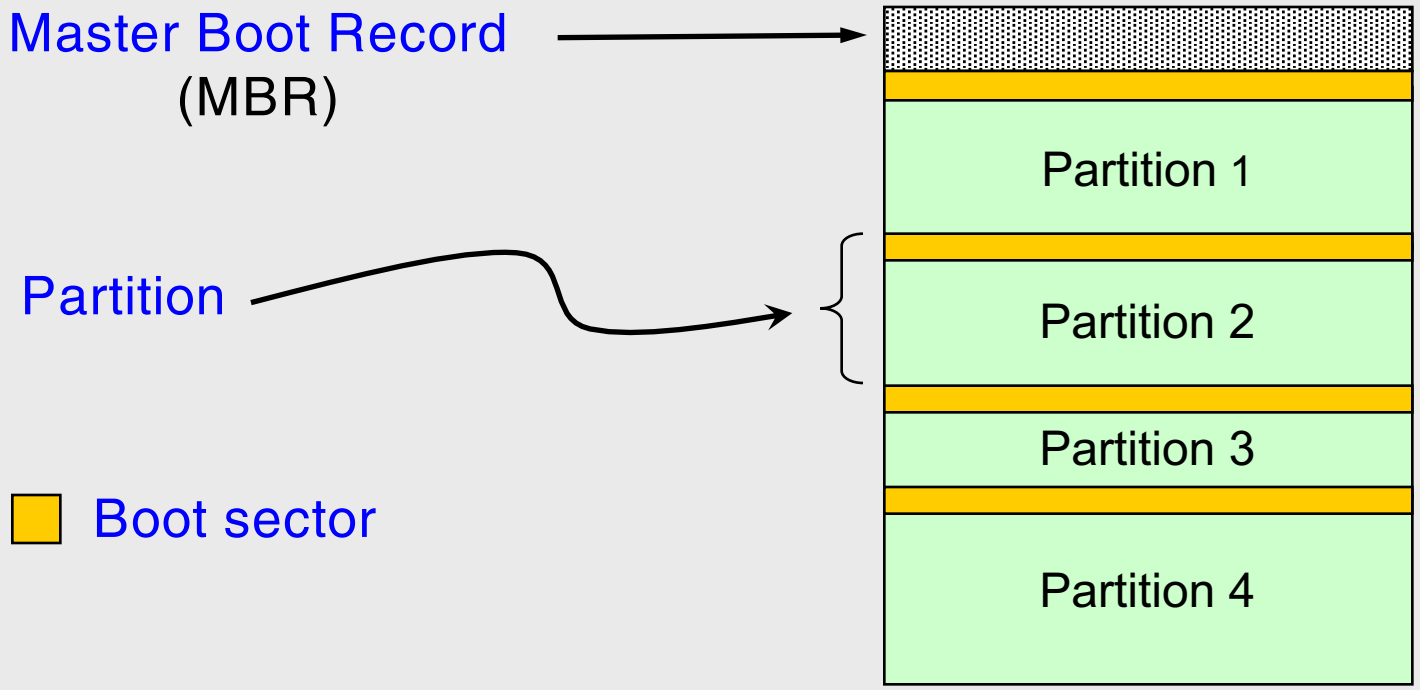
Định thời đĩa

Nội dung

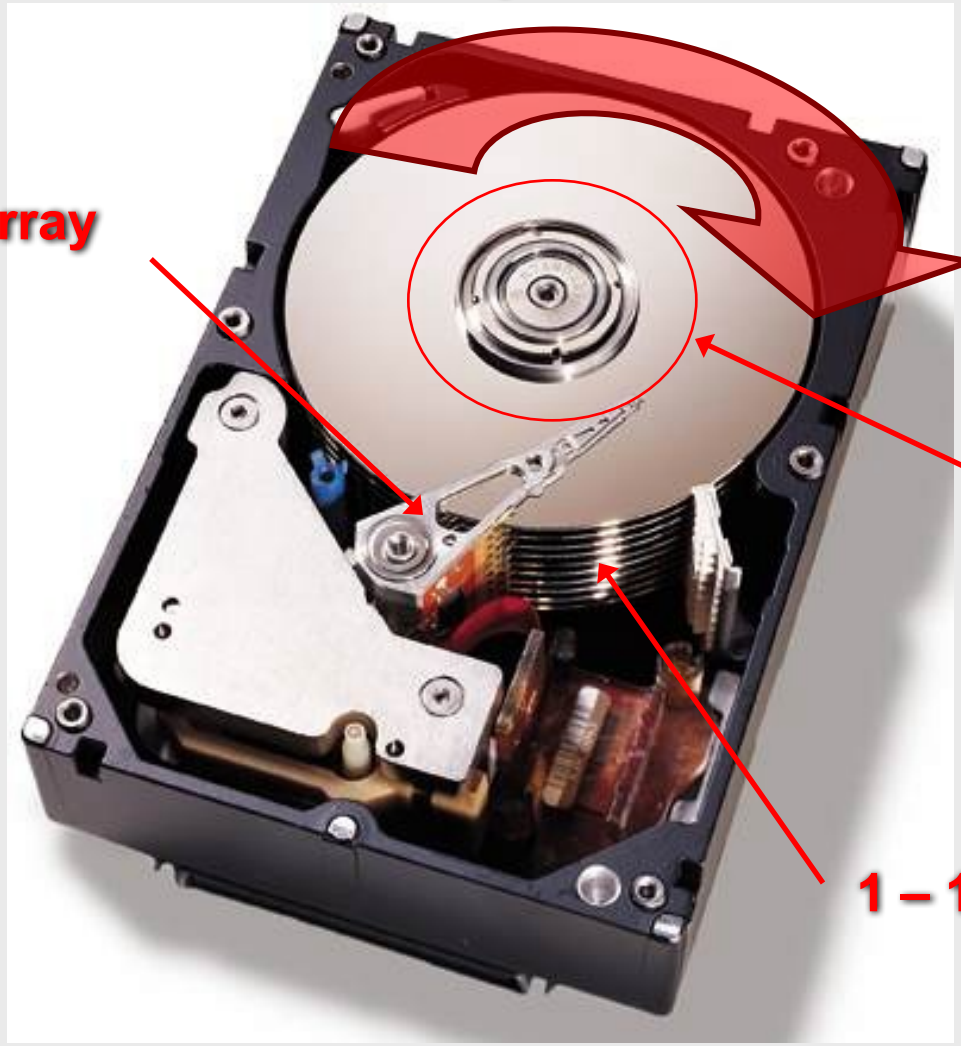
- Bên trong đĩa cứng
- Các giải thuật định thời truy cập đĩa
- Định dạng, phân vùng, raw disk
- RAID (Redundant Arrays of Independent Disks)

