Disks and RAID

Profs. Bracy and Van Renesse

based on slides by Prof. Sirer

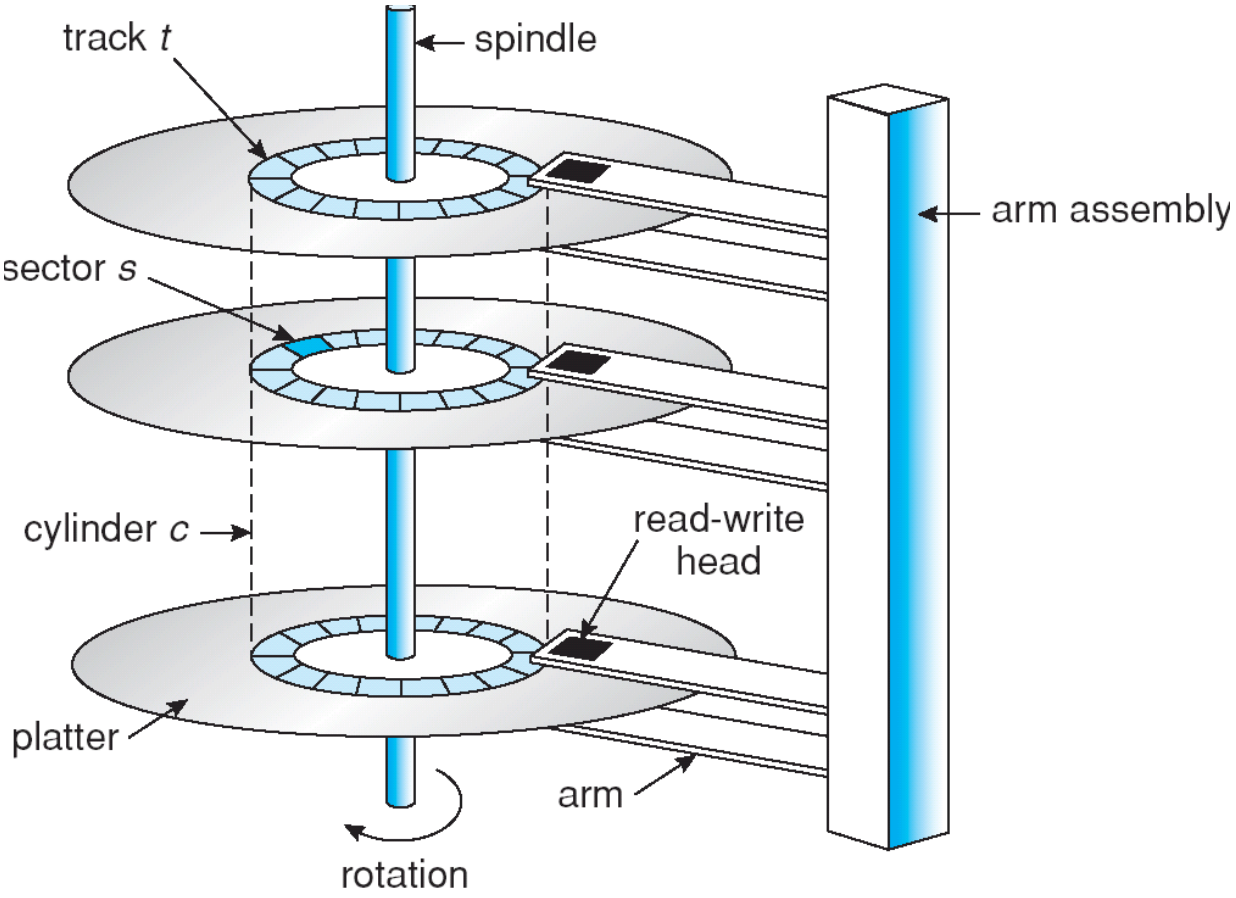
50 Years Old!

- 13th September 1956
- The IBM RAMAC 350
- Stored less than 5 MByte

Reading from a Disk

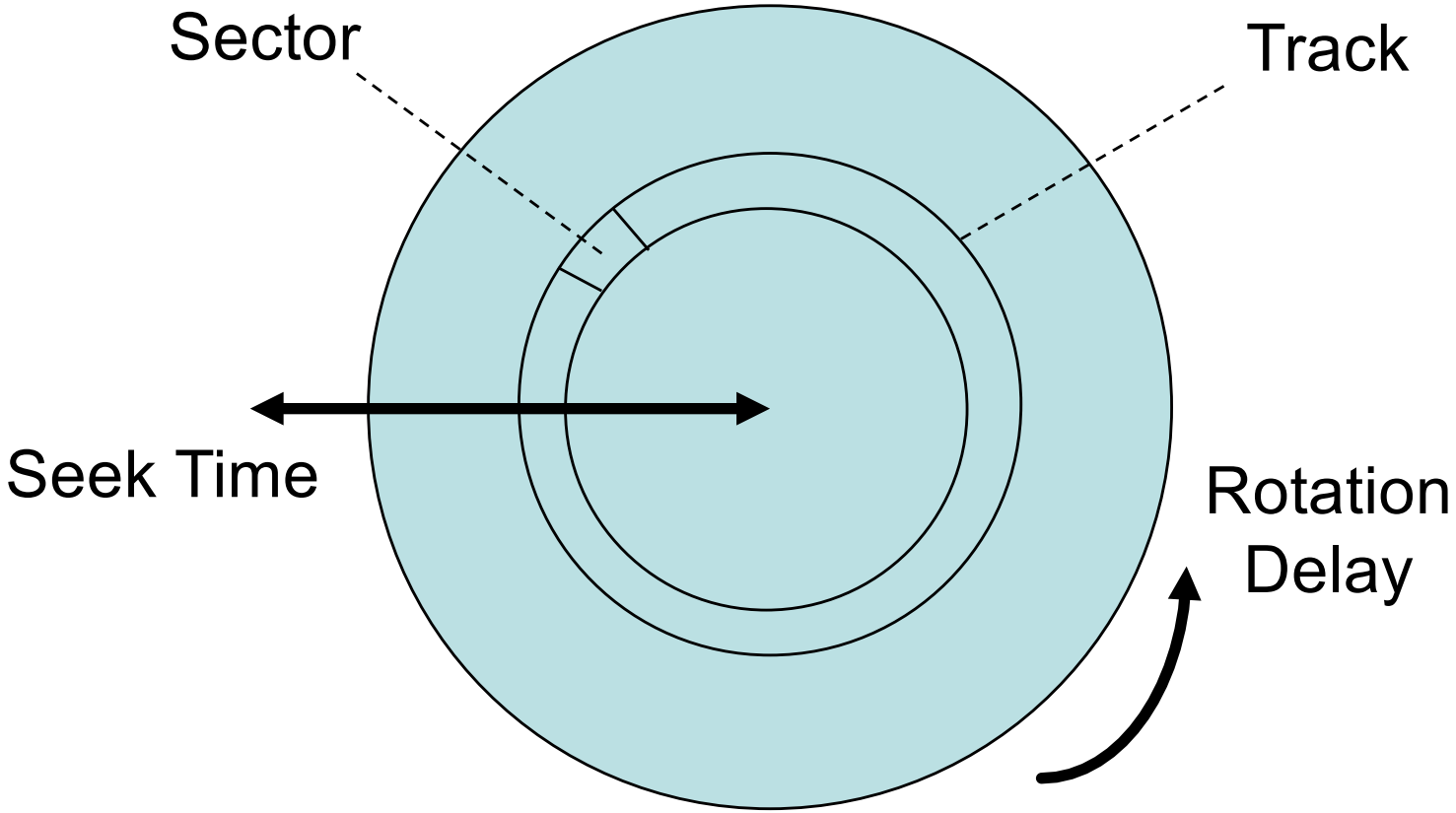
Must specify:

