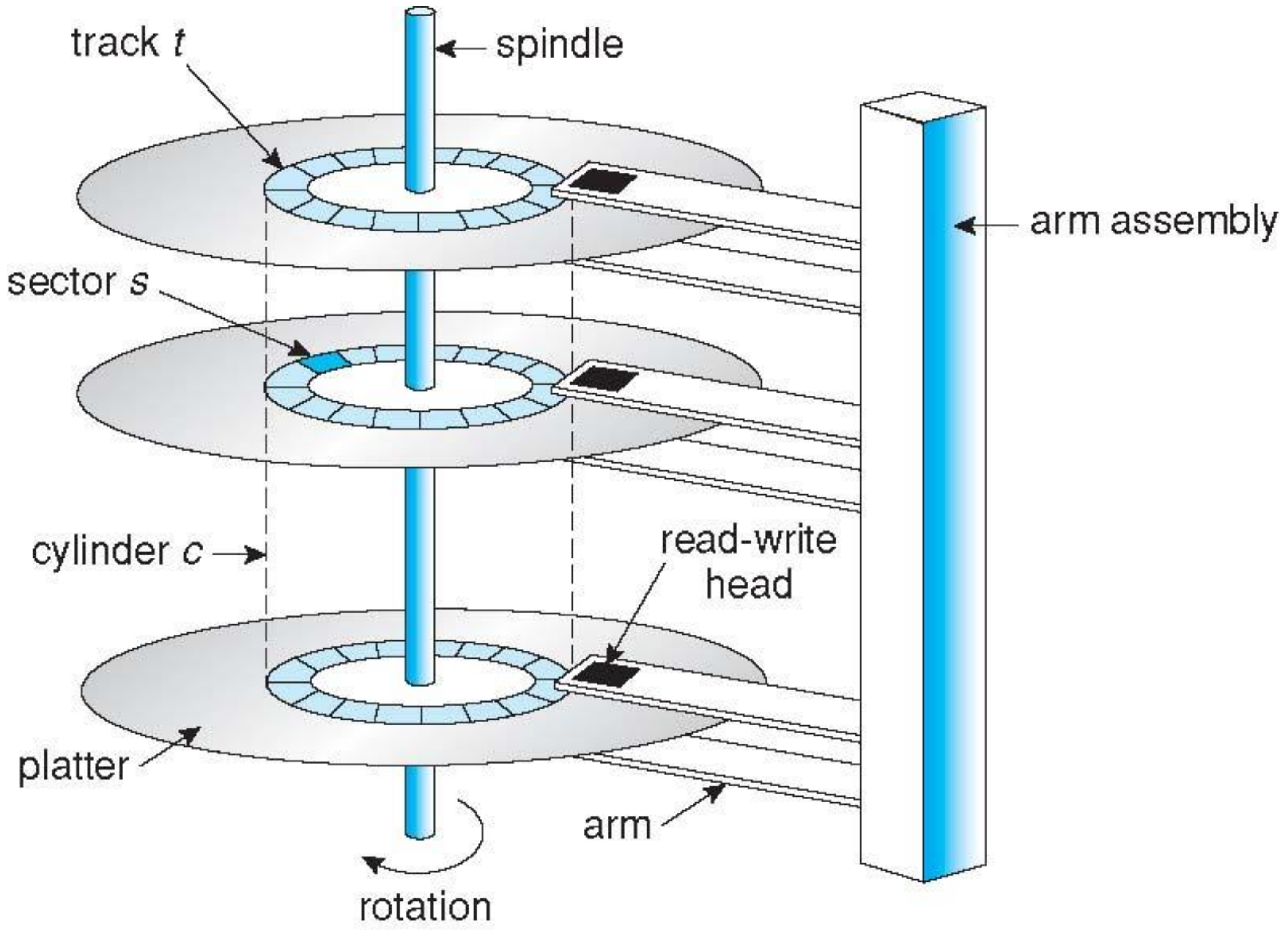
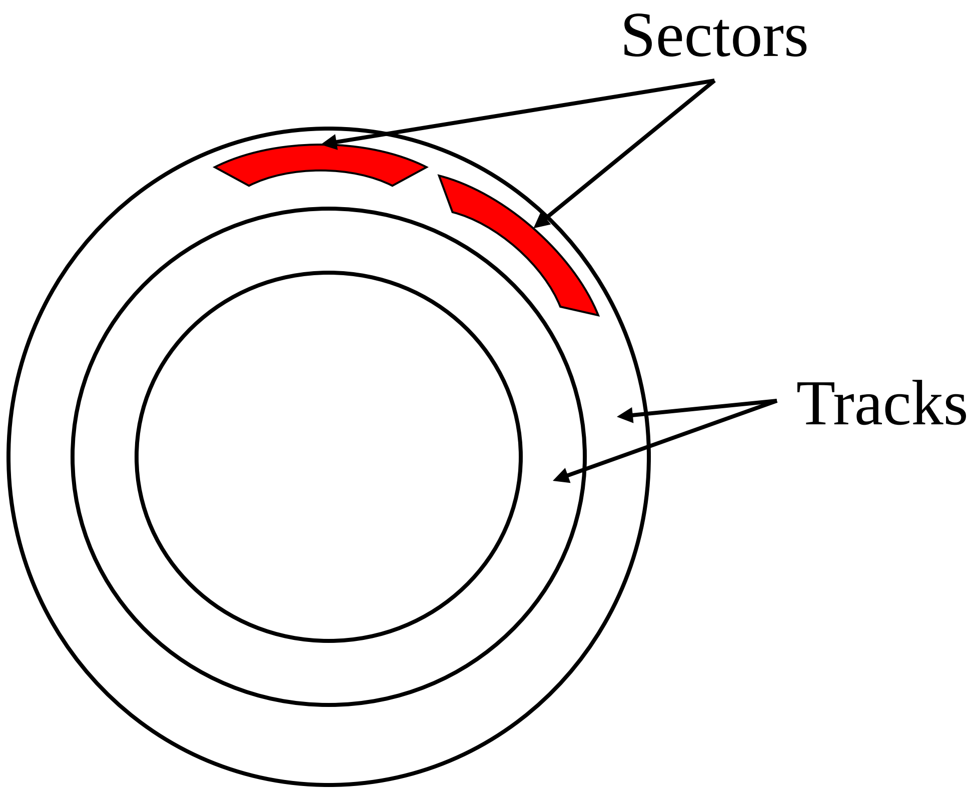
CS416 – Disks