Tổ chức của đĩa cứng

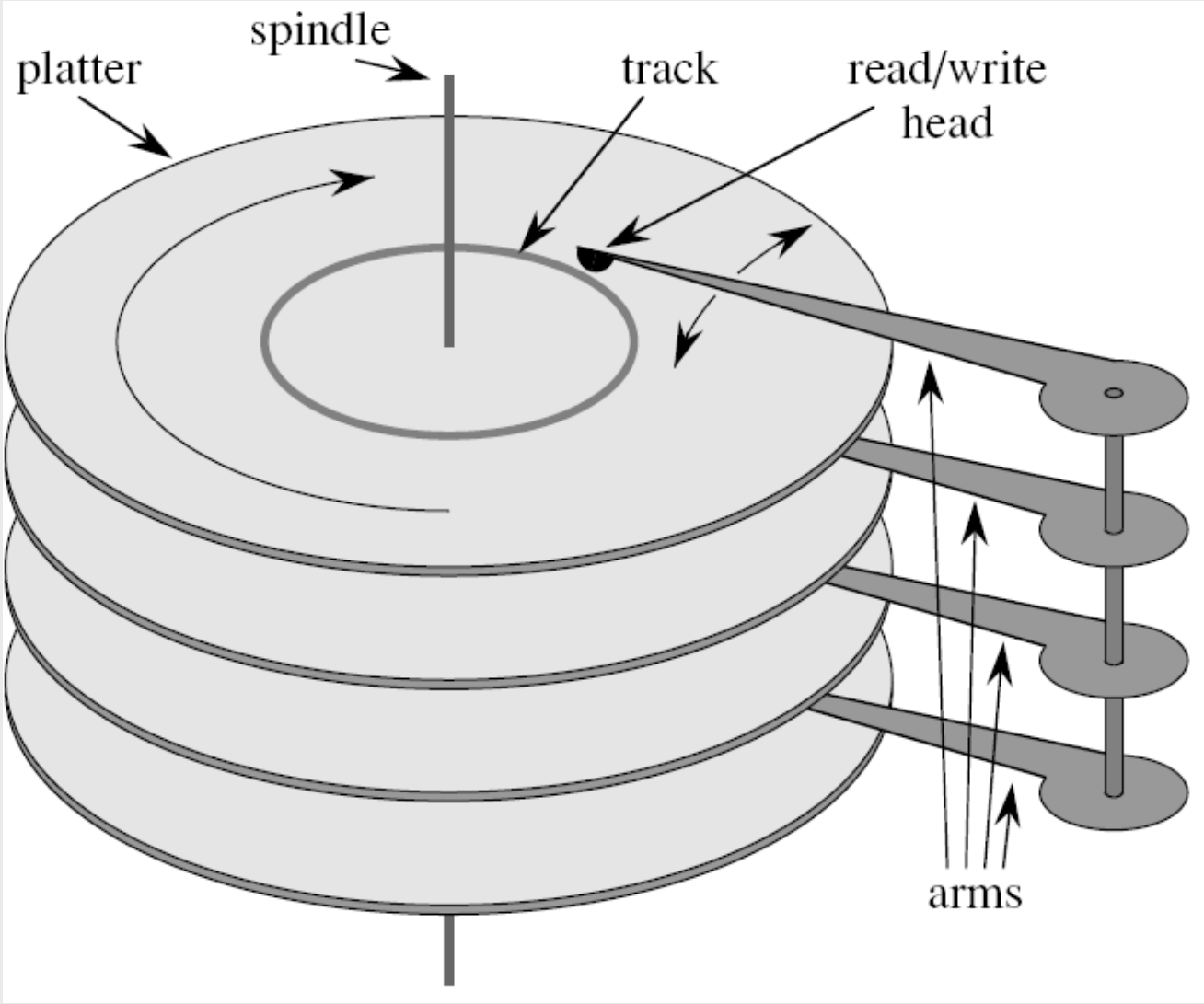
- Đĩa cứng trong hệ thống PC



Disk Anatomy



Bên trong đĩa cứng

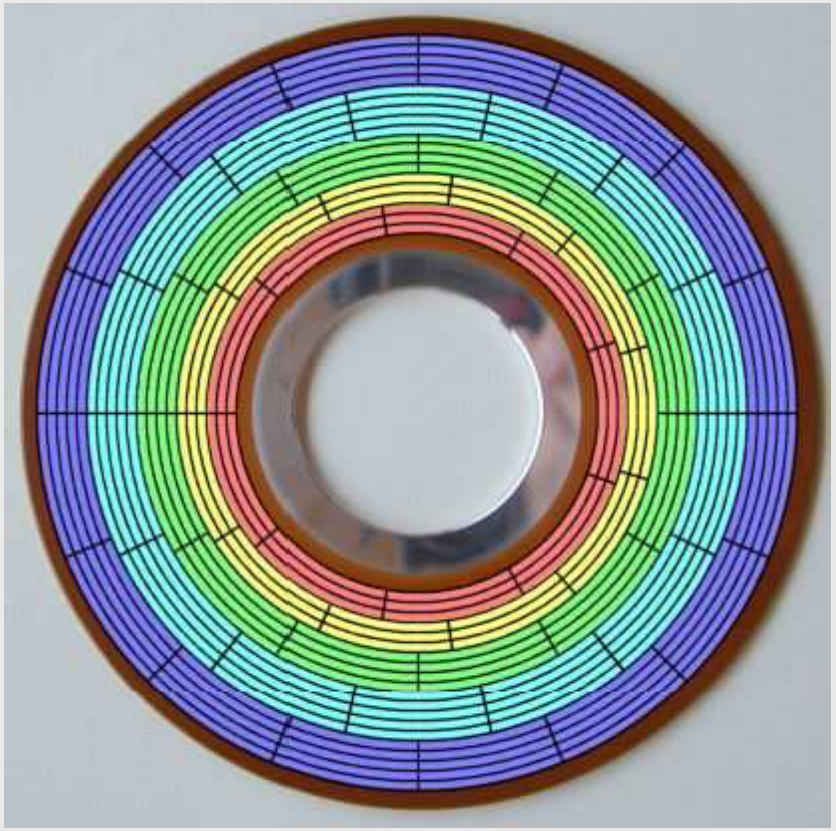


Các tham số của đĩa

- Thời gian đọc/ghi dữ liệu trên đĩa bao gồm
  - Seek time: thời gian di chuyển đầu đọc/ghi để định vị đúng track/cylinder, phụ thuộc tốc độ/cách di chuyển của đầu đọc/ghi
  - Latency (Rotational delay): thời gian đầu đọc chờ đến đúng sector cần đọc, phụ thuộc tốc độ quay của đĩa
  - Transfer time: thời gian chuyển dữ liệu từ đĩa vào bộ nhớ hoặc ngược lại, phụ thuộc băng thông kênh truyền giữa đĩa và bộ nhớ
- Disk I/O time = seek time + rotational delay + transfer time

Modern disks

- Modern hard drives use zoned bit recording
  - Disks are divided into zones with more sectors on the outer zones than the inner ones (why?)