- cylinder #
  (distance from spindle)
- surface #
- sector #
- transfer size
- memory address



Disk overheads

- **Seek time:** to get to the track (5-15 millisecs)
- **Rotational Latency time:** to get to the sector (4-8 millisecs)
- **Transfer time:** get bits off the disk (25-50 microsecs)



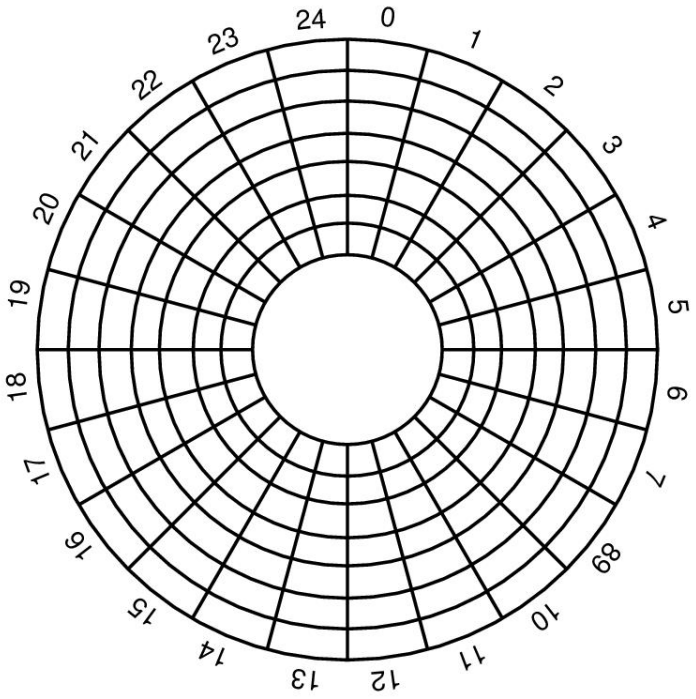
Hard Disks vs. RAM

	Hard Disks	RAM
Smallest write	sector	word
Atomic write	sector	word
Random access	5 ms	10-1000 ns
Sequential access	200 MB/s	200-1000MB/s
Cost	$50 / terabyte	$5 / gigabyte
Power reliance **(survives power outage?)**	Non-volatile (yes)	Volatile (no)

Number of sectors per track?

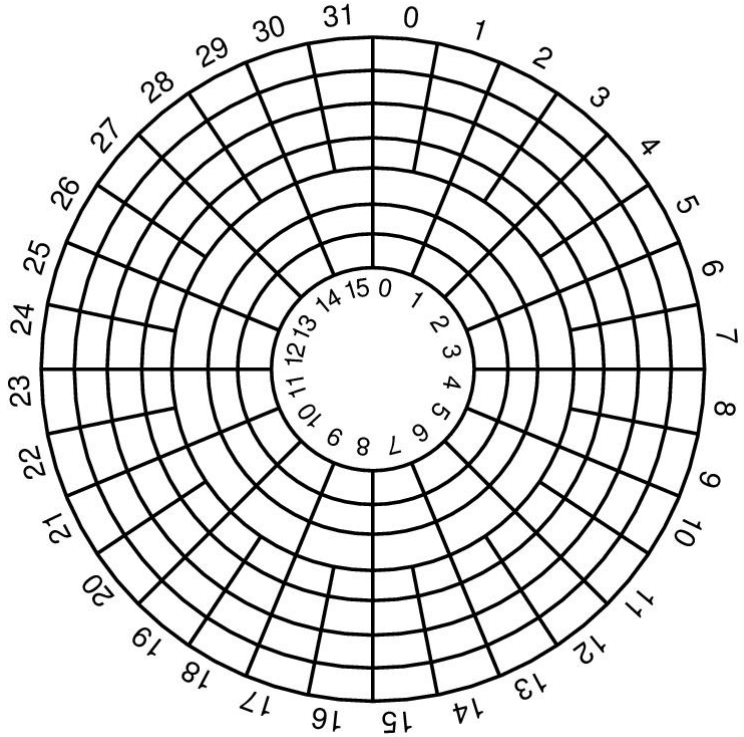
Reduce bit density per track for outer layers

- Constant Linear Velocity
- Typically HDDs

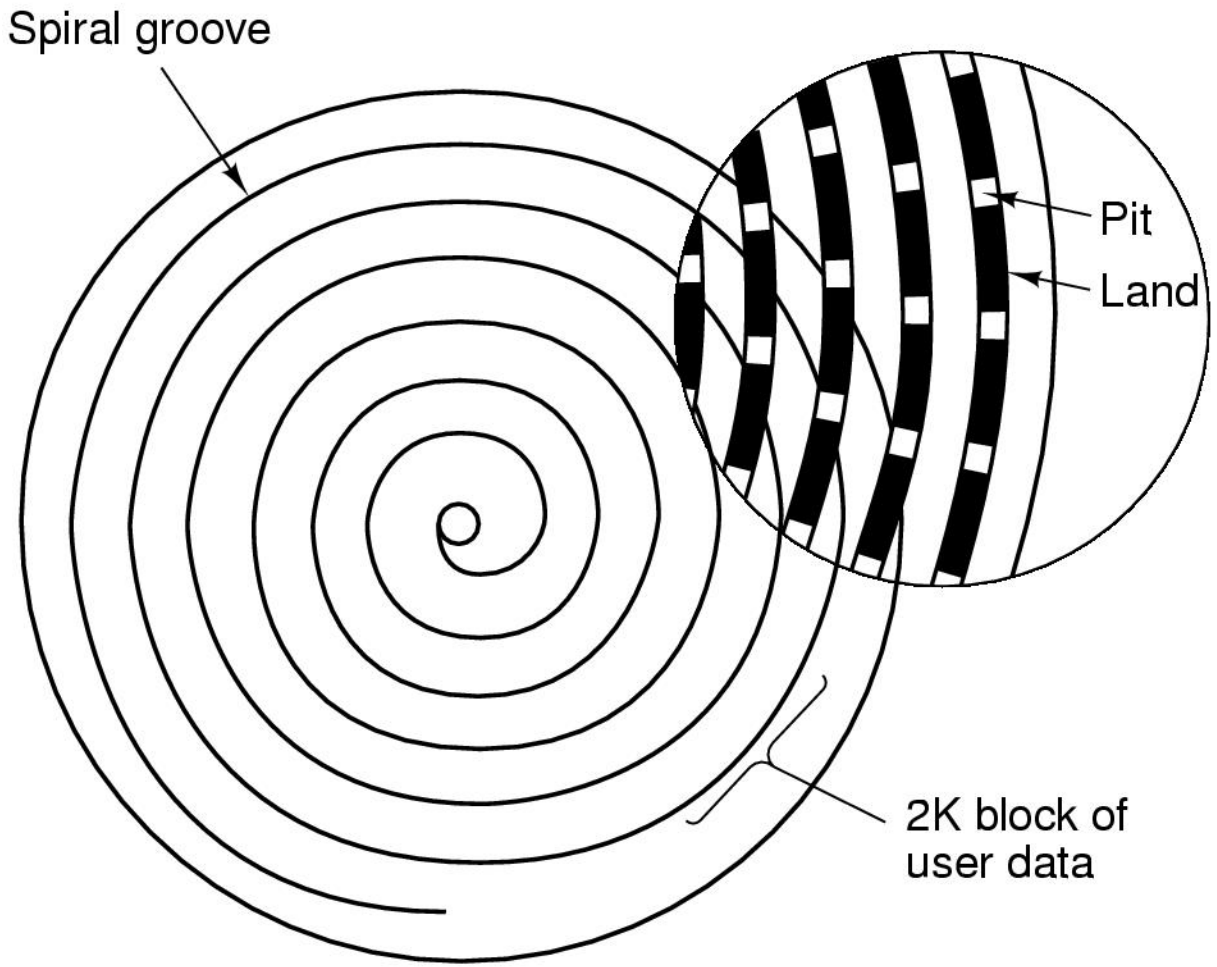


More sectors/track on outer layers

- Increase rotational speed when reading from outer tracks
- Constant Angular Velcity
- Typically CDs, DVDs