Moving-head Disk Mechanism



Disks



Seek time: time to move the disk head to the desired track

Rotational delay: time to reach desired sector once head is over the desired track

Transfer rate: rate data read/write to disk

Some typical parameters:

- Seek: ~4ms
- Rotational delay: ~4.15ms for 7200 rpm
- Transfer rate: ~ns

Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Access time has two major components
  - **Seek time** is the time for the disk are 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 to the disk head
- Minimize seek time
- Seek time ≈ seek distance
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Operation of a Hard-Drive

Disk Scheduling (Cont.)

- Several algorithms exist to schedule the servicing of disk I/O requests

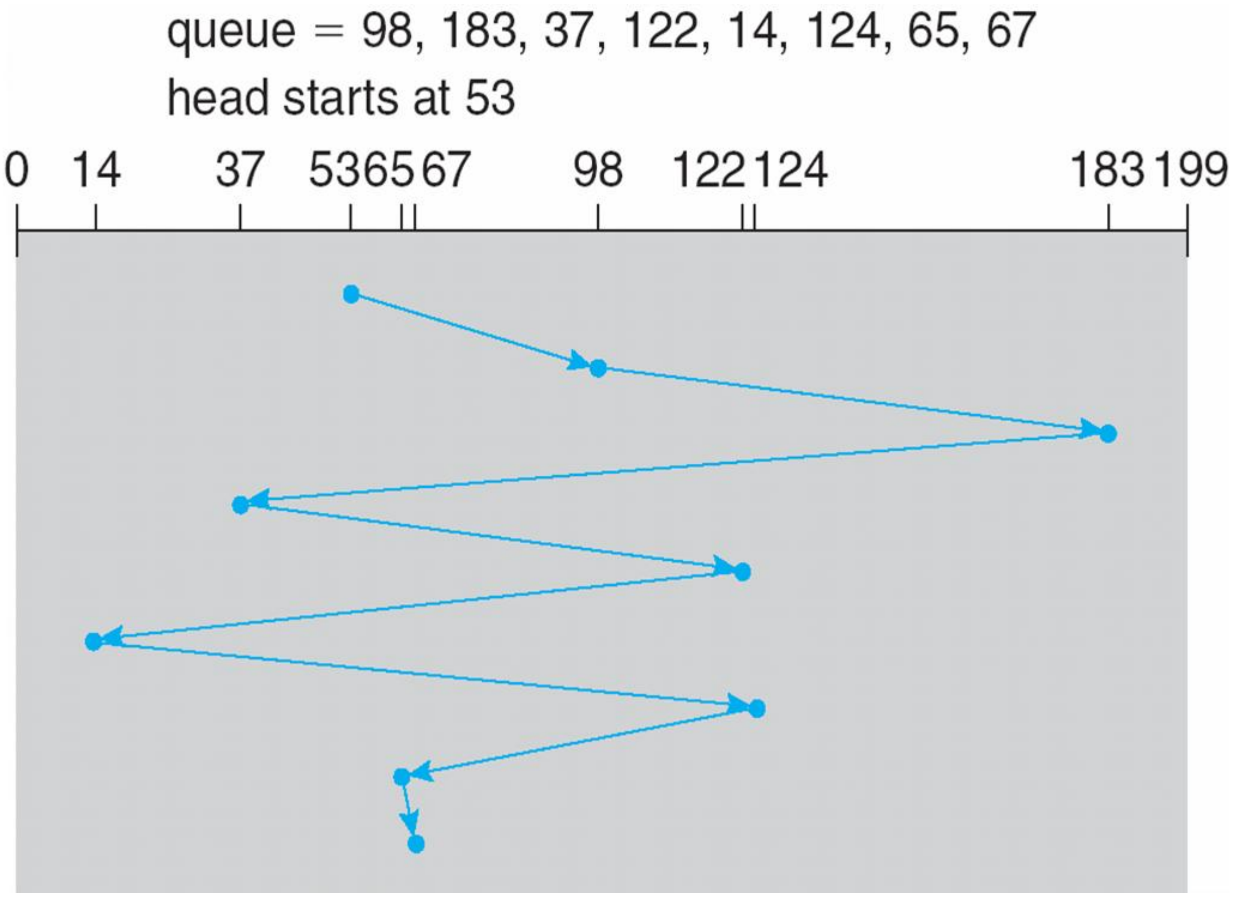
- We illustrate them with a request queue (0-199)