Addressing Disks

- What the OS knows about the disk
  - Interface type (IDE/SCSI/SATA), unit number, number of sectors
- What happened to sectors, tracks, etc?
  - Old disks were addressed by cylinder/head/sector (CHS)
  - Modern disks are addressed using a linear addressing scheme
    - LBA = logical block address
    - As an example, LBA = 0 .. 586,072,367 for a 300 GB disk
- Who uses sector numbers?
  - File system software assign logical blocks to files
  - Terminology
    - To disk people, “block” and “sector” are the same
    - To file system people, a “block” is some fixed number of sectors

Disk Addresses vs Scheduling

- Goal of OS disk-scheduling algorithm
  - Maintain queue of requests
  - When disk finishes one request, give it the ‘best’ request
    - e.g., whichever one is closest in terms of disk geometry
- Goal of disk's logical addressing
  - Hide messy details of which sectors are located where
- Oh, well
  - Older OS's tried to understand disk layout
  - Modern OS's just assume nearby sector numbers are close
  - Experimental OS's try to understand disk layout again
  - Next few slides assume “old” / “experimental”, not “modern”

Tăng hiệu suất truy cập đĩa

Các giải pháp

- Giảm kích thước đĩa
- Tăng tốc độ quay của đĩa
- Định thời các tác vụ lên đĩa (disk scheduling)
- Bố trí ghi dữ liệu trên đĩa
- Bố trí các file thường sử dụng vào vị trí thích hợp
- Chọn kích thước của logical block
- Read ahead
  - Đọc sẵn các khối dữ liệu trước khi ứng dụng yêu cầu dựa trên nguyên lý cục bộ.

Hiệu suất truy cập đĩa (1)

- Đặc trưng về đáp ứng yêu cầu đĩa (performance metric)
  - Thông năng (throughput) – số lượng tác vụ hoàn tất trong một đơn vị thời gian
  - Độ lợi (disk utilization) – phần thời gian truyền dữ liệu chiếm trong disk I/O time
  - Deadline – thời hạn hoàn tất của một yêu cầu
  - Thời gian đáp ứng (response time) – thời gian từ lúc yêu cầu đến lúc yêu cầu được phục vụ xong
  - Công bằng (fairness)

Hiệu suất truy cập đĩa (2)

- Các tiêu chí tối ưu ‘tự nhiên’
  - Tối đa thông năng
  - Tối đa độ lợi
  - Tối thiểu số lượng các deadline không giữ được
  - Tối thiểu thời gian đáp ứng trung bình
  - Công bằng với mọi yêu cầu đĩa