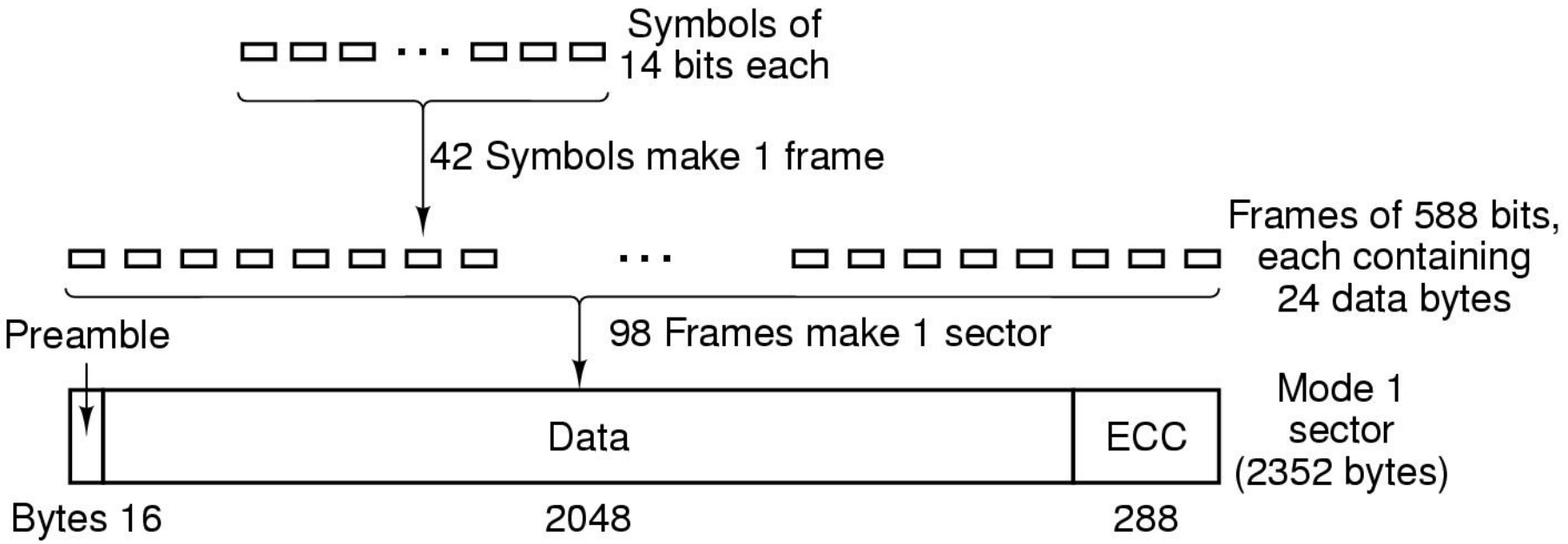


CD-ROM



Spiral makes 22,188 revolutions around disk (~600/mm).
Will be 5.6 km long. Rotation rate: 530 rpm to 200 rpm

CD-ROM: Logical Layout



Disk Scheduling

Objective: minimize seek time

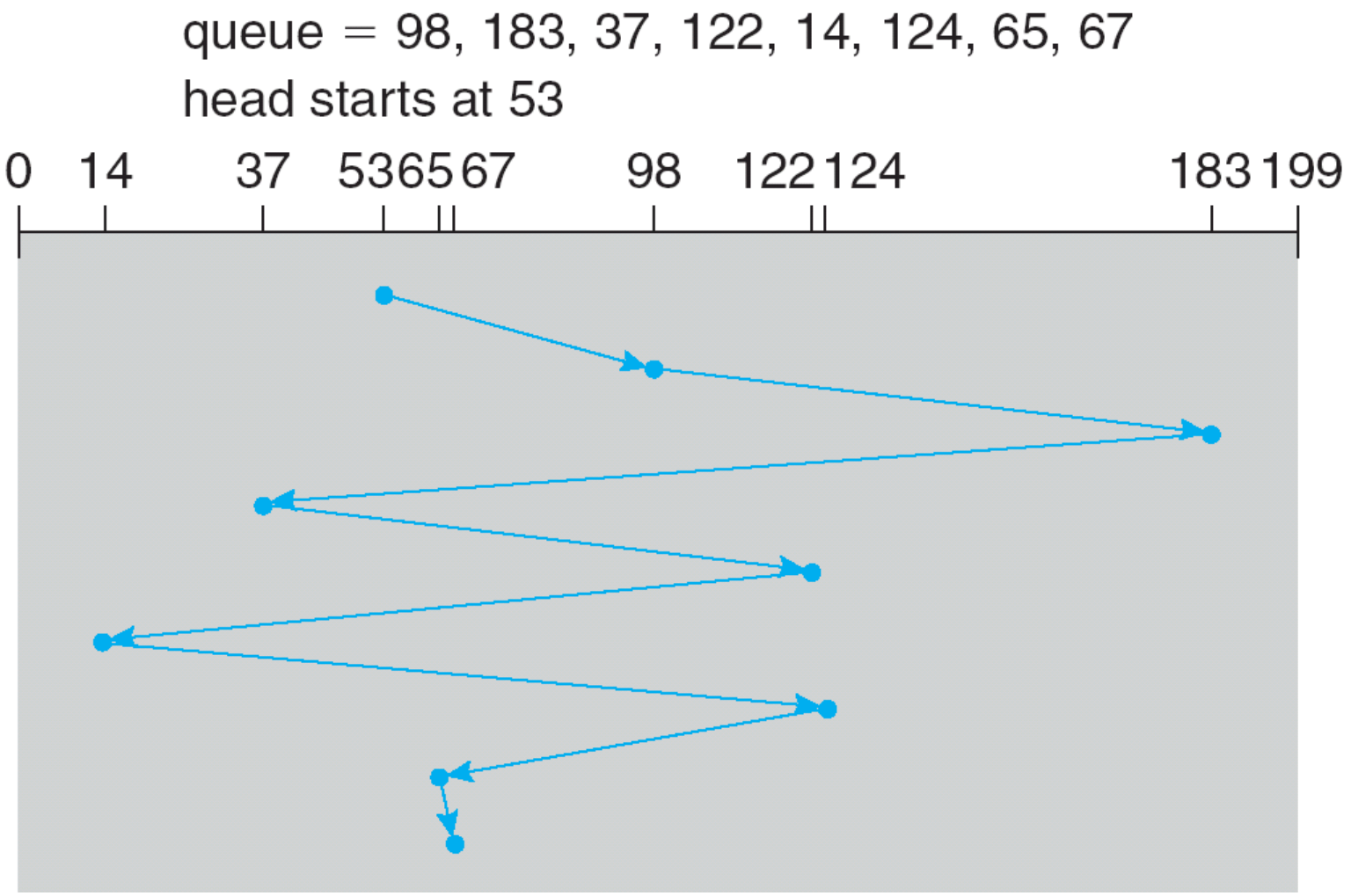
Illustrate with a request queue (0-199)

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

Head pointer 53

Metric: how many cylinders moved?