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

  Head pointer 53

FCFS

Illustration shows total head movement of 640 cylinders



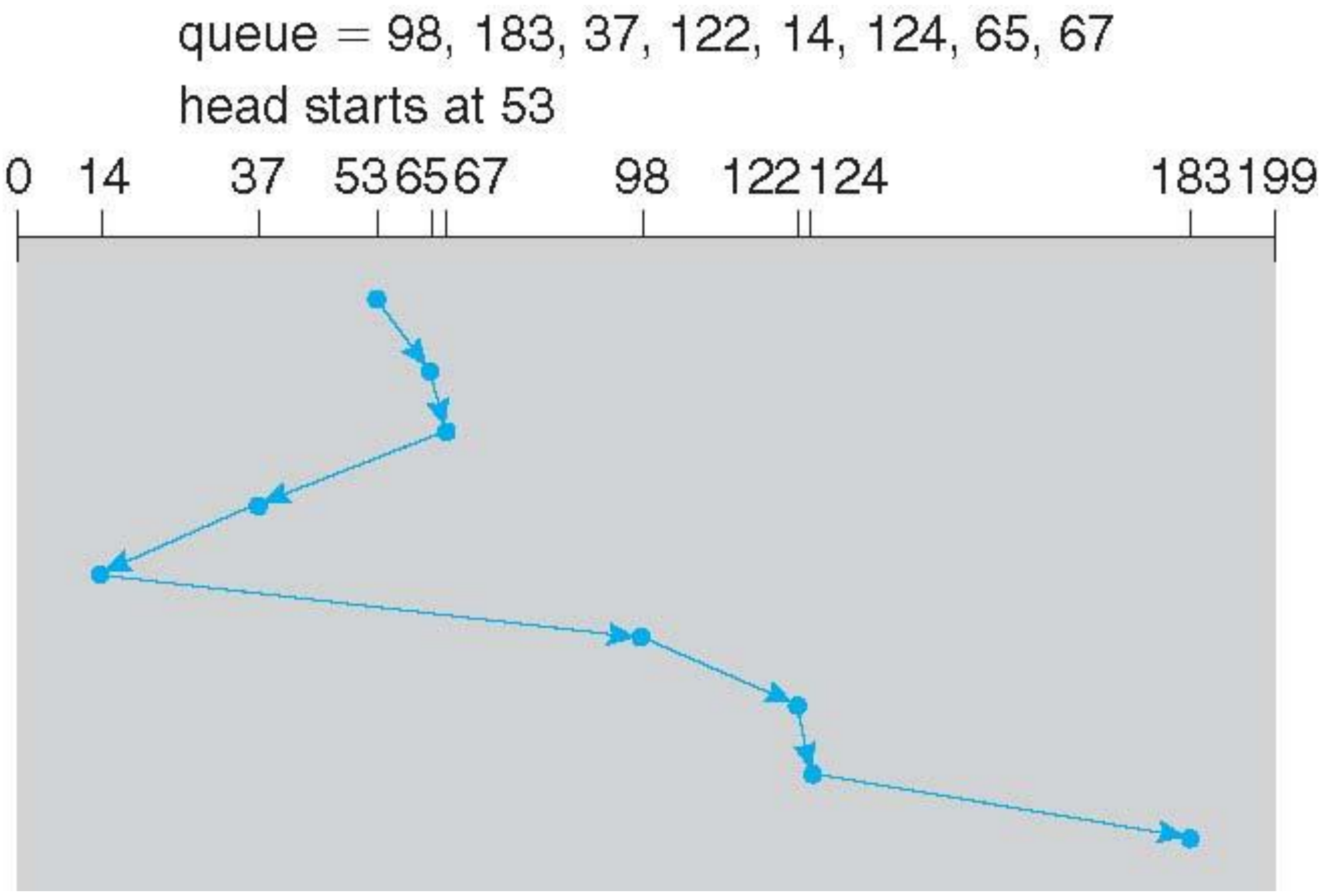
SSTF

- Selects the request with the minimum seek time from the current head position

- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests

- Illustration shows total head movement of 236 cylinders

SSTF (Cont.)



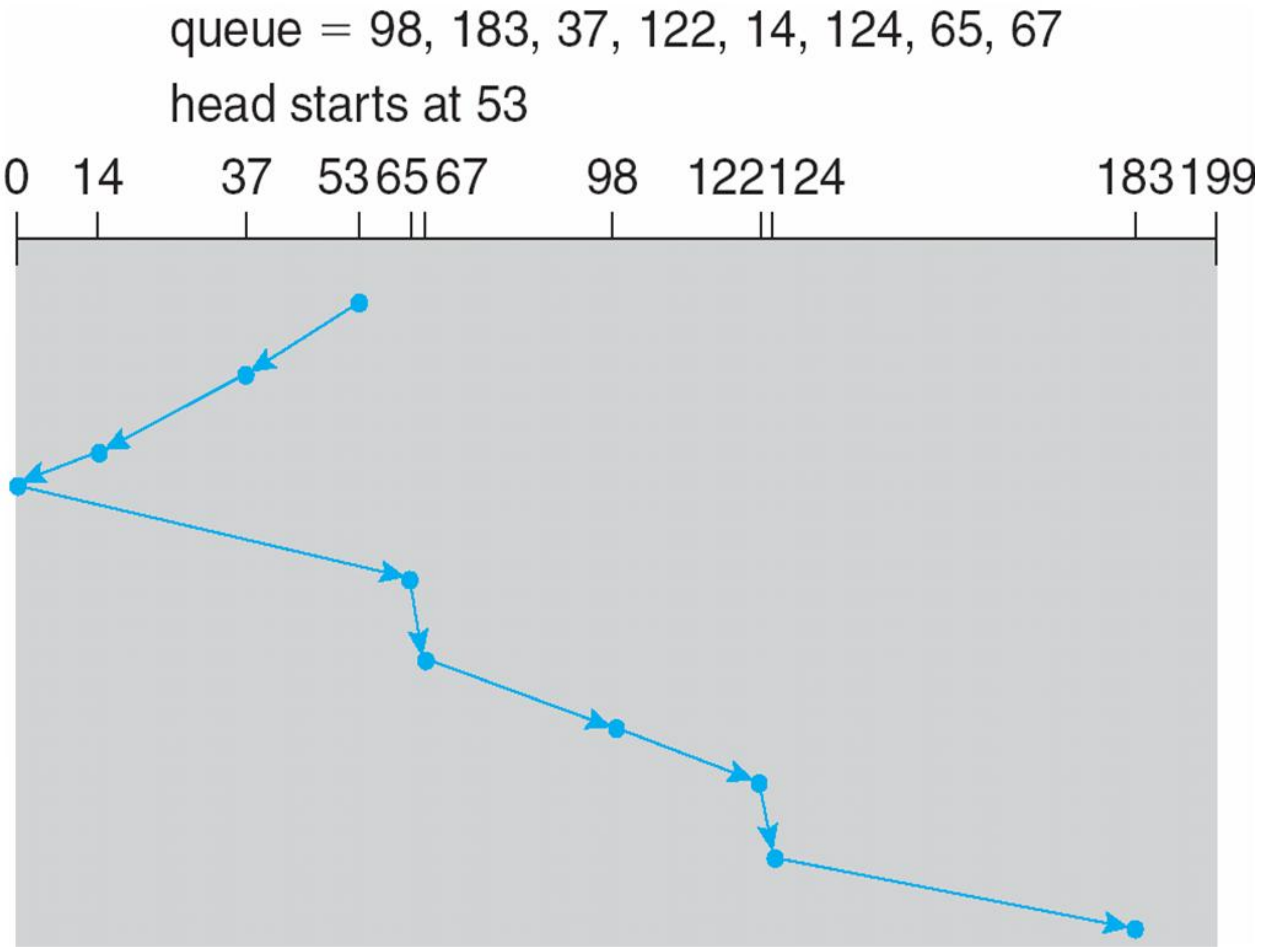
What is the Problem here ?