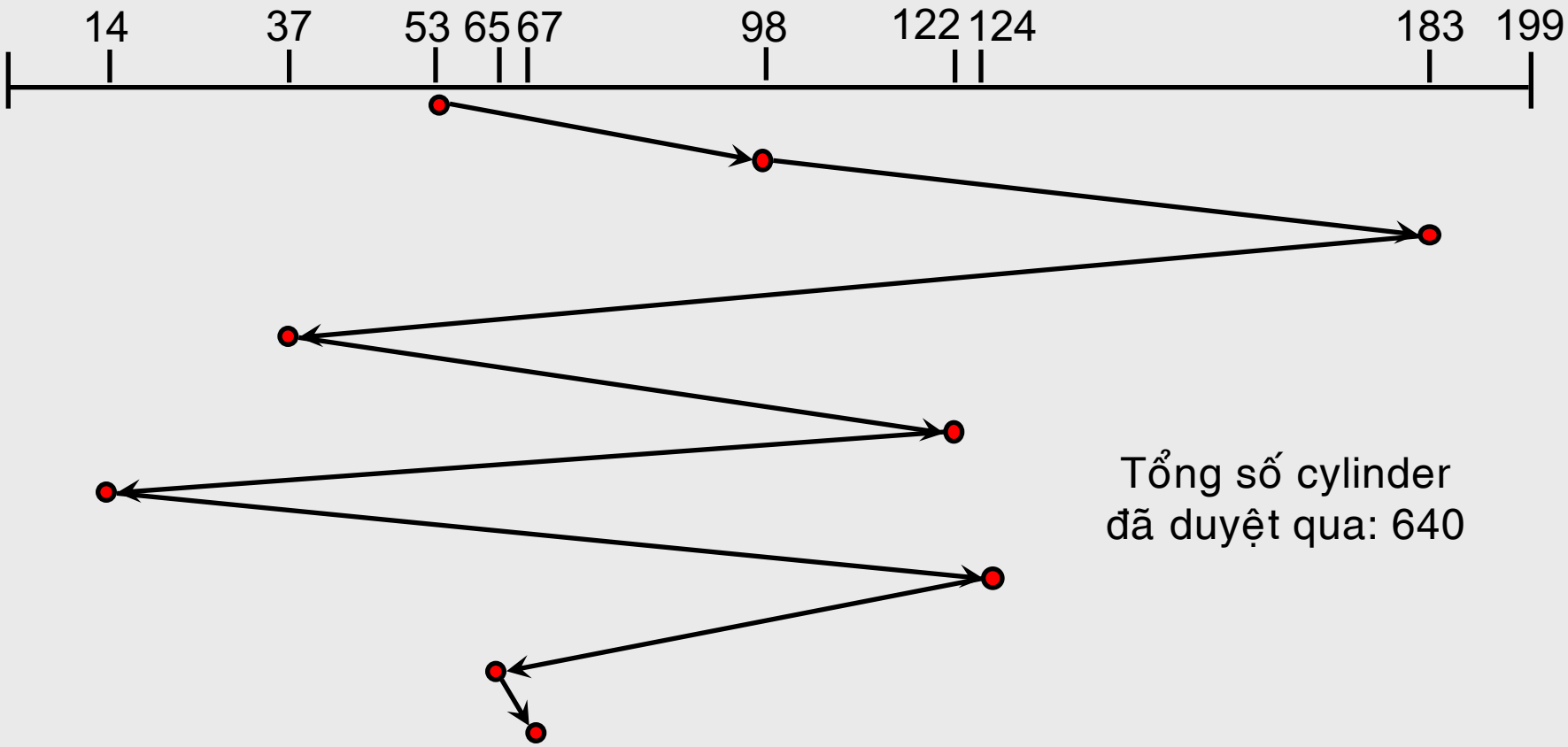
Định thời truy cập đĩa

- Ý tưởng chính
  - Sắp xếp lại thứ tự của các yêu cầu đĩa để thỏa tiêu chí tối ưu của hệ thống
- Các giải thuật định thời truy cập đĩa
  - First Come, First Served (FCFS)
  - Shortest-Seek-Time First (SSTF, SSF)
  - SCAN
  - C-SCAN (Circular SCAN)
  - LOOK (C-LOOK)

First Come First Served (FCFS)

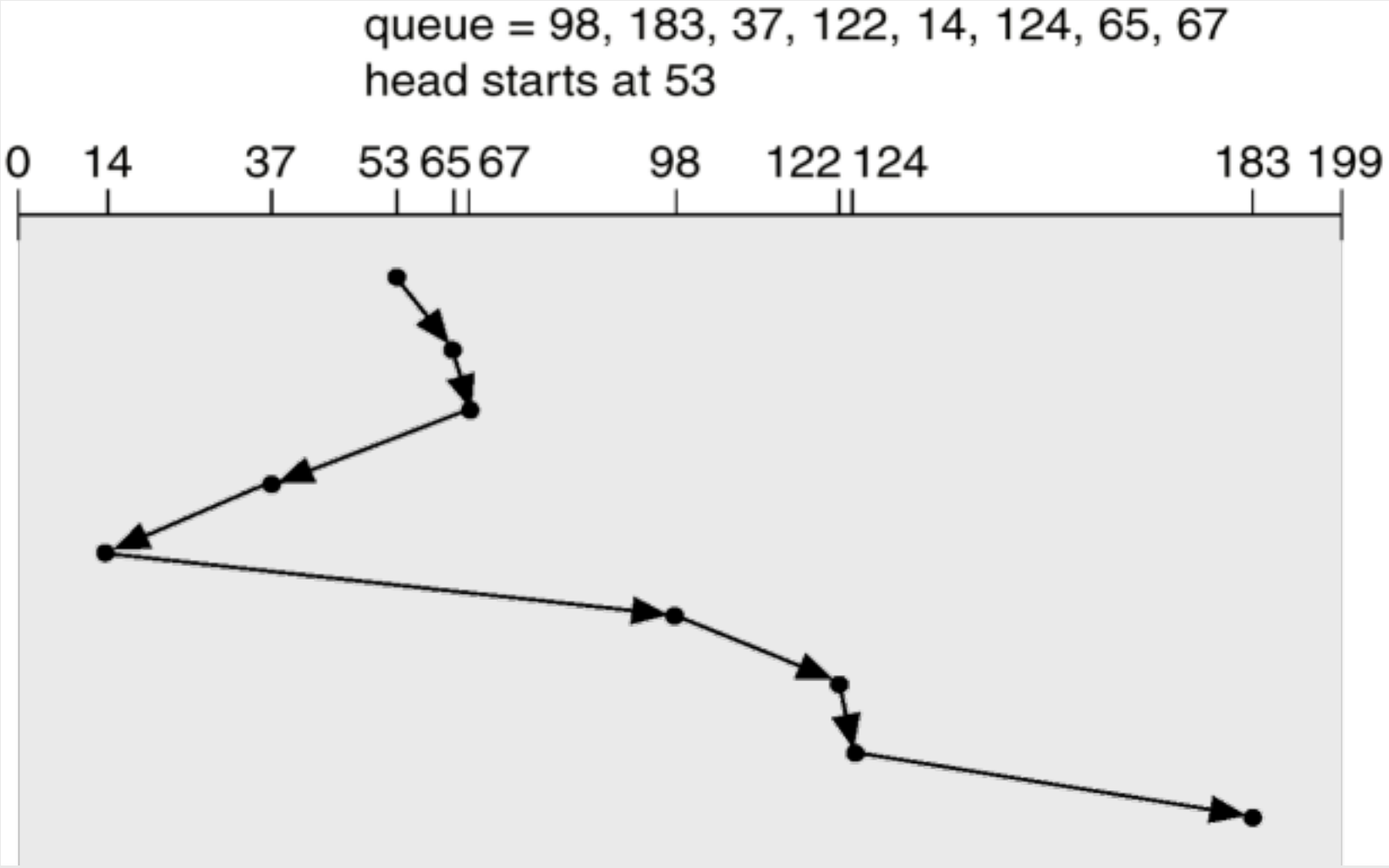
Hàng đợi (cylinder number): 98, 183, 37, 122, 14, 124, 65, 67

Đầu đọc đang ở cylinder số 53

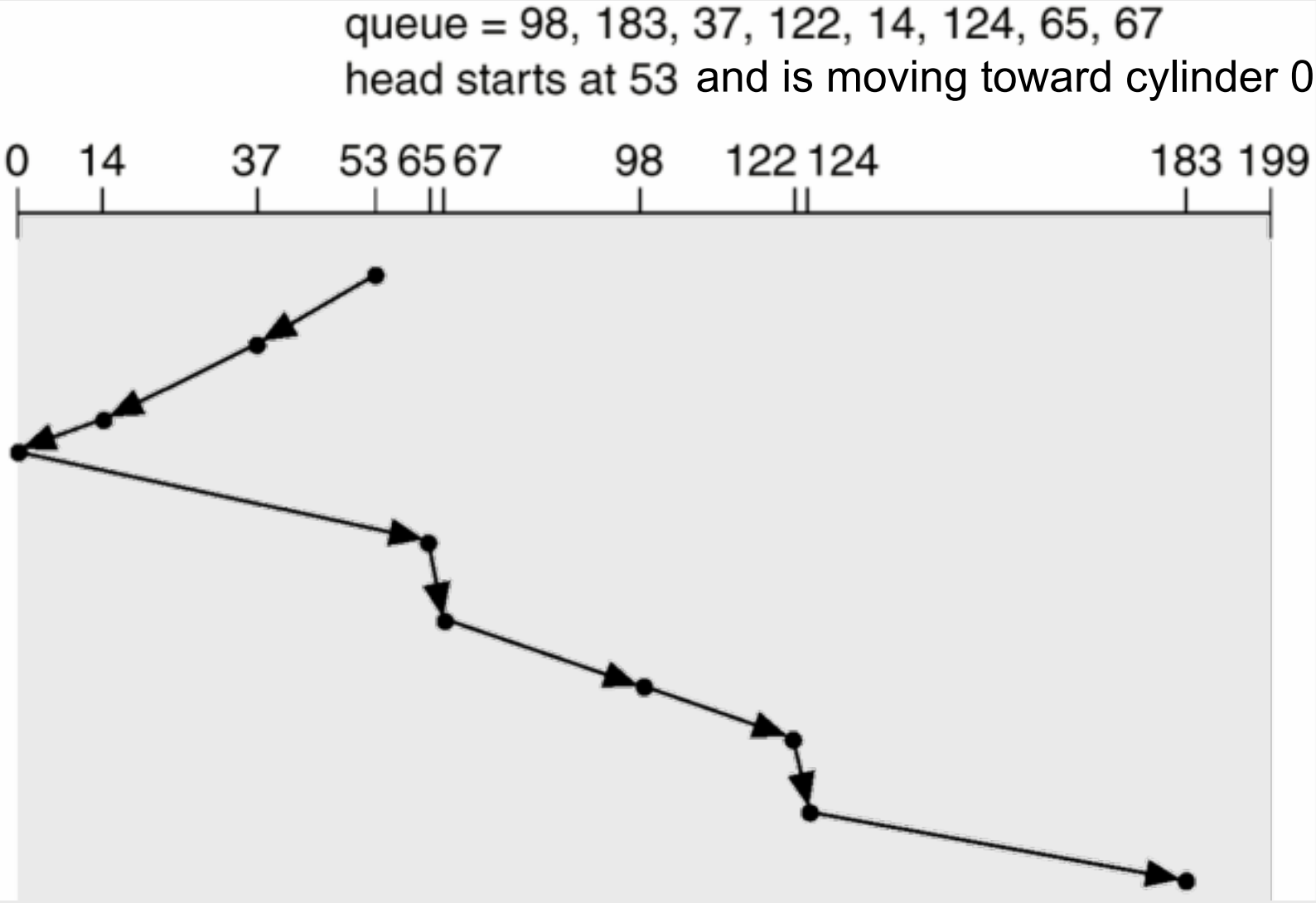


Tổng số cylinder
đã duyệt qua: 640