FCFS: first come first served

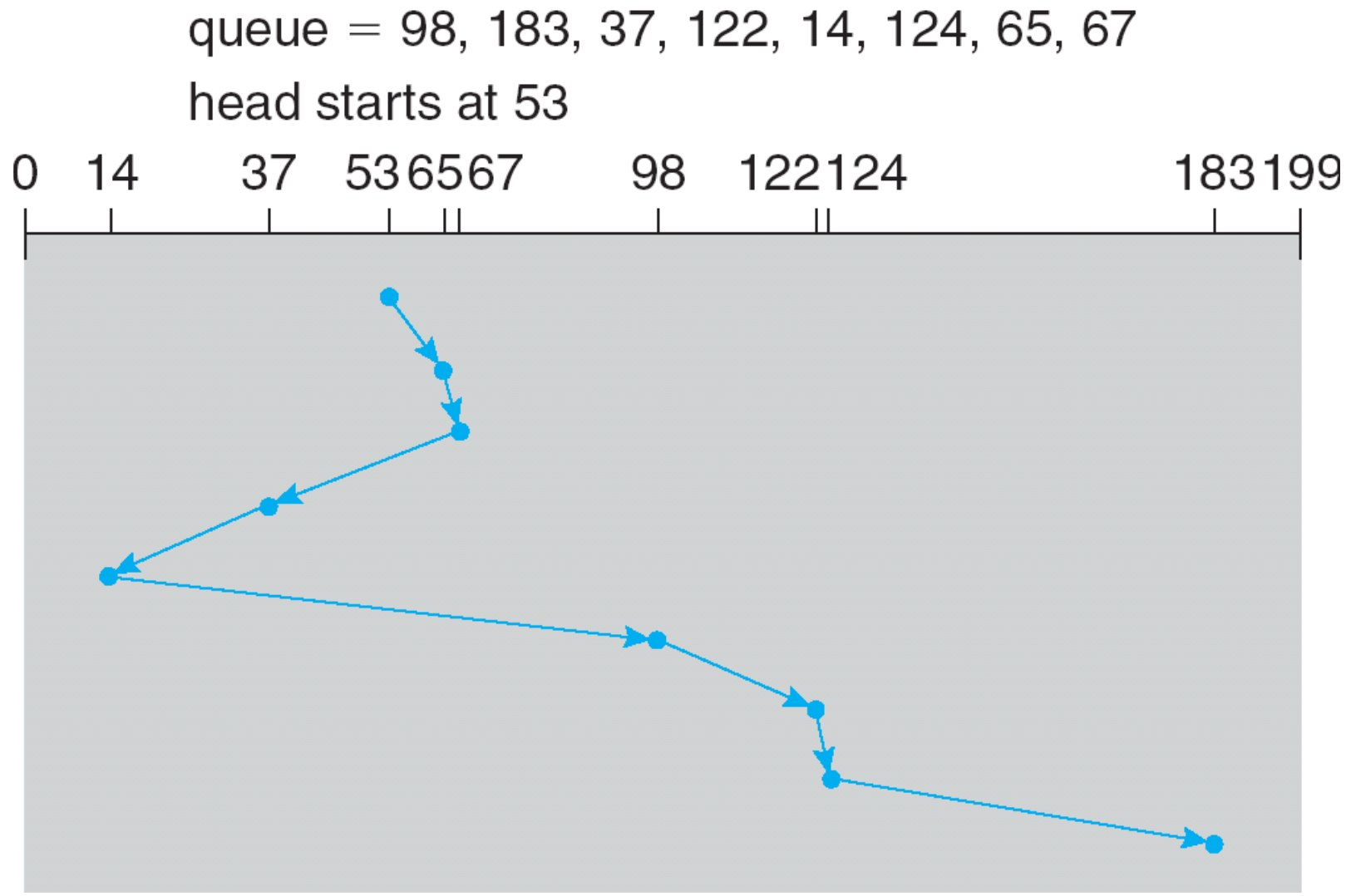


Queue is list of cylinder numbers. Total head movement of 640 cylinders.

SSTF: shortest seek time first

- Select request with minimum seek time from current head position
- A form of Shortest Job First (SJF) scheduling
  - may cause starvation of some requests

SSTF Illustrated



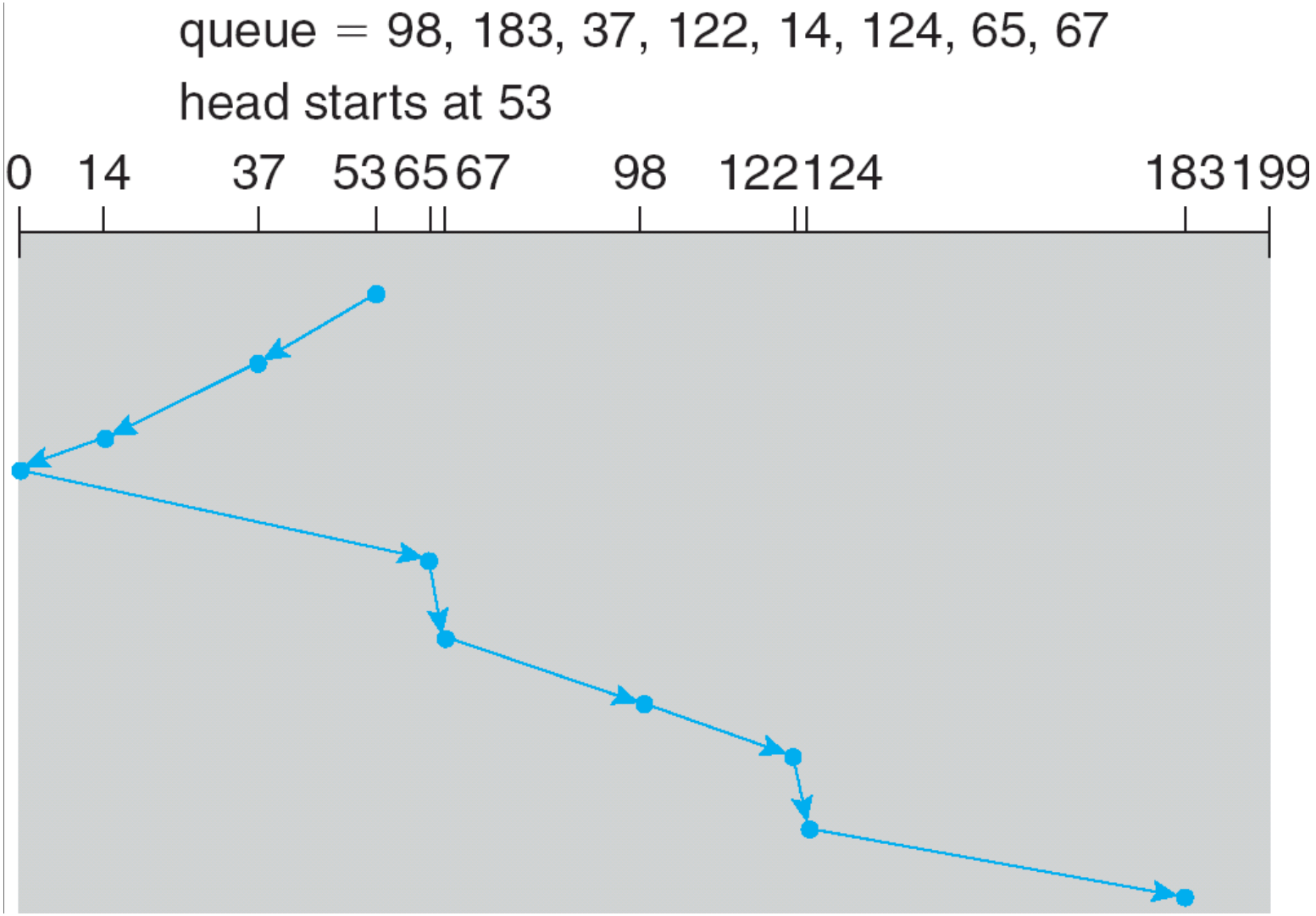
Total head movement of 236 cylinders.

SCAN

- The disk arm starts at one end of the disk
  - moves toward the other end, servicing requests
  - head movement is reversed when it gets to the other end of disk
  - servicing continues

- Sometimes called the *elevator algorithm*

SCAN Illustrated



Total head movement of 208 cylinders.

C-SCAN

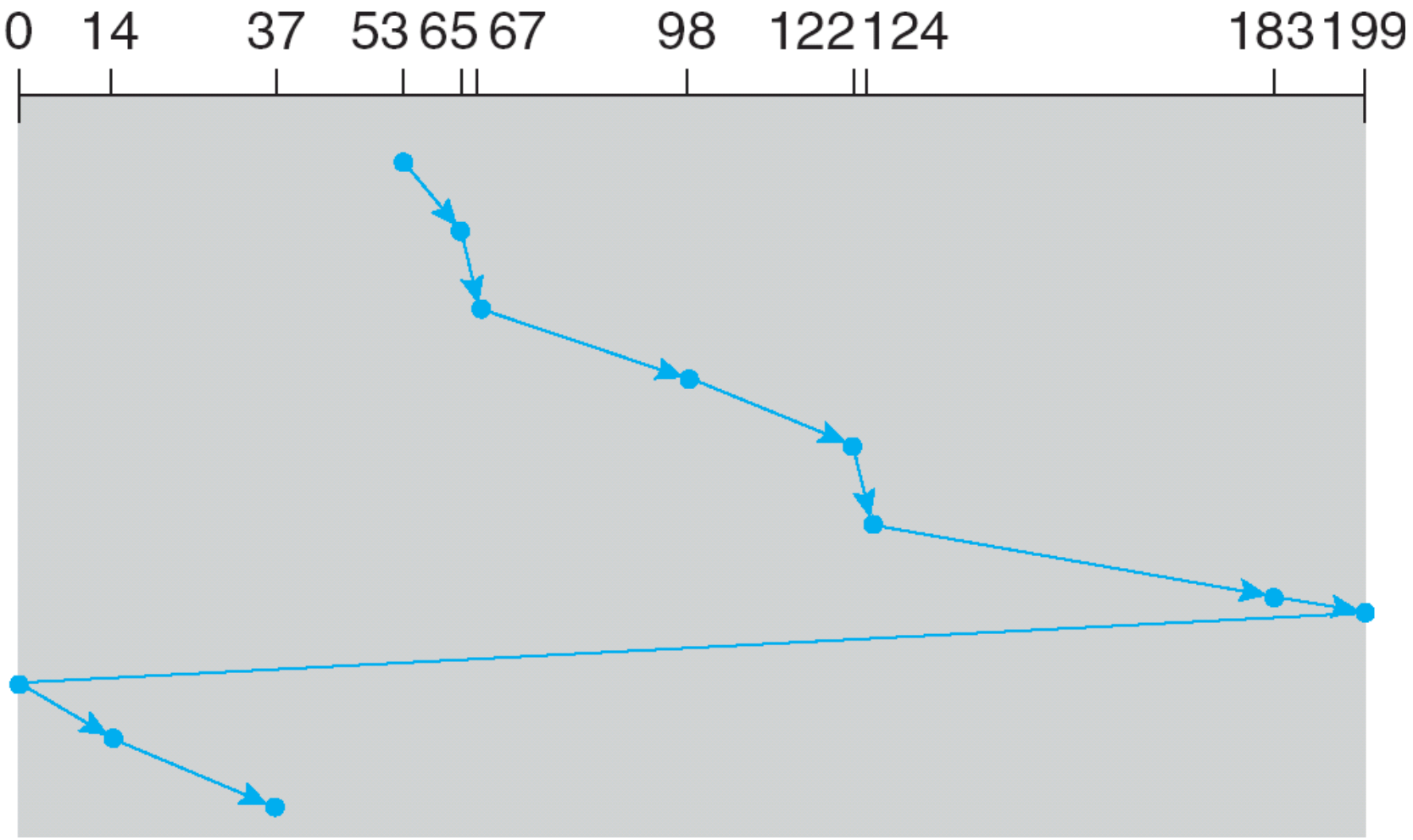
- More uniform wait time than SCAN
- Head moves from one end of disk to other
  - servicing requests as it goes
  - when it reaches the other end, immediately returns to beginning of the disk
  - No requests serviced on return trip

- Treats the cylinders as a circular list
  - wraps around from the last cylinder to first

C-SCAN Illustrated

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

head starts at 53



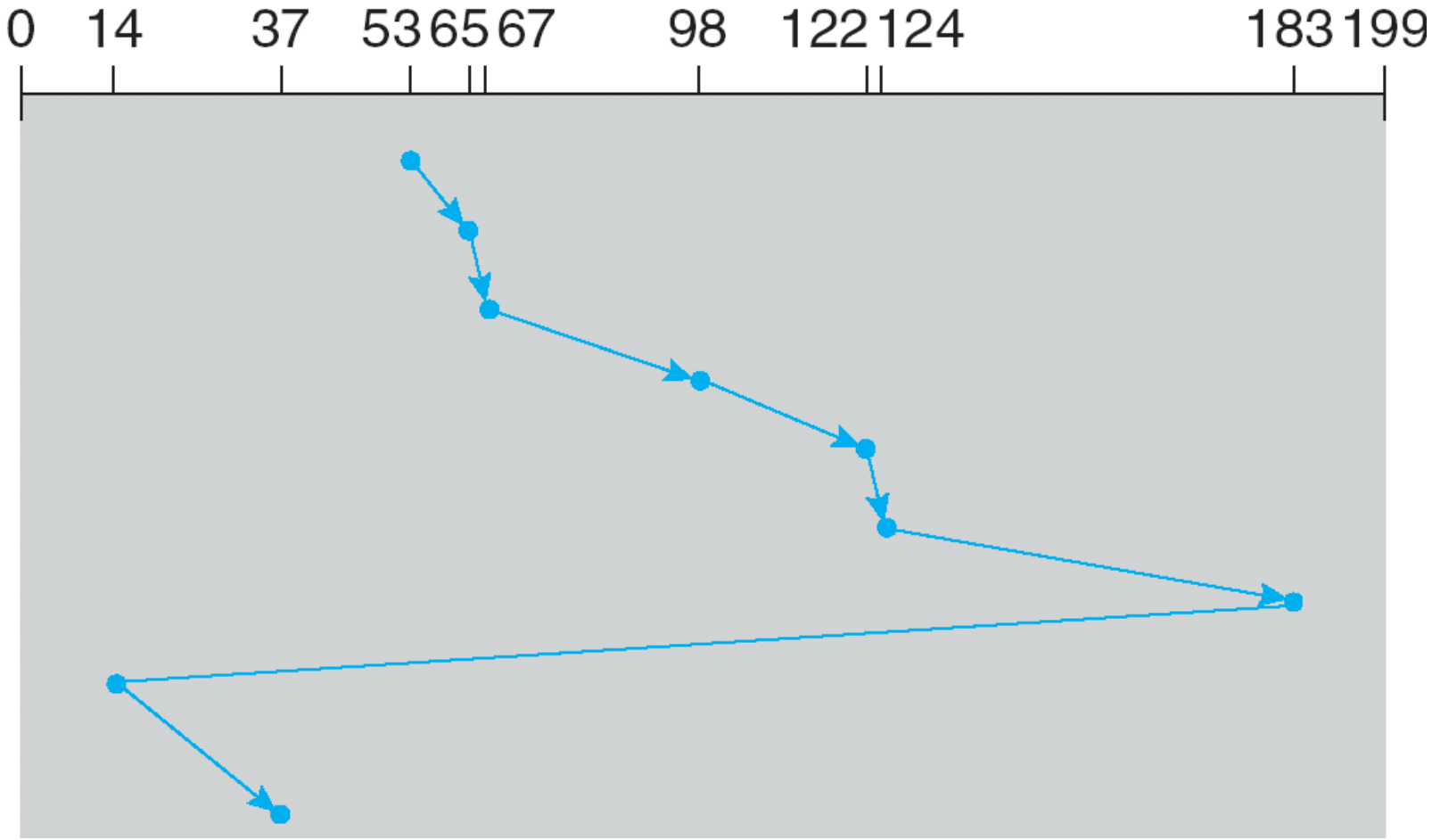
C-LOOK

- Version of C-SCAN
- Arm only goes as far as last request in each direction,
  - then reverses direction immediately,
  - without first going all the way to the end of the disk.

C-LOOK (Cont.)

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

head starts at 53



Solid State Drives (Flash)

- Most SSDs based on NAND-flash
  - other options include DRAM with battery or NOR gates

NAND Flash

- Structured as a set of blocks, each consisting of a set of pages
- Typical page size is .5 – 4 Kbytes
- Typical block size is 16-512 Kbytes

NAND-flash Limitations

- can't overwrite a single byte or word. Instead, have to erase entire blocks
- number of erase cycles per block is limited (memory wear)
  - *wear leveling*: trying to distribute erasures across the entire driver
- reads can "disturb" nearby words and overwrite them with garbage

SSD vs HDD

	SSD	HDD
Cost	10cts/gig	6cts/gig
Power	2-3W	6-7W
Typical Capacity	1TB	2TB
Write Speed	250MB/sec	200MB/sec
Read Speed	700MB/sec	200MB/sec

RAID Motivation

- Disks are improving, but not as fast as CPUs
  - 1970s seek time: 50-100 ms.
  - 2000s seek time: <5 ms.
  - Factor of 20 improvement in 3 decades
- We can use multiple disks for improving performance
  - By striping files across multiple disks (placing parts of each file on a different disk), parallel I/O can improve access time
- Striping reduces reliability
  - 100 disks have 1/100th mean time between failures of one disk
- So, we need striping for performance, but we need something to help with reliability / availability
- To improve reliability, we can add redundancy

RAID

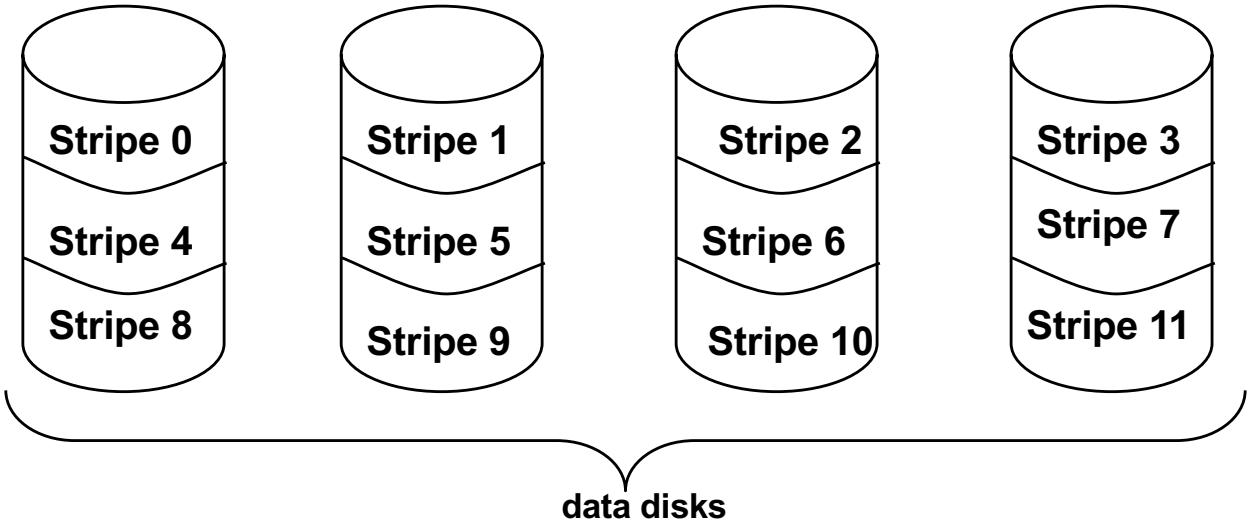
- A RAID is a Redundant Array of Inexpensive Disks
  - In industry, “I” is for “Independent”
  - The alternative is SLED, single large expensive disk
- Disks are small and cheap, so it’s easy to put lots of disks (10s to 100s) in one box for increased storage, performance, and availability
- The RAID box with a RAID controller looks just like a SLED to the computer
- Data plus some redundant information is striped across the disks in some way
- How that striping is done is key to performance and reliability.

Some RAID Issues

- **Granularity**
  - fine-grained: Stripe each file over all disks. This gives high throughput for the file, but limits to transfer of 1 file at a time
  - coarse-grained: Stripe each file over only a few disks. This limits throughput for 1 file but allows more parallel file access
- **Redundancy**
  - uniformly distribute redundancy info on disks: avoids load-balancing problems
  - concentrate redundancy info on a small number of disks: partition the set into data disks and redundant disks

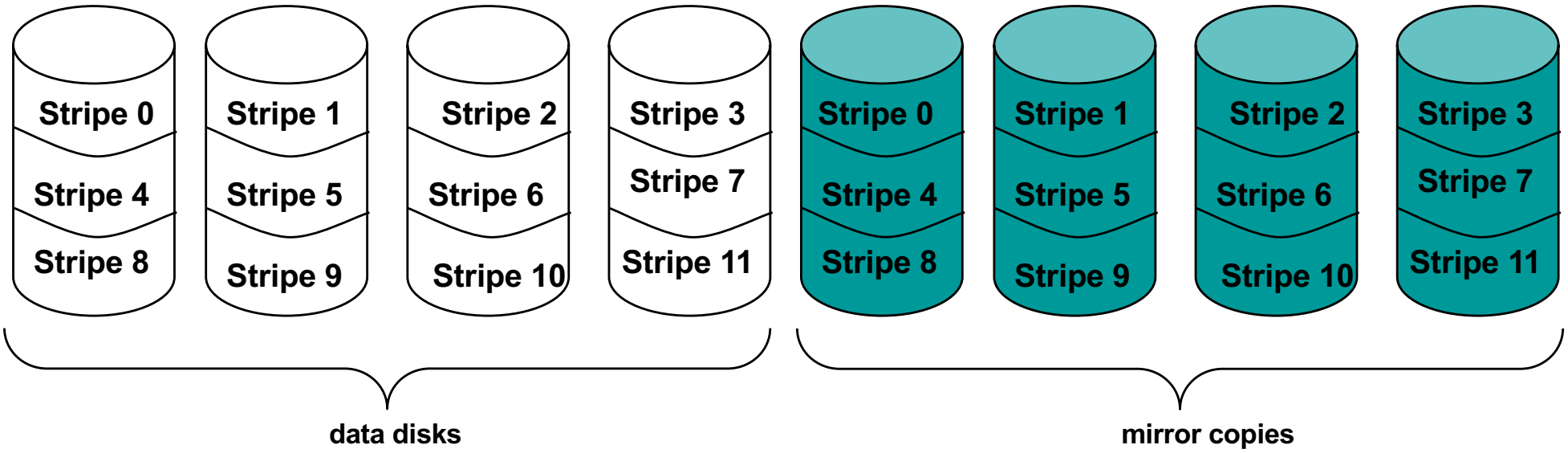
RAID Level 0

- Level 0 is non-redundant disk array
- Files are striped across disks, no redundant info
- High read throughput
- Best write throughput (no redundant info to write)
- Any disk failure results in data loss
  - Reliability worse than SLED, typically



RAID Level 1

- Mirrored Disks --- data is written to two places
  - On failure, just use surviving disk
  - *In theory, can this detect, and if so, correct bit flip errors??*
- Spread read operations across all mirrors
  - Write performance is same as single drive
  - Read performance is 2x better
- Simple but expensive



Detecting a bit flip: Parity Code

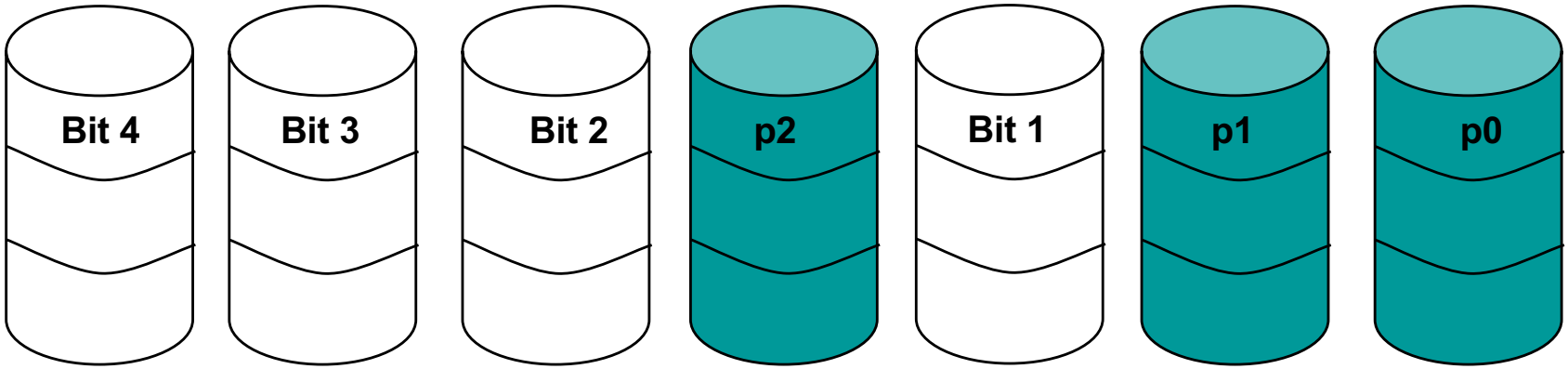
- Suppose you have a binary number, represented as a collection of bits: <b4, b3, b2, b1>, e.g. <1101>
- XOR all the bits
  - parity bit is 0 iff the number of 1 bits is even
- Parity(<b4, b3, b2, b1>) = p = b1 ⊗ b2 ⊗ b3 ⊗ b4
- Parity(<b4, b3, b2, b1, p>) = 0 if all bits are intact
- Parity(<1101>) = 1, Parity(<11011>) = 0
- Parity(<11111>) = 1 => ERROR!
- Parity can detect a single error, but can't tell which bits got flipped
  - May be the parity bit that got flipped --- that's ok
  - Method breaks if an even number of bits get flipped

Hamming Code

- Hamming codes can detect double bit errors and detect & correct single bit errors
- Insert parity bits at bit positions that are powers of 2 (1, 2, 4, …)
  - <b4, b3, b2, b1> ➔ <b4, b3, b2, p3, b1, p1, p0>
- 7/4 Hamming Code
  - p0 = b1 ⊗ b2 ⊗ b4 // all positions that are of the form xxx1
  - p1 = b1 ⊗ b3 ⊗ b4 // all positions that are of the form xx1x
  - p2 = b2 ⊗ b3 ⊗ b4 // all positions that are of the form x1xx
- For example:
  - p0(<1101>) = 0, p1(<1101>) = 1, p2(<1101>) = 0
  - Hamming(<1101>) = <b4, b3, b2, p2, b1, p1, p0> = <1100110>
  - If a bit is flipped, e.g. <1110110>
    - Hamming(<1111>) = <1111111>
    - p0 and p2 are wrong. Error occurred in bit 0b101 = 5.

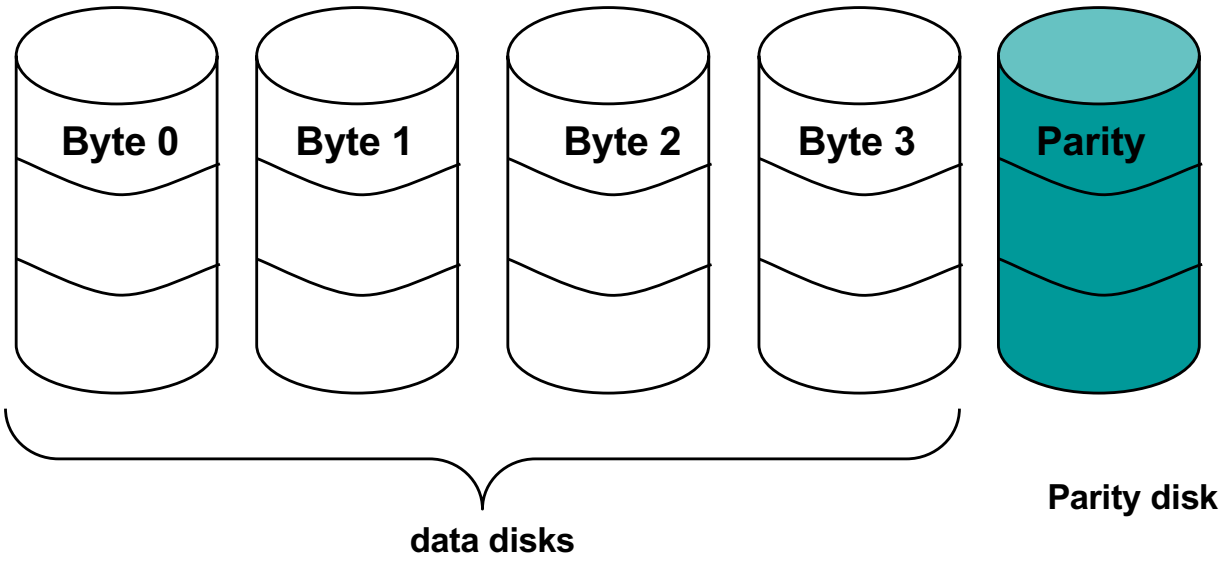
RAID Level 2

- Bit-level striping with Hamming (ECC) codes for error correction
- All 7 disk arms are synchronized and move in unison
- Complicated controller (and hence very unpopular)
- Single access at a time
- Tolerates only one error, but with no performance degradation



RAID Level 3

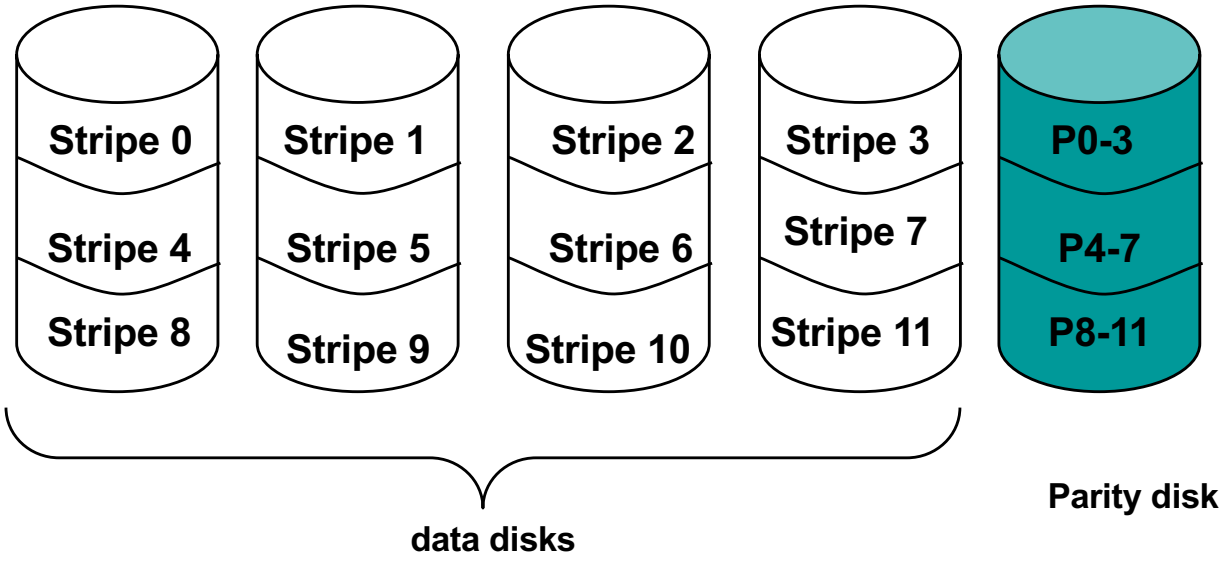
- Byte-level striping
- Use a parity disk
  - Not usually to detect failures, but to compute missing data in case disk fails
- A read accesses all data disks
  - On disk failure, read parity disk to compute the missing data
- A write accesses all data disks plus the parity disk
- Also rarely used



Single parity disk can be used to fill in missing data if one disk fails

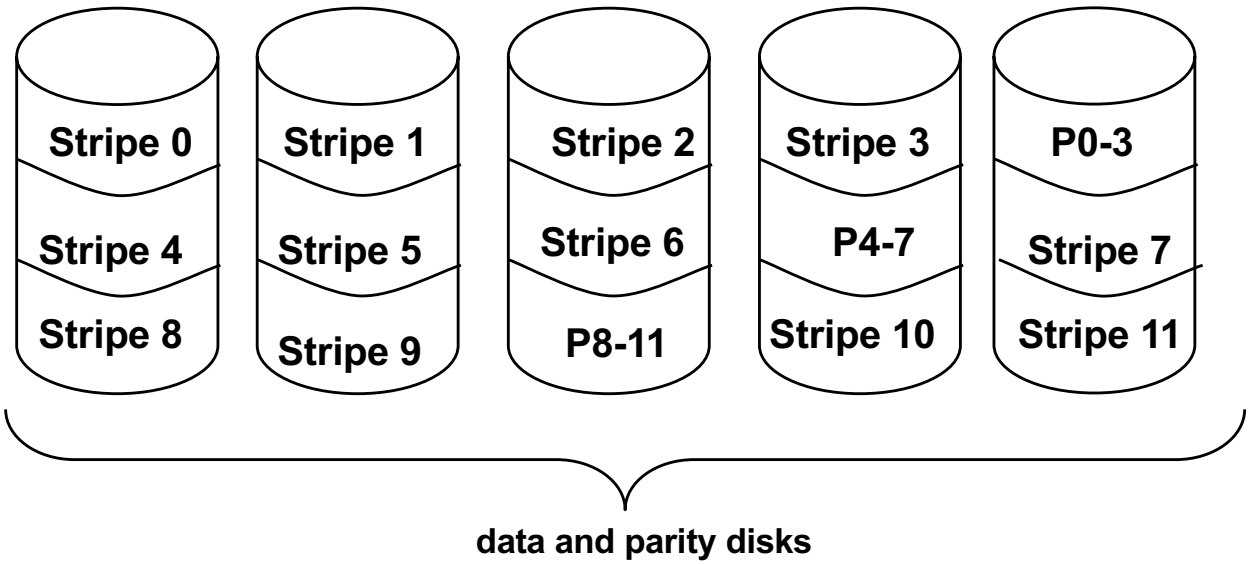
RAID Level 4

- Combines Level 0 and 3 – block-level parity with stripes
- A read accesses just the relevant data disk
- A write accesses all data disks plus the parity disk
  - Optimization: can read/write just the data disk and the parity disk, at the expense of a longer latency. Can you see how?
- Parity disk is a bottleneck for writing
- Also rarely used



RAID Level 5

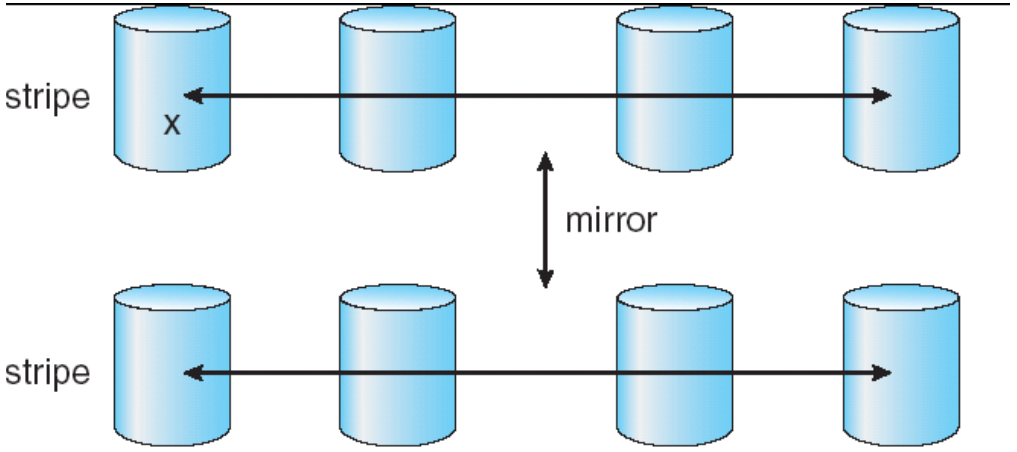
- Block Interleaved Distributed Parity
- Like parity scheme, but distribute the parity info over all disks (as well as data over all disks)
- Better read performance, large write performance
  - Reads can outperform SLEDs and RAID-0



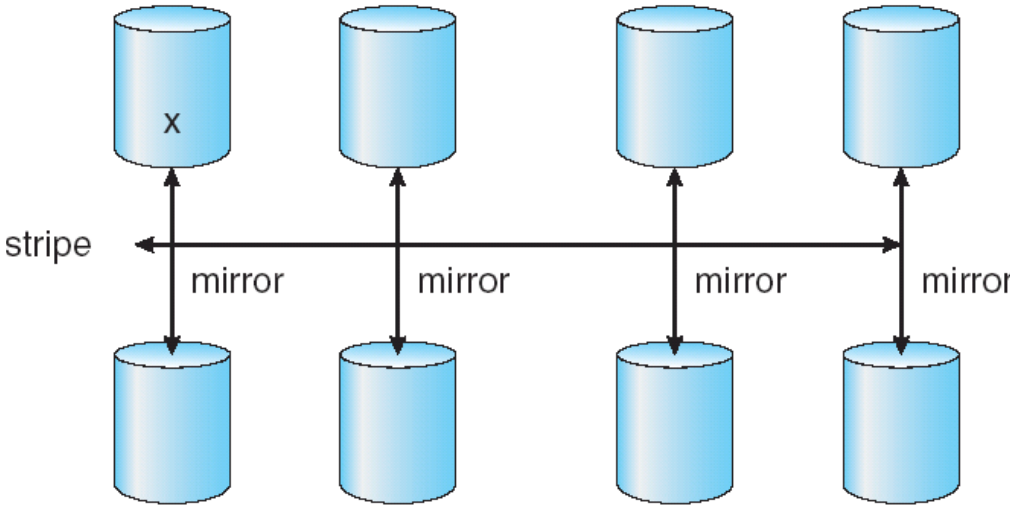
RAID Level 6

- Level 5 with an extra parity bit (*sort of*)
- Can tolerate two disk failures
  - What are the odds of having two concurrent disk failures ?
- May outperform Level-5 on reads, slower on writes

RAID 0+1 and 1+0



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.