Favors middle tracks : Head Rarely Moves to edges

SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- Illustration shows total head movement of 208 cylinders

SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

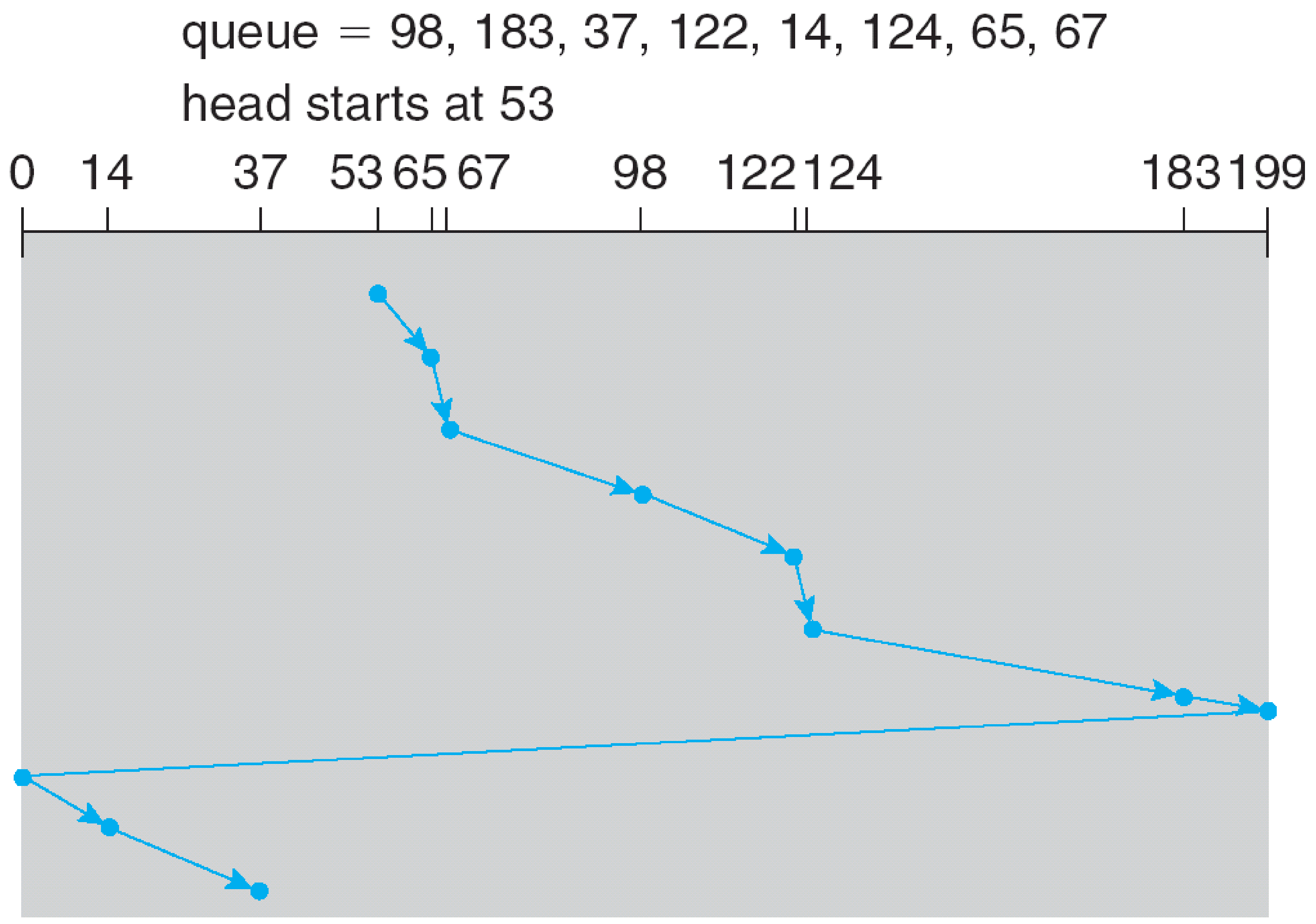
C-SCAN

- Provides a more uniform wait time than SCAN

- The head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip

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

C-SCAN (Cont.)

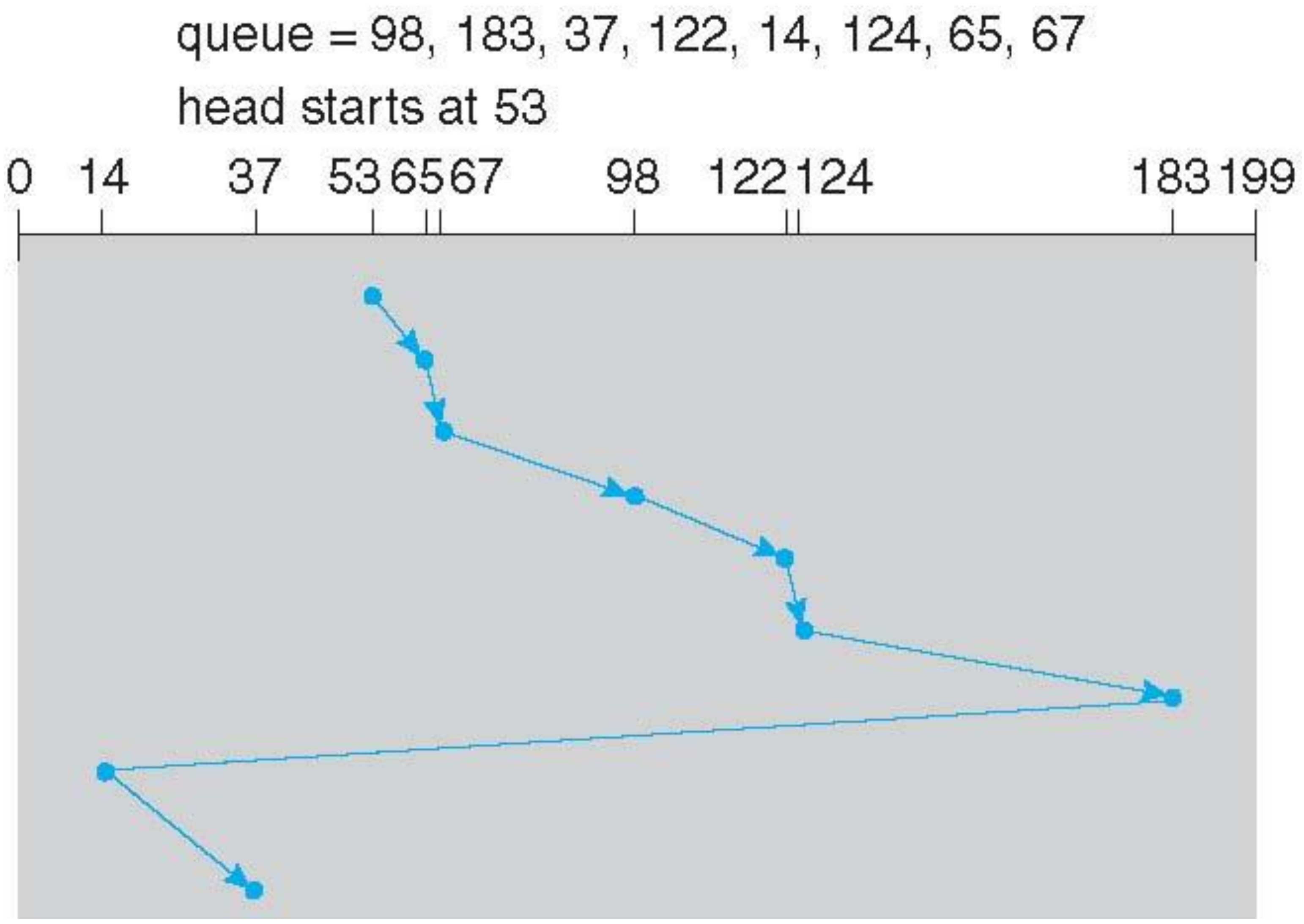


- Idea: Reduce variance in seek times, avoid discriminating against the highest and lowest tracks

C-LOOK

- Version of C-SCAN
- Arm only goes as far as the 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
0
14
37
53
65
67
98
122
124
183
199

RAID - Motivation

Speed of disks not matching that of other components

- Moore's Law: CPU speed doubles every 18 months
- SRAM speeds increasing by 40-100% a year
- In contrast, disk seek time only improving 7% a year
  - *Although greater density leads to improved transfer times once seek is done*

RAID

Basic idea: Build I/O systems as arrays of cheap disks

- Allow data to be **striped** across multiple disks
- Means you can read/write multiple disks in parallel – greatly improve performance

Problem: disks are extremely unreliable

Mean Time to Failure (MTTF)

- MTTF (disk array) = MTTF (single disk) / # disks
- Adding more disks means that failures happen more frequently..
- An array of 100 disks with an MTTF of 30,000 hours = just under 2 weeks!

Increasing Reliability

Idea: Replicate data across multiple disks

- When a disk fails, lost information can be regenerated from the redundant data

Simplest form: Mirroring (also called “RAID 1”)

- All data is mirrored across two disks

Advantages?

-
-

Disadvantages?

-
-

Increasing Reliability

Idea: Replicate data across multiple disks

- When a disk fails, lost information can be regenerated from the redundant data

Simplest form: Mirroring (also called “RAID 1”)

- All data is mirrored across two disks

Advantages:

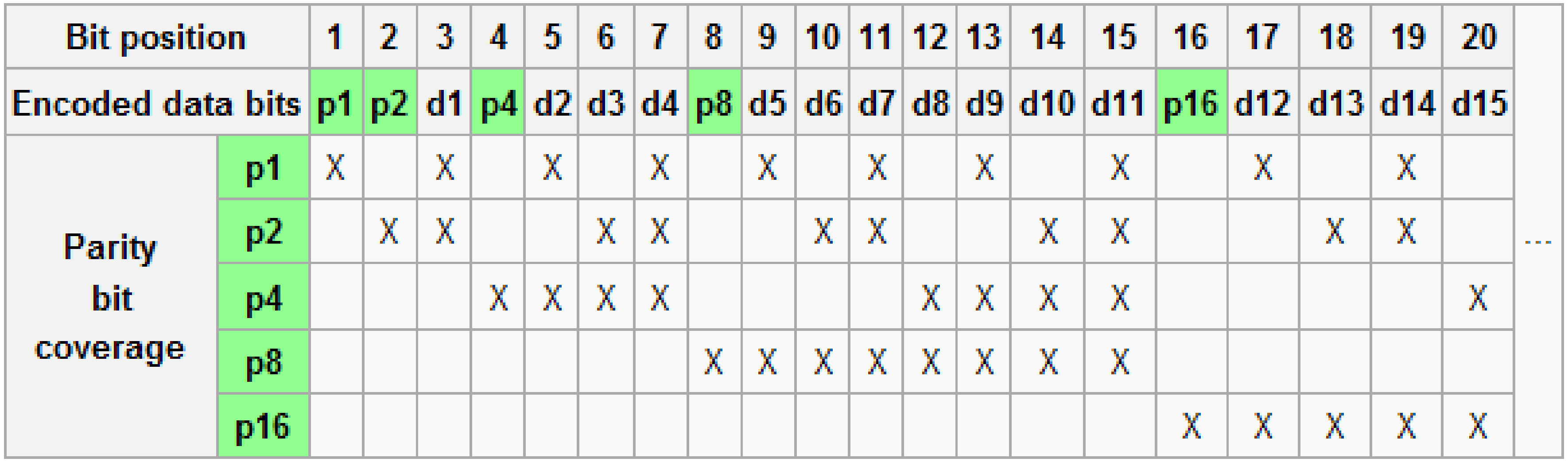
- Reads are faster, since both disks can be read in parallel
- Higher reliability (of course)

Disadvantages:

- Writes are slightly slower, since OS must wait for both disks to do write
- This approach also doubles the cost of the storage system!

RAID-2

- Bit-Level Stripping, Hamming Code Error Detection & Correction



Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	X		X		X		X		X		X		X		X		X		X		
	p2		X	X			X	X			X	X			X	X			X	X		...
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

- P1=d1 XOR d3 XOR d5 XOR….
- P2 = d1 XOR d3 XOR d4 XOR …..
- P4 = d2 XOR d3 XOR d4 XOR d8 XOR …
- Can detect all 1 bit errors just by checking which of the parity bits are wrong.
- Disadvantage: 4 disks require 3 parity disks. Not efficient !

RAID 3

Rather than mirroring, use *parity codes*

- Given N bits $\{b_1, b_2, \dots b_N\}$, the *parity bit* P is the bit {0,1} that yields an even number of "1" bits in the set $\{b_1, b_2, \dots b_N, P\}$
- Idea: If any bit in $\{b_1, b_2, \dots b_N\}$ is lost, can use the remaining bits (plus P) to recover it.

Where to store the parity codes?

- Add an extra "check disk" that stores parity bits for the data stored on the rest of the N disks

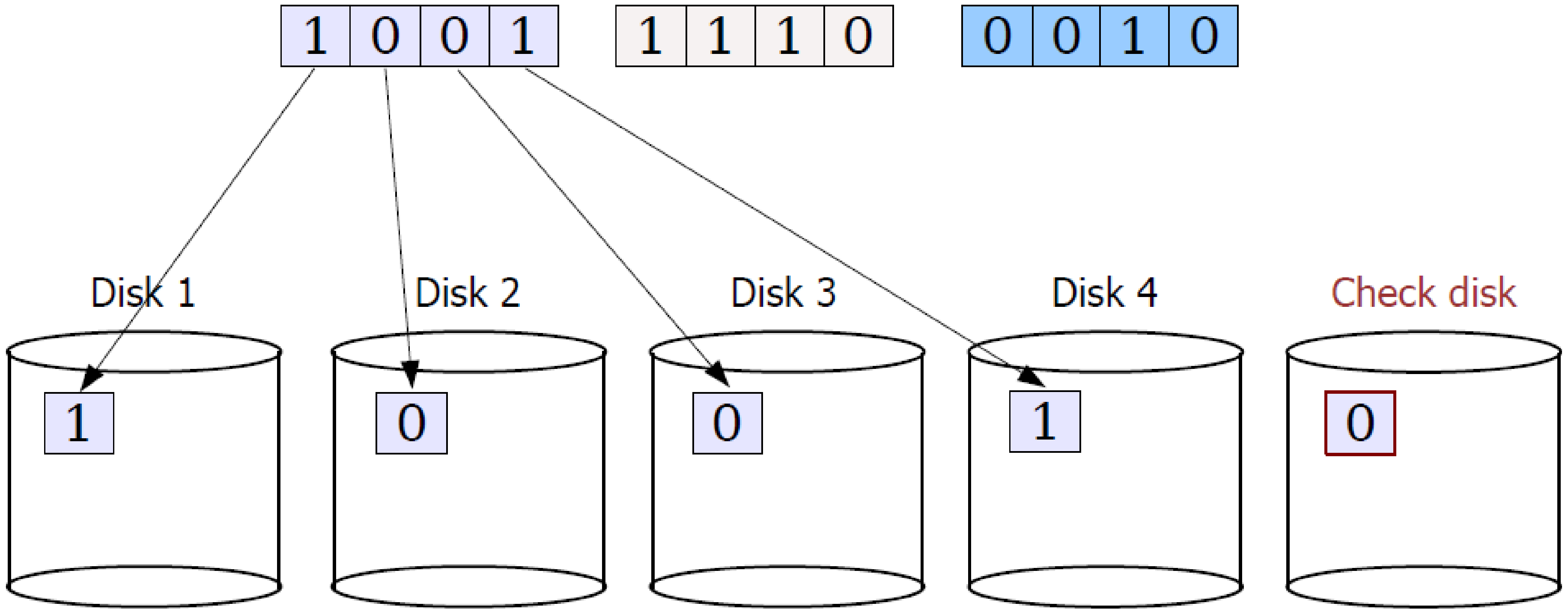
Advantages:

- If a single disk fails, can easily recompute the lost data from the parity code
- Can use one parity disk for *several* data disks (reduces cost)

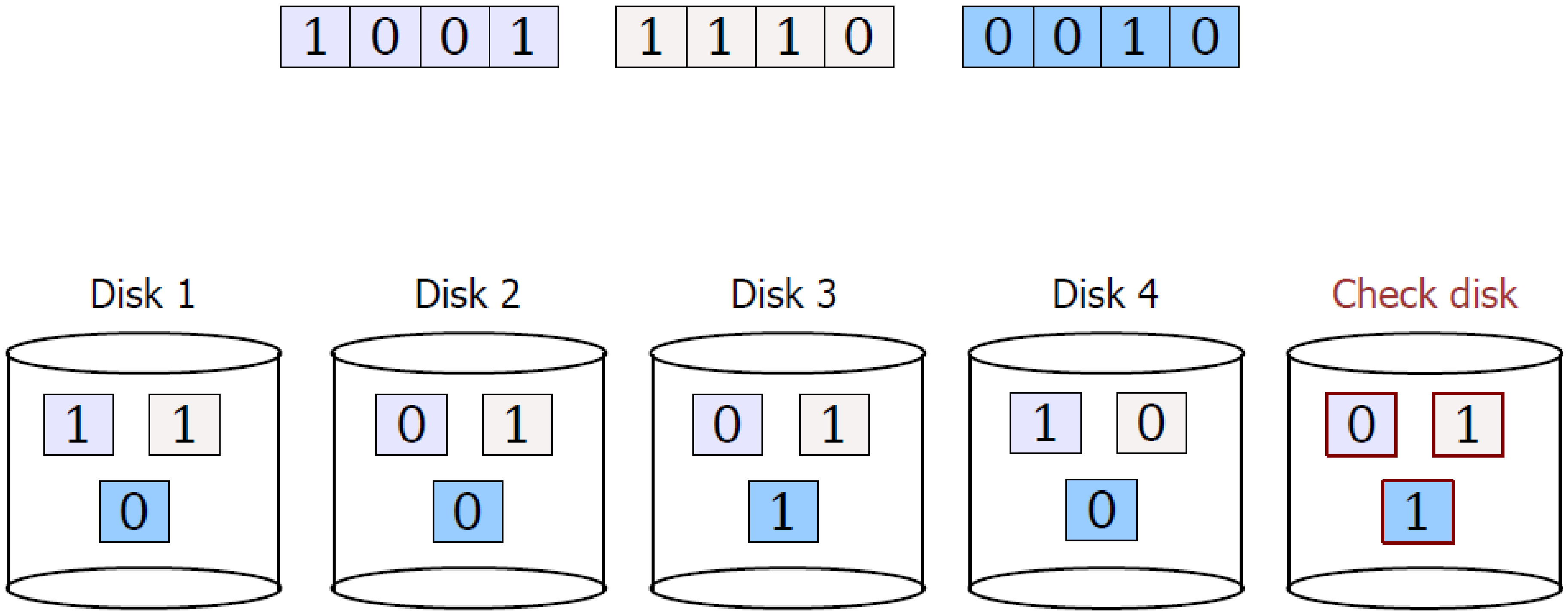
Disadvantages:

- Each write to a block must update the corresponding parity block as well

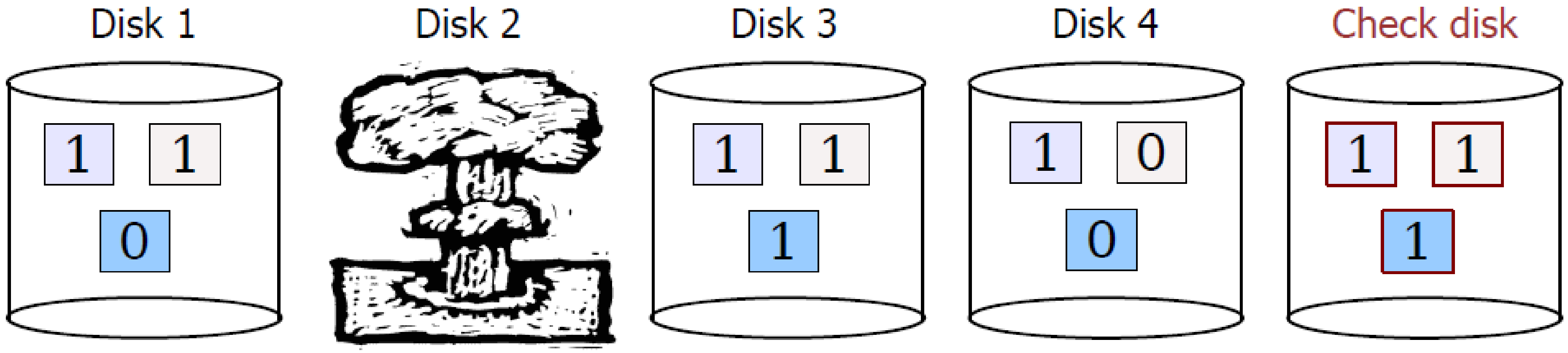
RAID-3 Example



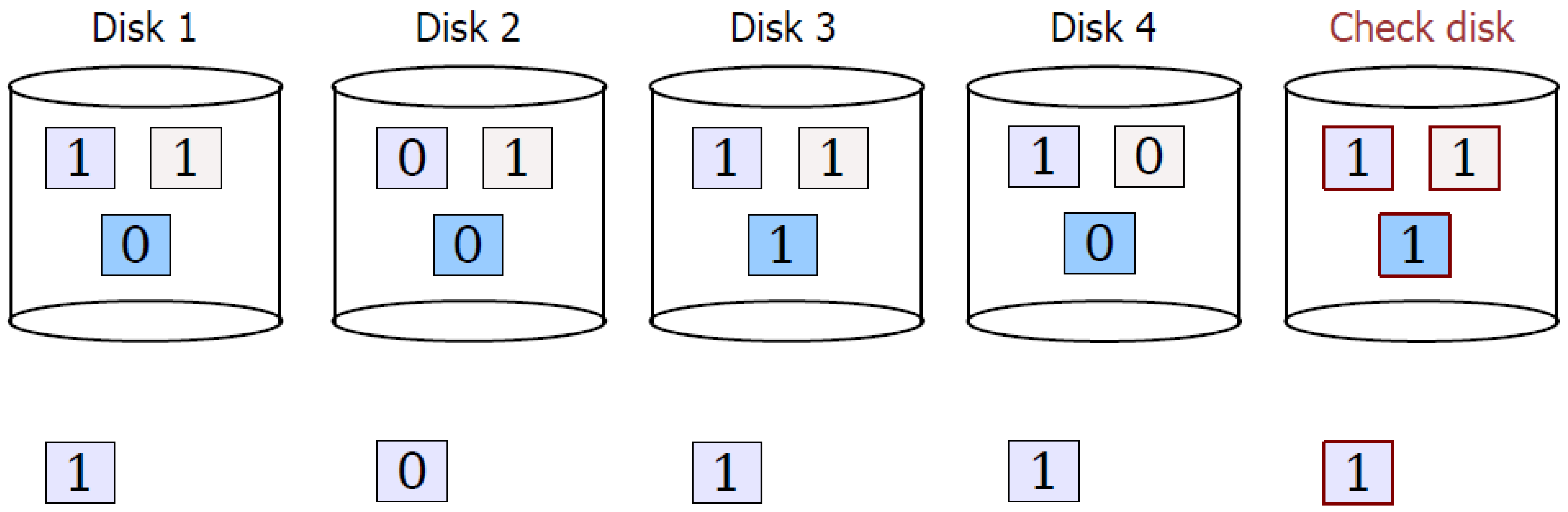
RAID-3


1 0 0 1
1 1 1 0
0 0 1 0
Disk 1
1 1
0
Disk 2
0 1
0
Disk 3
0 1
1
Disk 4
1 0
0
Check disk
0 1
1

RAID-3



RAID-3



1. Read back data from other disks
2. Recalculate lost data from parity code
3. Rebuild data on lost disk

When to Use Raid-3

- When we serve a large file that does a sequential read access.
- When a small number of read requests come in, it performs poorly.
  - So, We have Raid-4

RAID-4

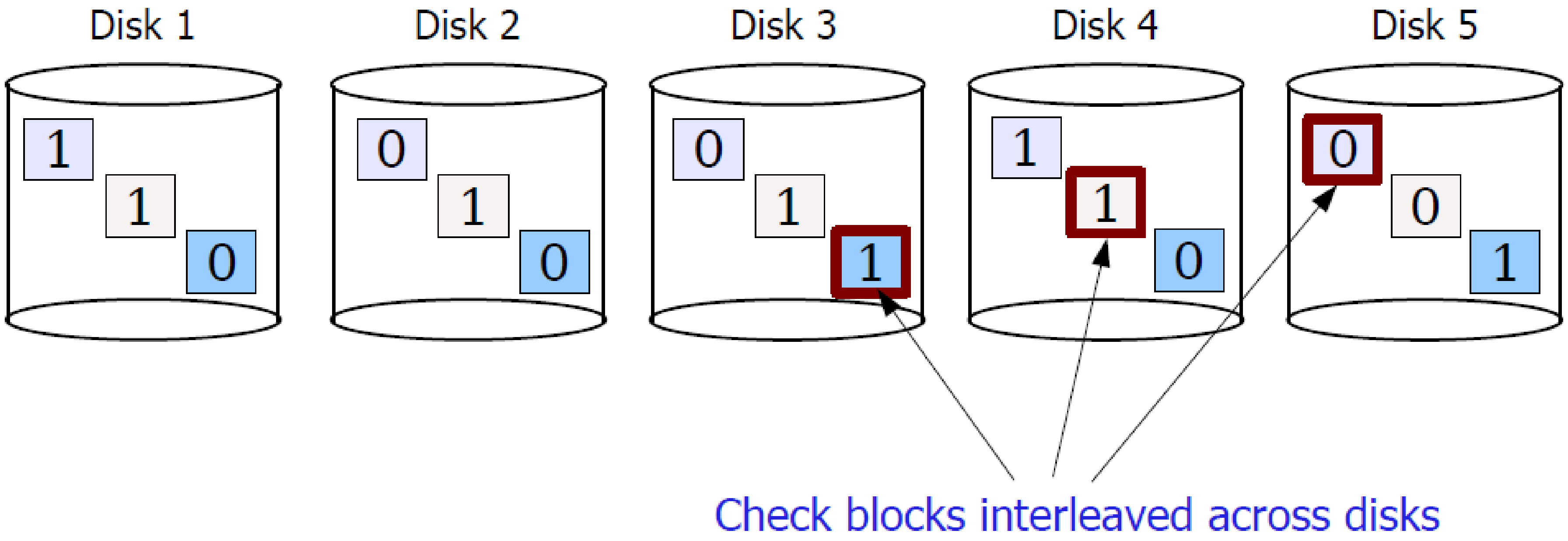
- Block Level Striping, 1 Parity Disk
  - Can Serve Simultaneous Read Reqeust
  - Cannot Serve simultaneous write request

- For writing, the parity disk becomes a bottleneck

RAID-5

Another approach: Interleaved check blocks (“RAID 5”)

- Rotate the assignment of data blocks and check blocks across disks
- Avoids the bottleneck of a single disk for storing check data
- Allows multiple reads/writes to occur in parallel (since different disks affected)


Disk 1
Disk 2
Disk 3
Disk 4
Disk 5
1
1
0
0
1
0
0
1
1
1
1
0
0
0
1
Check blocks interleaved across disks