Shortest-Seek-Time First (SSTF)

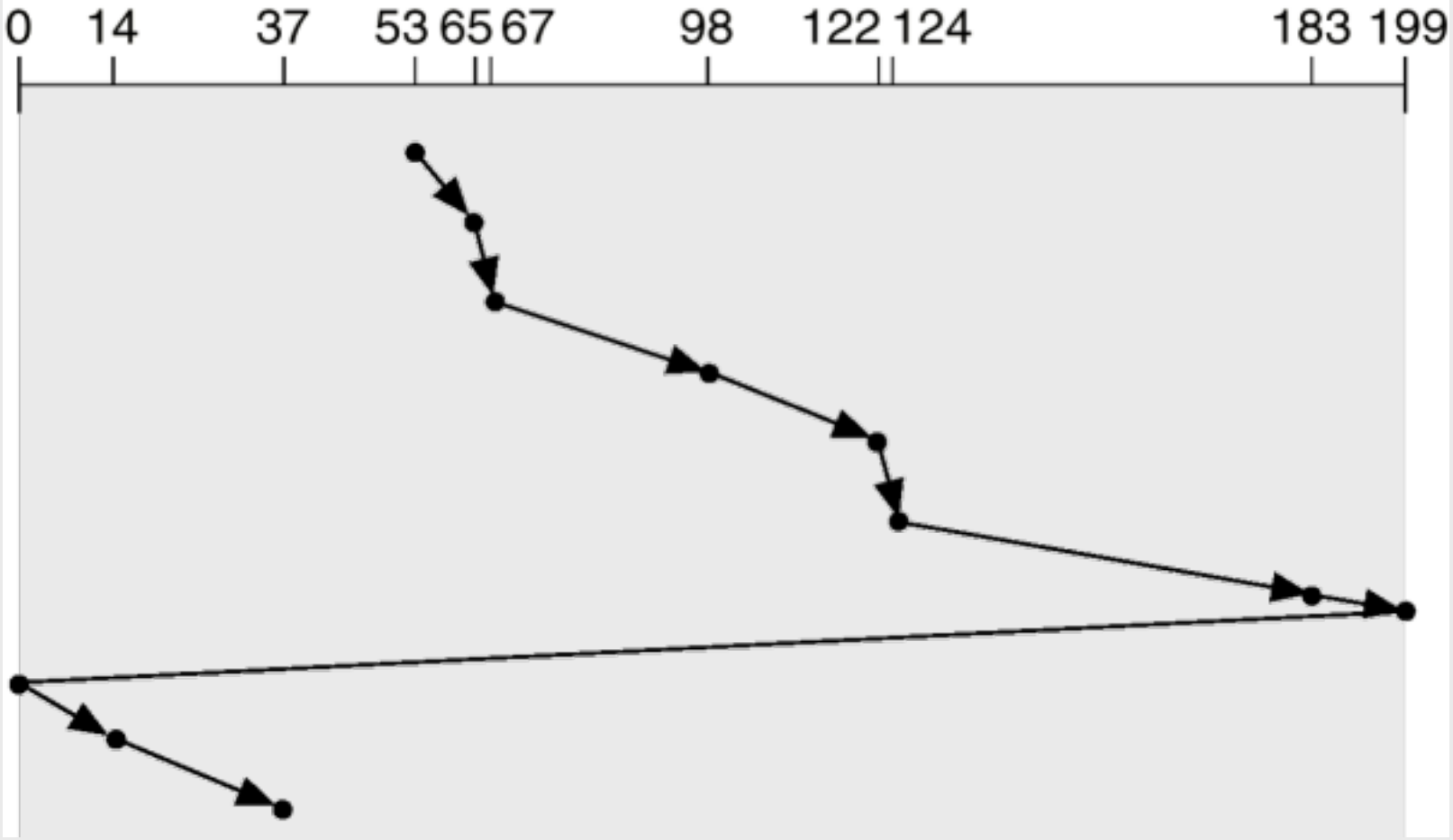


SCAN (elevator algorithm)



C-SCAN (Circular SCAN)

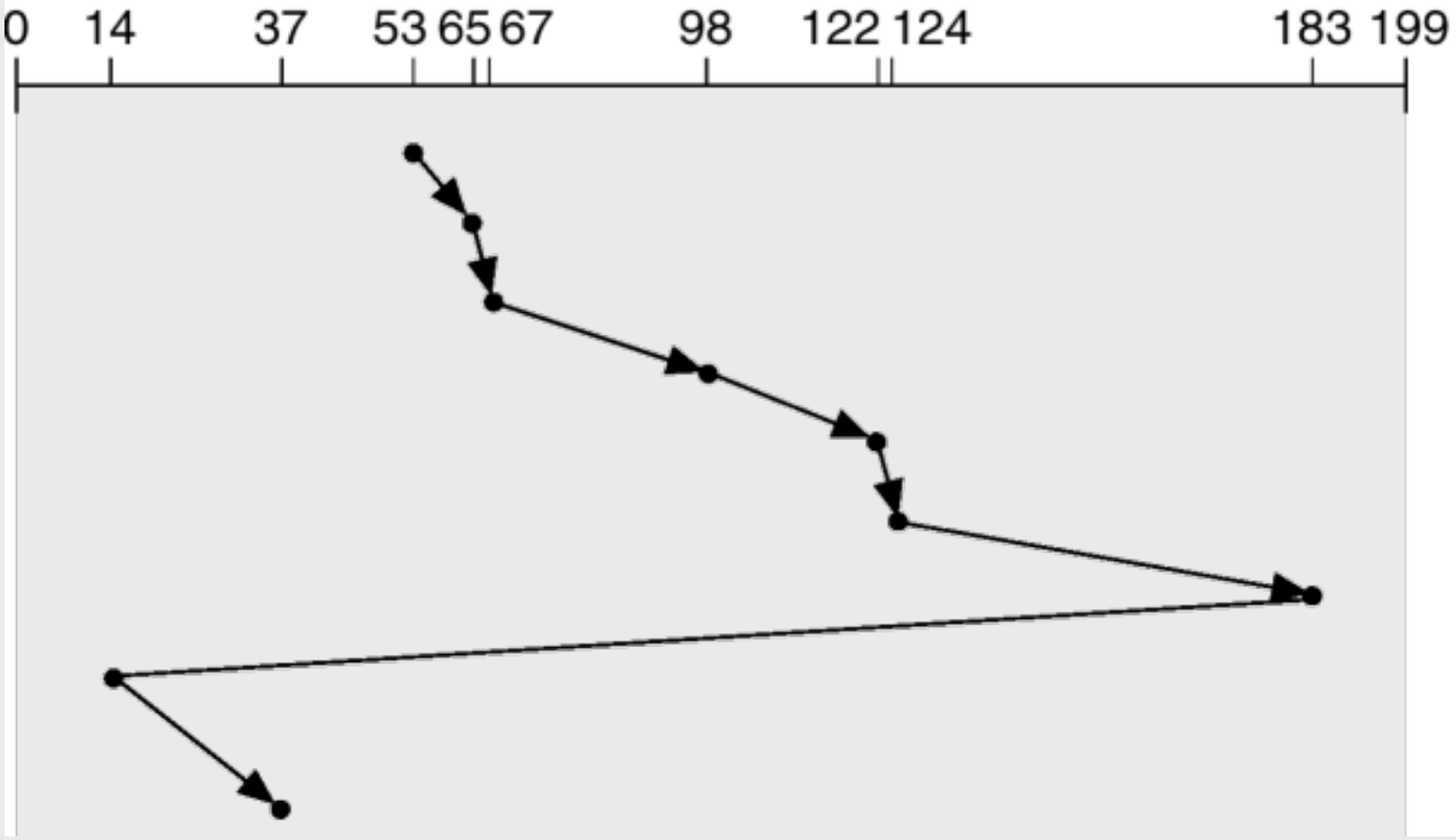
queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53 and is servicing on the way to cyl. 199



C-LOOK

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

head starts at 53 and is servicing on the way to cyl. 199



Đánh giá giải thuật định thời đĩa

- FCFS thỏa fairness nhưng không quan tâm đến tối đa độ lợi đĩa
- SSTF có mục tiêu là tối đa độ lợi đĩa nhưng không thