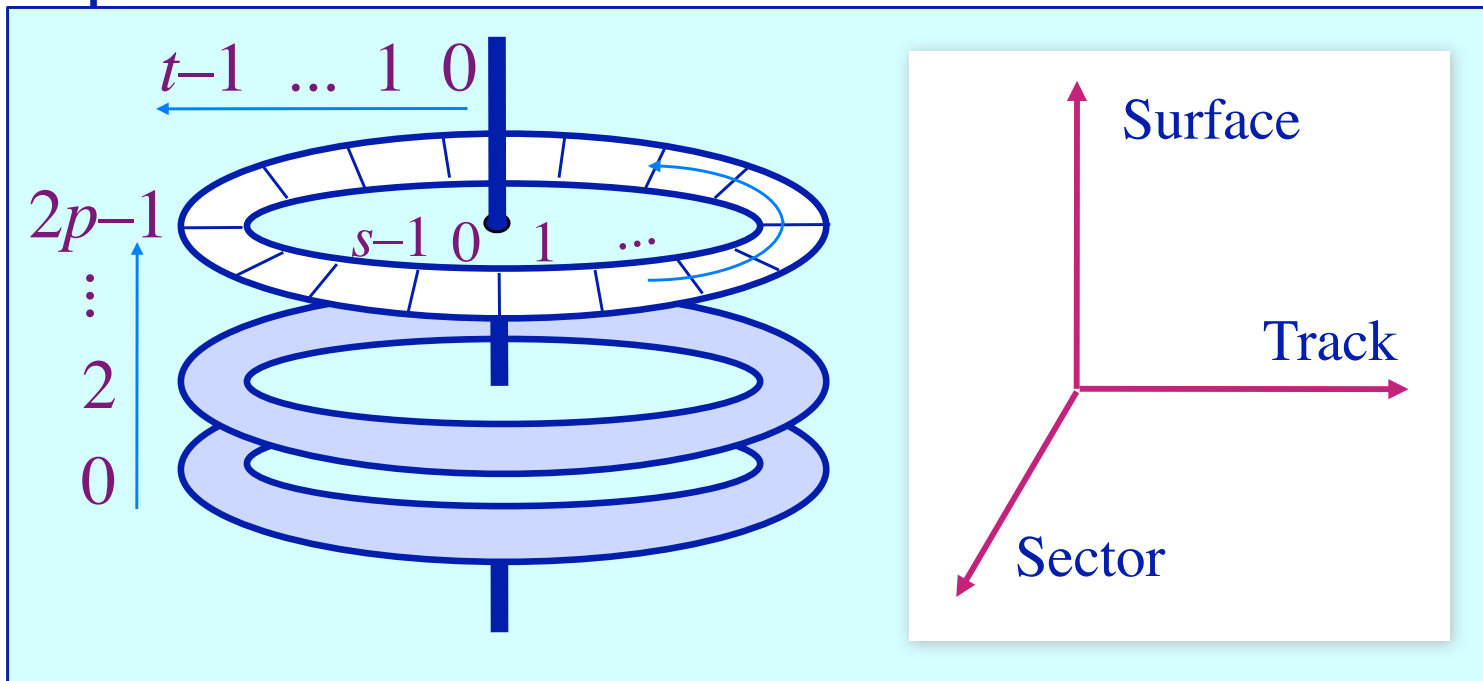
- ◆ Which component of disk access time is the longest?
 - A. Rotational latency
 - B. Transfer latency
 - C. Seek latency

Disk Addressing

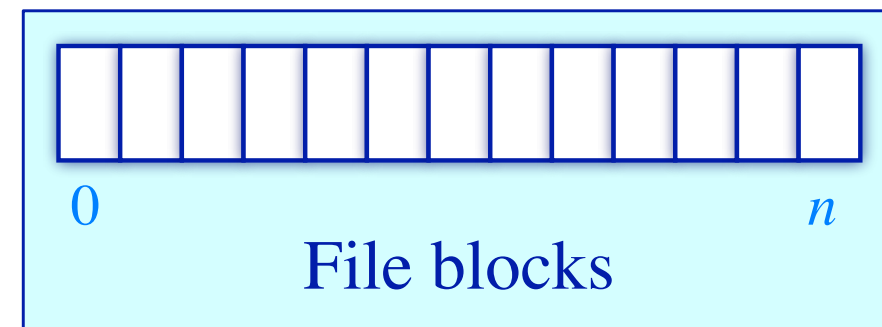
- ◆ Software wants a simple “disc virtual address space” consisting of a linear array of sectors.
 - Sectors numbered 1..N, each 512 bytes (typical size).
 - Writing 8 surfaces at a time writes a 4KB page.
- ◆ Hardware has structure:
 - Which platter?
 - Which track within the platter?
 - Which sector within the track?
- ◆ The hardware structure affects latency.
 - Reading from sectors in the same track is fast.
 - Reading from the same cylinder group is faster than seeking.

Disk Addressing

Mapping a 3-D structure to a 1-D structure



- ◆ Mapping criteria
 - block $n+1$ should be as "close" as possible to block n



The Impact of File Mappings

File access times: Contiguous allocation

- ◆ Array elements map to contiguous sectors on disk
 - Case1: Elements map to the middle of the disk

$$\underbrace{5.6}_{\text{Seek Time}} + \underbrace{3.0}_{\text{Latency}} + \underbrace{6.0 \frac{2,048}{424}}_{\text{Transfer Time}} = 8.6 + 29.0 = 37.6 \text{ ms}$$

$\underbrace{\text{Seek Time} + \text{Latency}}_{\text{Constant Terms}} = \underbrace{\text{Transfer Time}}_{\text{Variable Term}} = \left[\text{time per revolution} \right] \times \left[\text{number of revolutions required to transfer data} \right]$

The Impact of File Mappings

File access times: Contiguous allocation

- ◆ Array elements map to contiguous sectors on disk
 - Case1: Elements map to the middle tracks of the platter

$$5.6 + 3.0 + 6.0 \frac{2,048}{424} = 8.6 + 29.0 = 37.6 \text{ ms}$$

Case2: Elements map to the inner tracks of the platter

$$5.6 + 3.0 + 6.0 \frac{2,048}{212} = 8.6 + 58.0 = 66.6 \text{ ms}$$

Case3: Elements map to the outer tracks of the platter

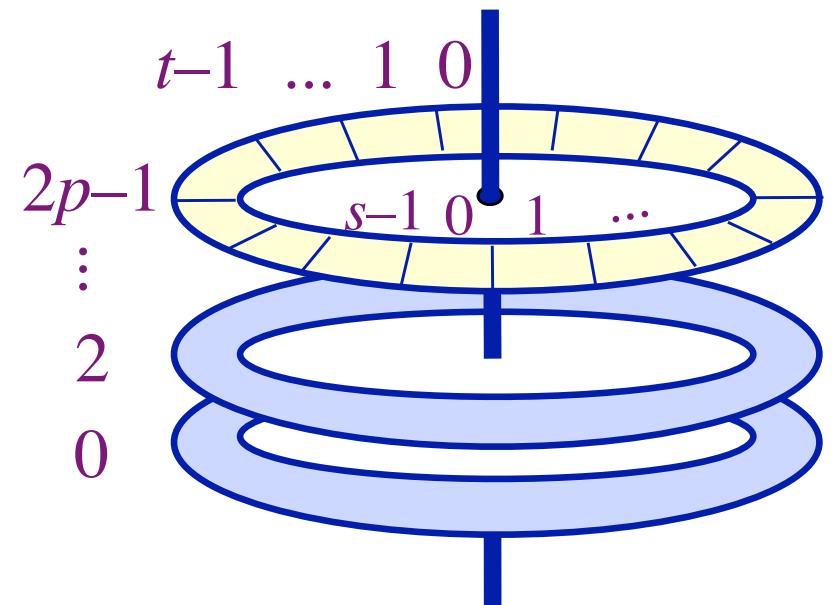
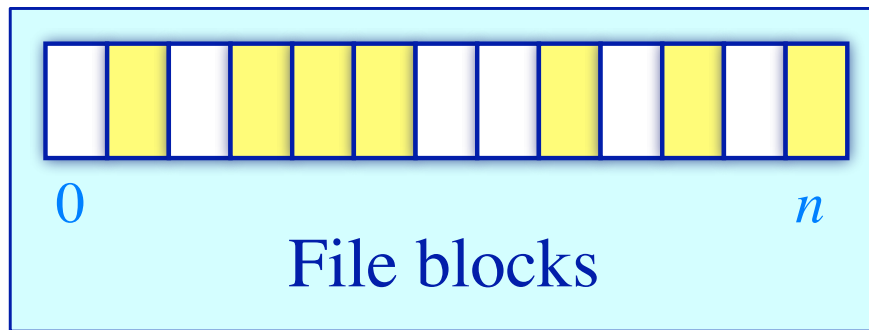
$$5.6 + 3.0 + 6.0 \frac{2,048}{636} = 8.6 + 19.3 = 27.9 \text{ ms}$$

Disk Addressing

The impact of file mappings: Non-contiguous allocation

- ◆ Array elements map to random sectors on disk
 - Each sector access results in a disk seek

$$2,048 \times (5.6 + 3.0) = 17.6 \text{ seconds}$$



Practical Knowledge

- ◆ If the video you are playing off your hard drive skips, defragment your file system.
- ◆ OS block allocation policy is complicated. Defragmentation allows the OS to revisit layout with global information.
- ◆ Unix file systems need defragmentation less than Windows file systems, because they have better allocation policies.

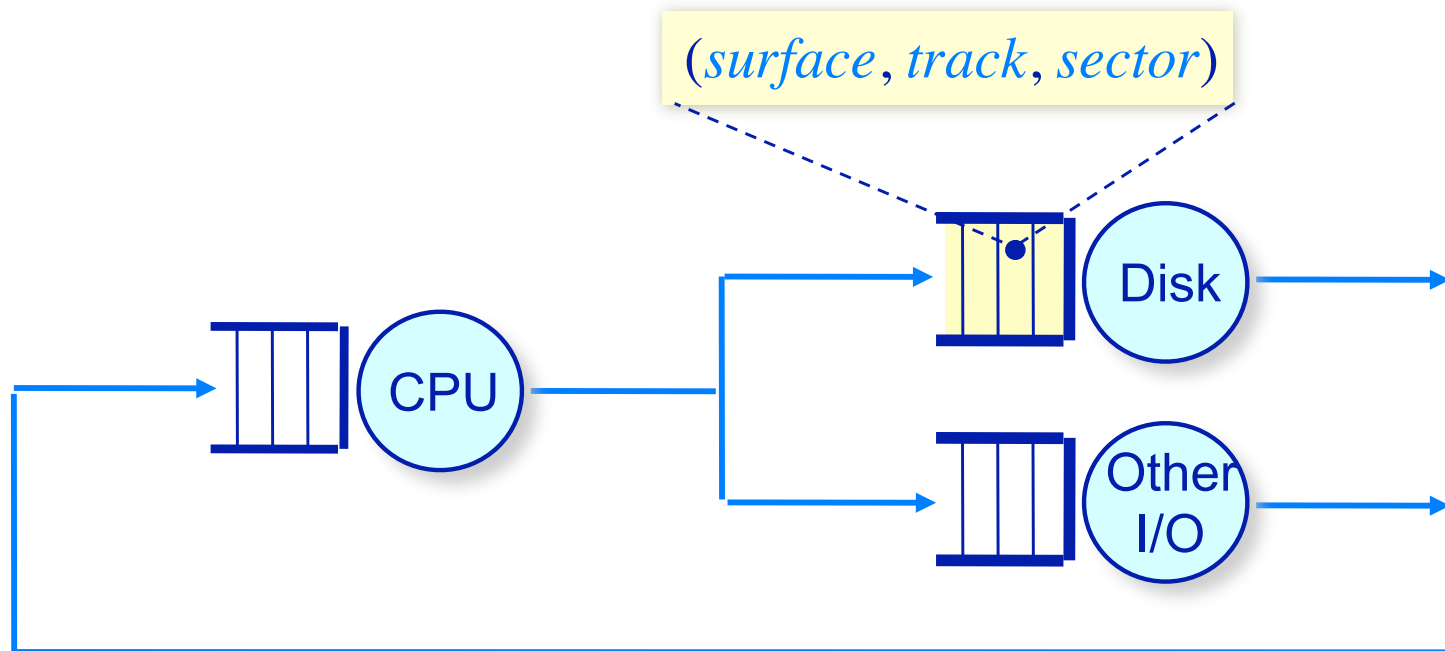
Defragmentation Decisions

- ◆ Files written when the disk is nearly full are more likely to be fragmented.
 - A. True
 - B. False

Disk Head Scheduling

Maximizing disk throughput

- ◆ In a multiprogramming/timesharing environment, a queue of disk I/O requests can form



The OS maximizes disk I/O throughput by minimizing head movement through **disk head scheduling**

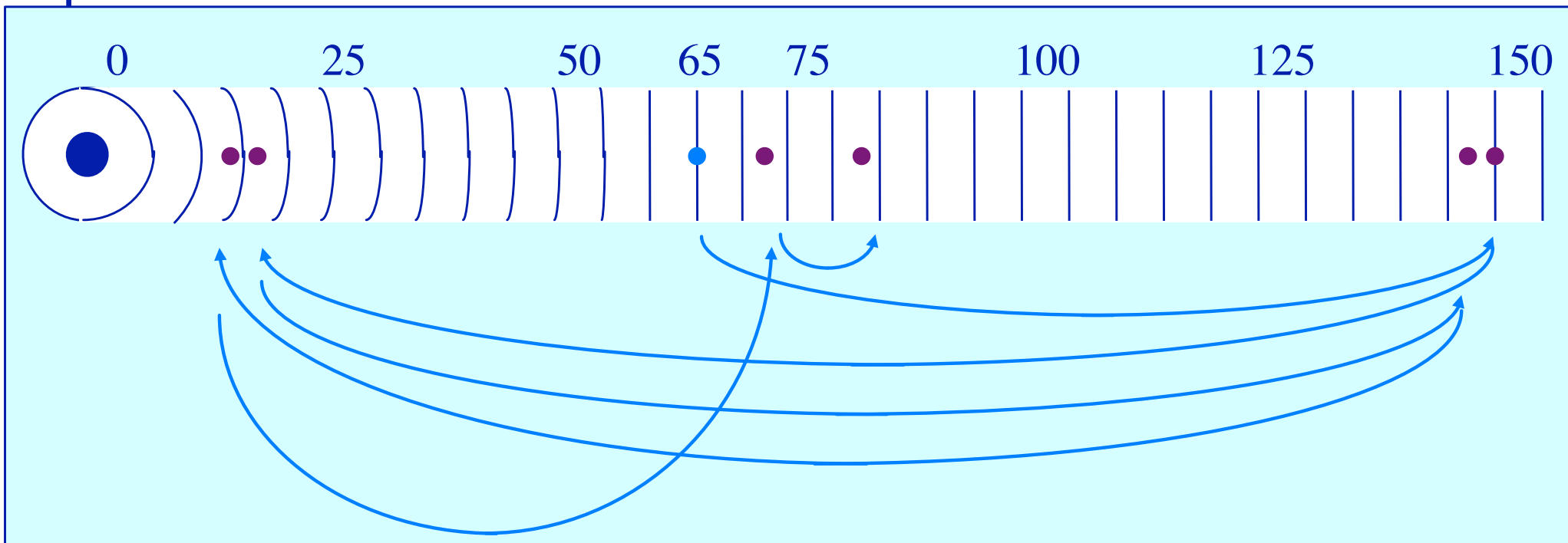
Disk Head Scheduling

Examples

- ◆ Assume a queue of requests exists to read/write tracks:

83	72	14	147	16	150
----	----	----	-----	----	-----

 and the head is on track 65



FCFS scheduling results in the head moving 550 tracks

Can we do better?

Disk Head Scheduling

Minimizing head movement

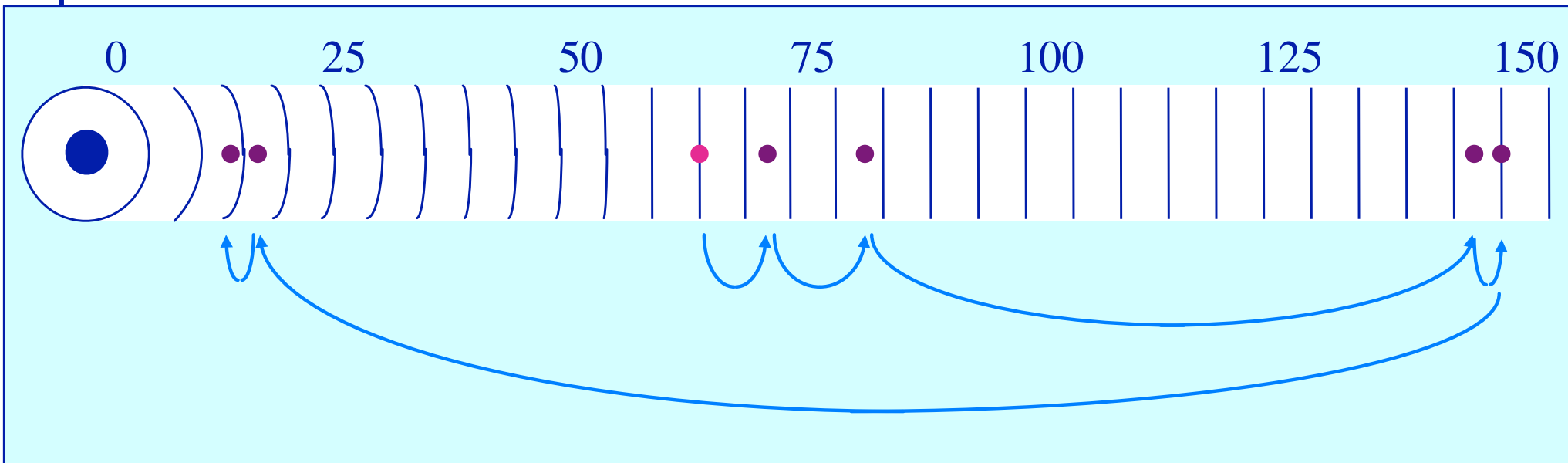
- ◆ Greedy scheduling: *shortest seek time first*

➤ Rearrange queue from:

83	72	14	147	16	150
----	----	----	-----	----	-----

To:

14	16	150	147	82	72
----	----	-----	-----	----	----



SSTF scheduling results in the head moving 221 tracks
Can we do better?

Disk Head Scheduling

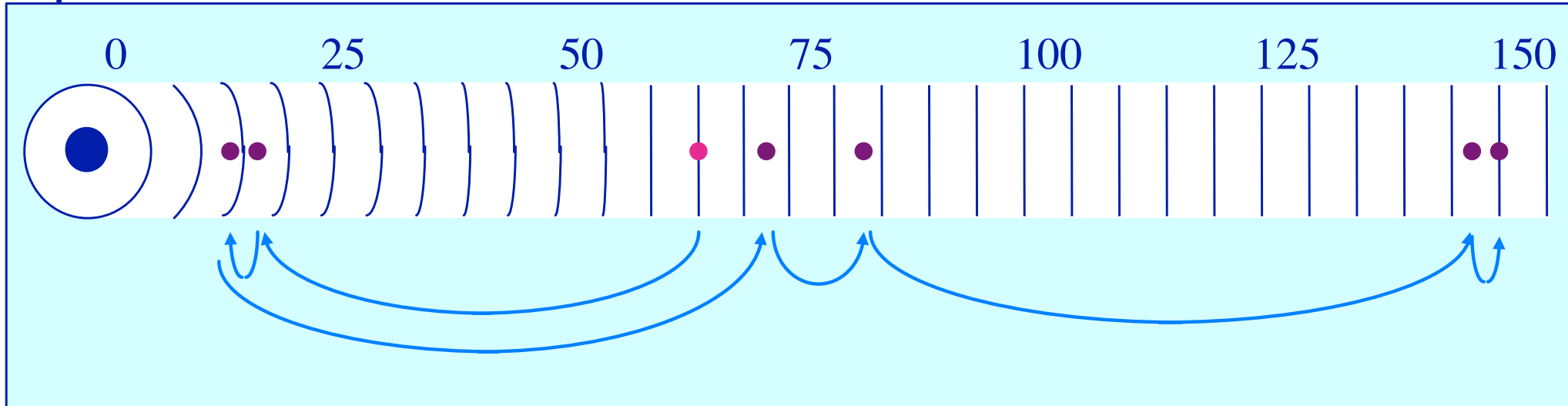
SCAN scheduling

◆ Rearrange queue from:

83	72	14	147	16	150
----	----	----	-----	----	-----

To:

150	147	83	72	14	16
-----	-----	----	----	----	----



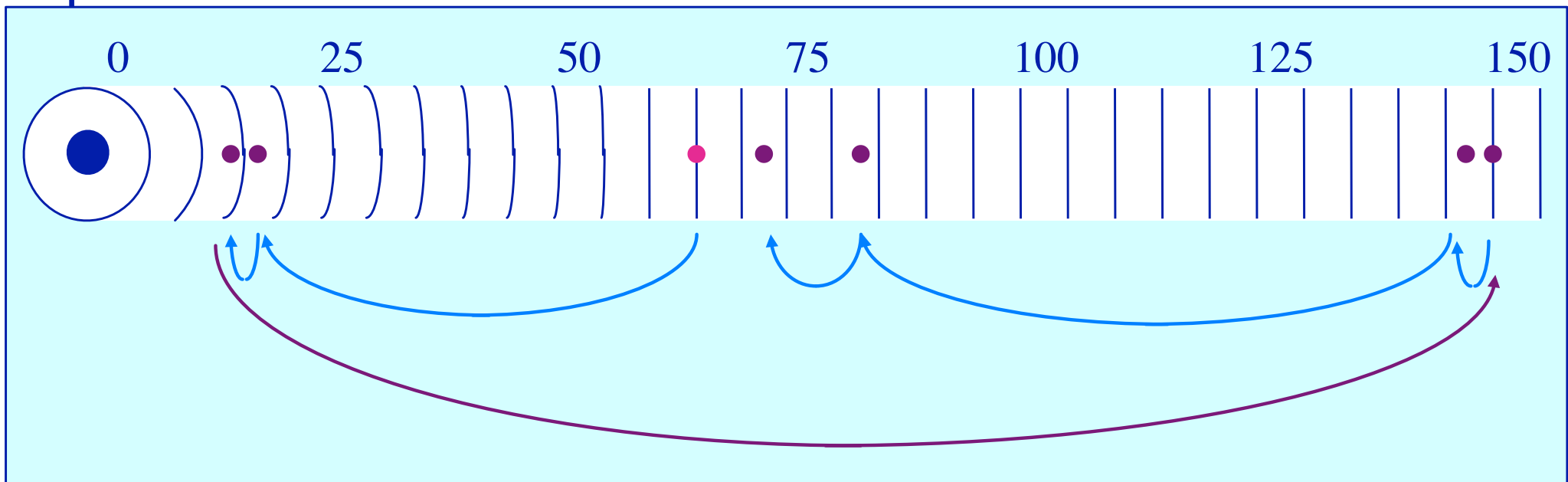
"SCAN" scheduling: Move the head in one direction until all requests have been serviced and then reverse. Also called elevator scheduling.

Moves the head 187 tracks

Disk Head Scheduling

Other variations

- ◆ C-SCAN scheduling (“Circular”-SCAN)
 - Move the head in one direction until an edge of the disk is reached and then reset to the opposite edge



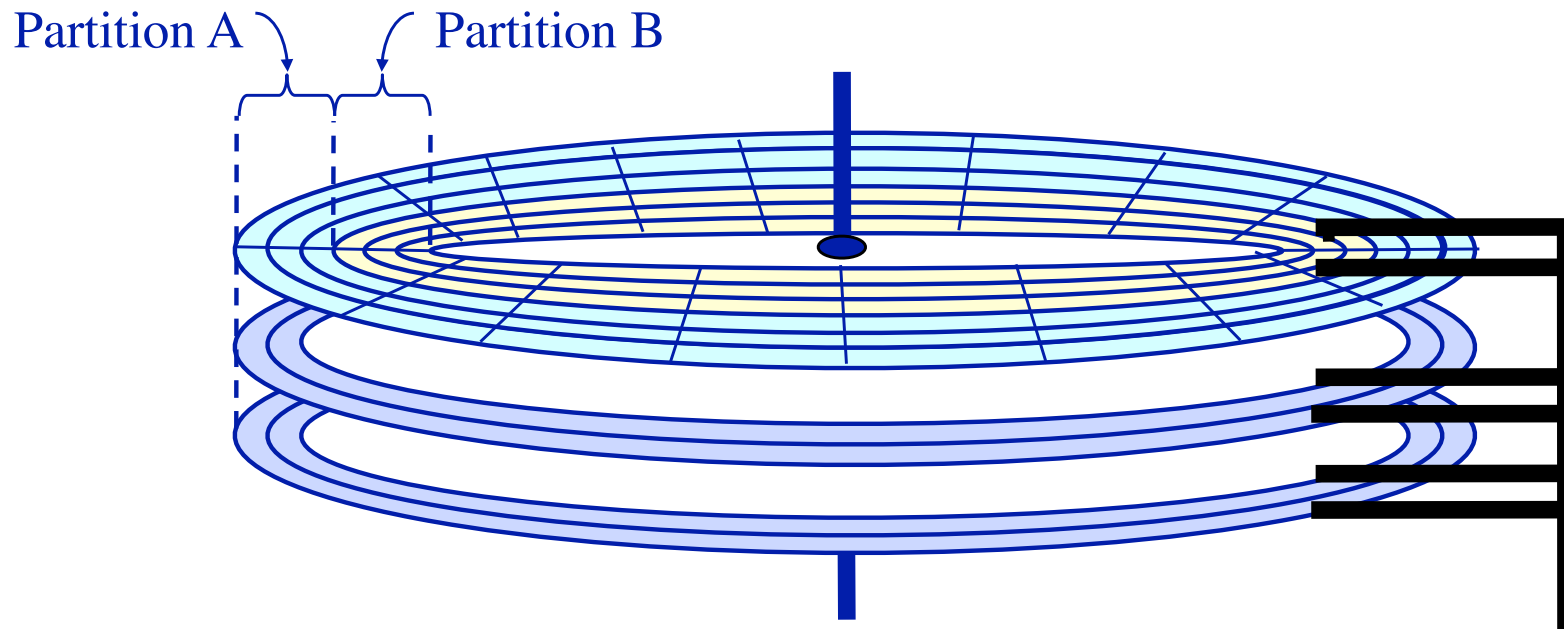
LOOK scheduling

Same as C-SCAN except the head is reset when no more requests exist between the current head position and the approaching edge of the disk

Disk Performance

Disk partitioning

- ◆ Disks are typically partitioned to minimize the largest possible seek time
 - A partition is a collection of cylinders
 - Each partition is a logically separate disk



Disks – Technology Trends

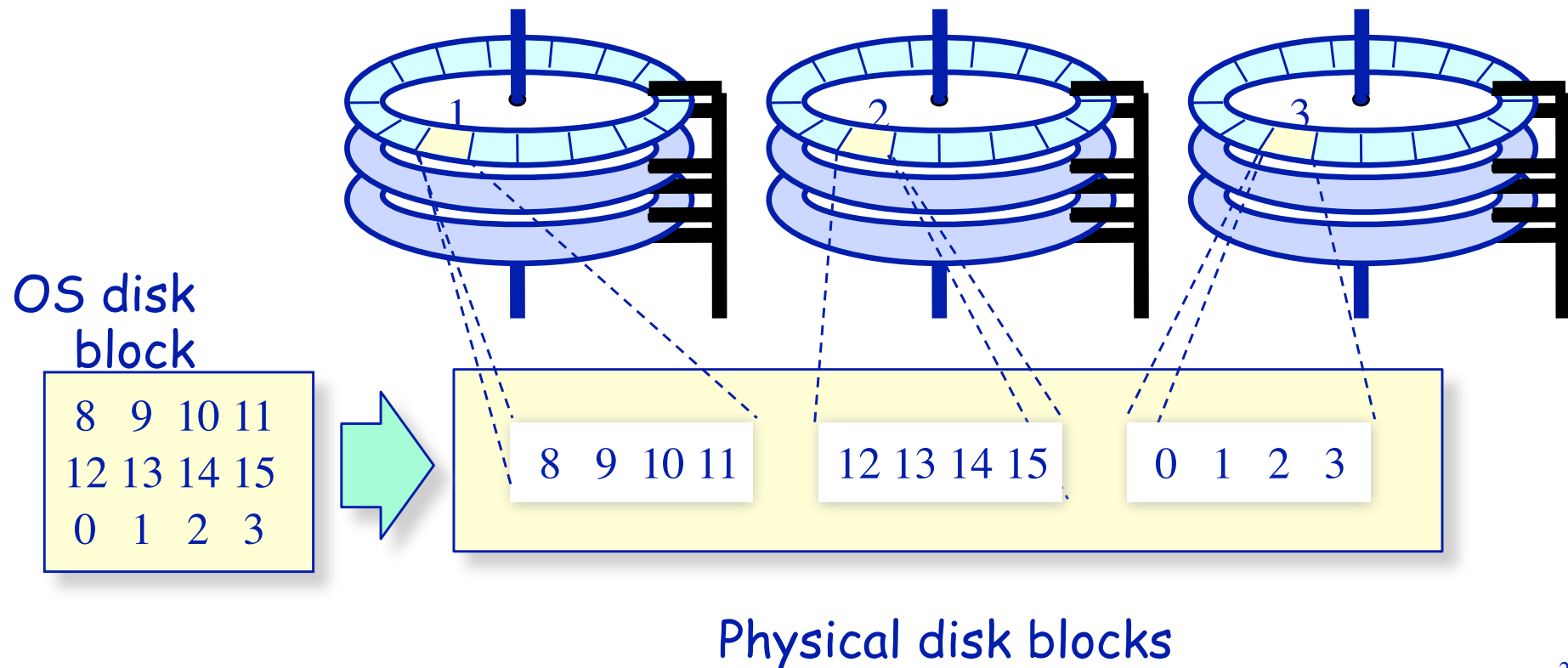
- ◆ Disks are getting smaller in size
 - Smaller → spin faster; smaller distance for head to travel; and lighter weight
- ◆ Disks are getting denser
 - More bits/square inch → small disks with large capacities
- ◆ Disks are getting cheaper
 - 2x/year since 1991
- ◆ Disks are getting faster
 - Seek time, rotation latency: 5-10%/year (2-3x per decade)
 - Bandwidth: 20-30%/year (~10x per decade)
- ◆ Overall:
 - Disk capacities are improving much faster than performance

Management of Multiple Disks

Using multiple disks to increase disk throughput

◆ Disk striping (RAID-0)

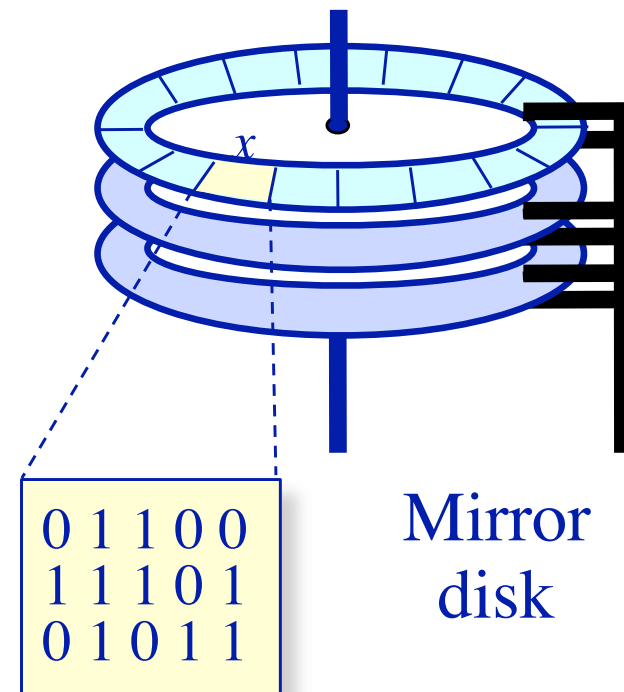
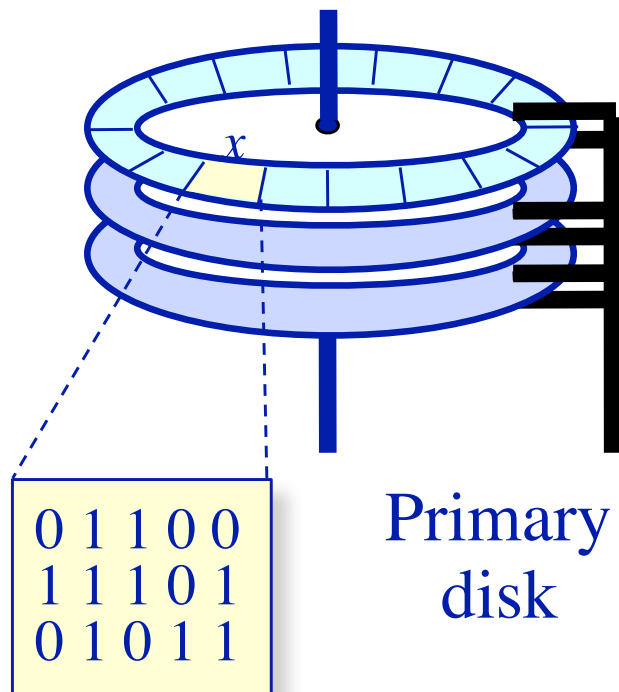
- Blocks broken into sub-blocks that are stored on separate disks
 - ❖ similar to memory interleaving
- Provides for higher disk bandwidth through a larger effective block size



Management of Multiple Disks

Using multiple disks to improve reliability & availability

- ◆ To increase the reliability of the disk, redundancy must be introduced
 - Simple scheme: *disk mirroring (RAID-1)*
 - *Write to both disks, read from either.*



Who controls the RAID?

◆ Hardware

- +Tend to be reliable (hardware implementers test)
- +Offload parity computation from CPU
 - ❖ Hardware is a bit faster for rewrite intensive workloads
- -Dependent on card for recovery (replacements?)
- -Must buy card (for the PCI bus)
- -Serial reconstruction of lost disk

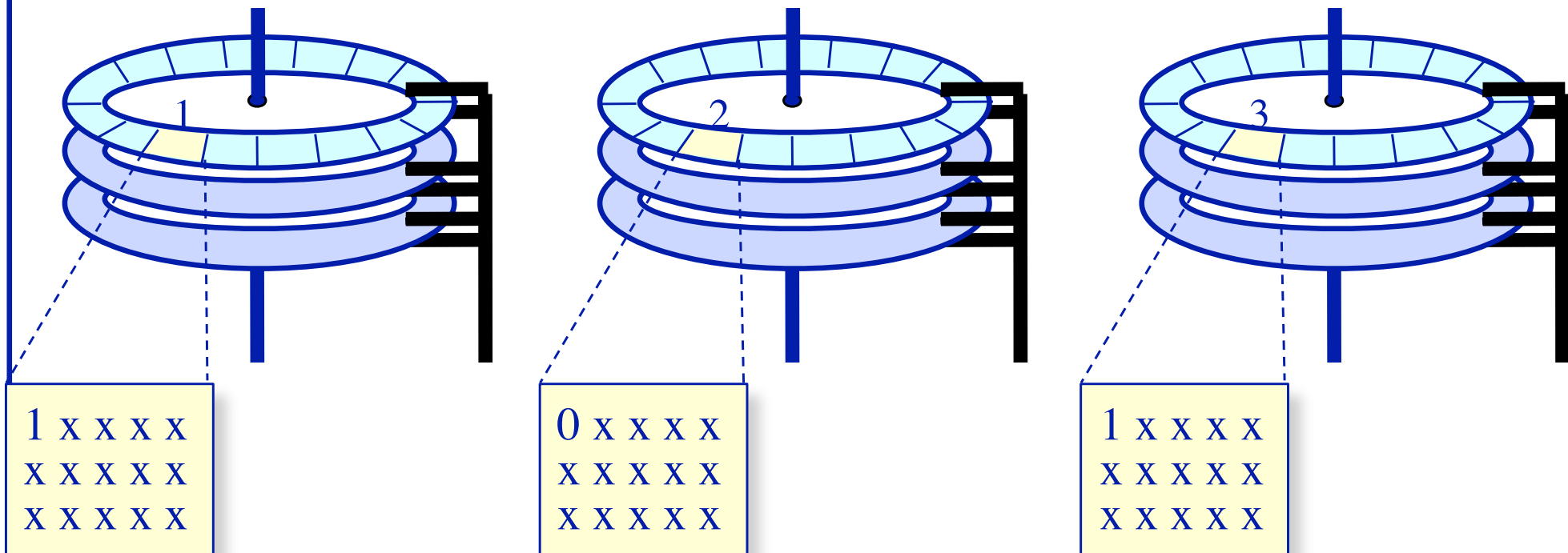
◆ Software

- -Software has bugs
- -Ties up CPU to compute parity
- +Other OS instances might be able to recover
- +No additional cost
- +Parallel reconstruction of lost disk

Management of Multiple Disks

Using multiple disks to increase disk throughput

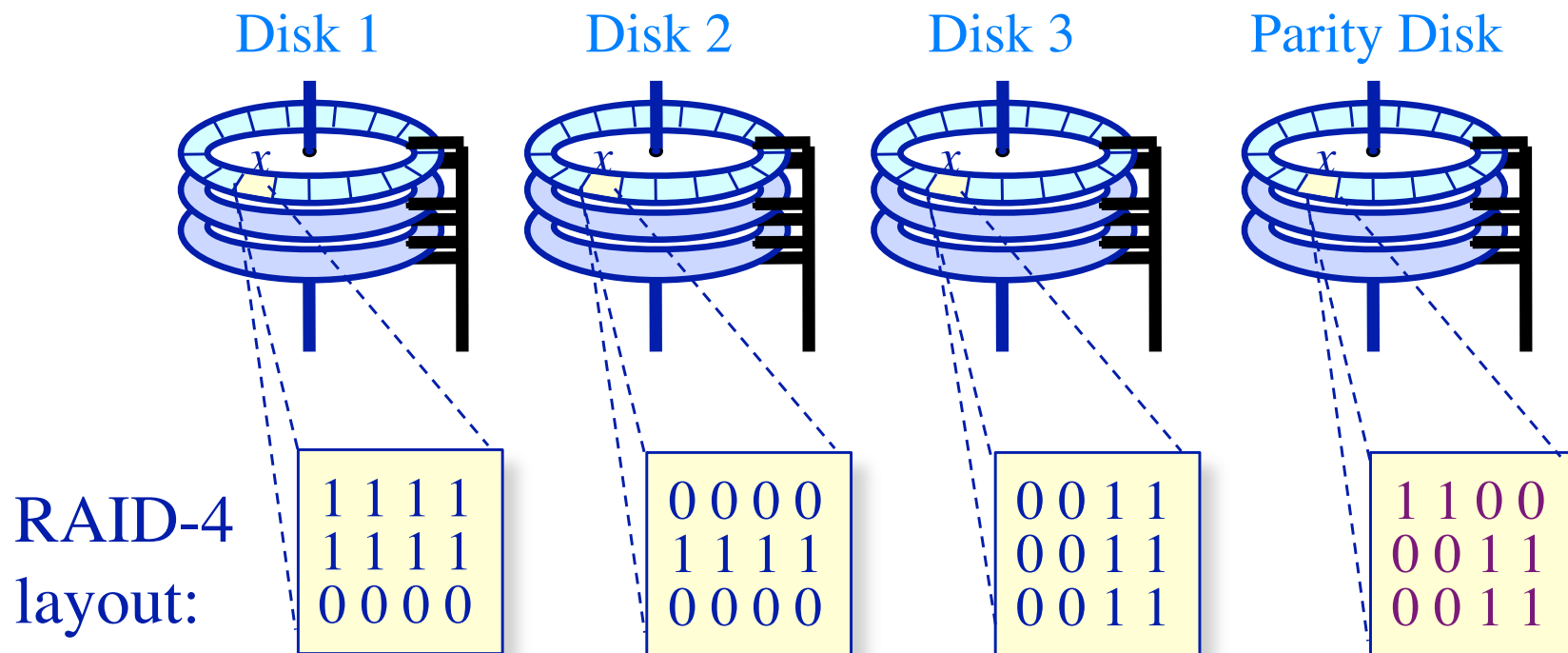
- ◆ RAID (*redundant array of inexpensive disks*) disks
 - Byte-wise striping of the disks (RAID-3) or block-wise striping of the disks (RAID-0/4/5)
 - Provides better performance and reliability
- ◆ Example: storing the byte-string 101 in a RAID-3 system



Improving Reliability and Availability

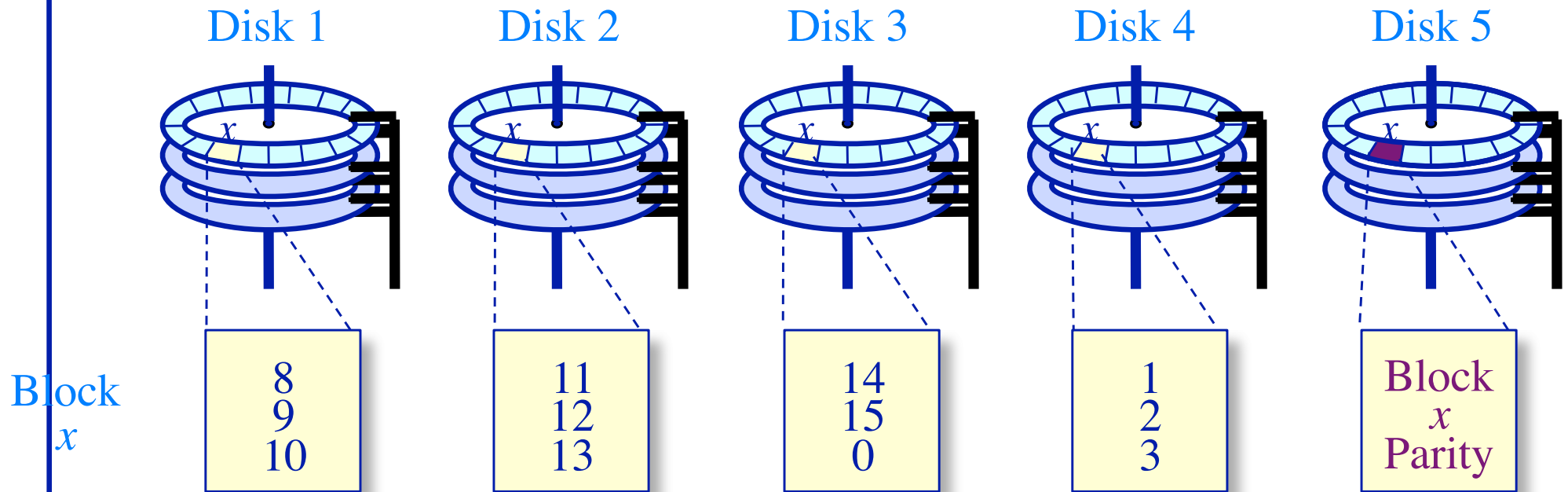
RAID-4

- ◆ Block interleaved parity striping
 - Allows one to recover from the crash of any one disk
 - Example: storing 8, 9, 10, 11, 12, 13, 14, 15, 0, 1, 2, 3



Improving Reliability and Availability

RAID-5 Block interleaved parity striping



Improving Reliability and Availability

RAID-5 Block interleaved parity striping

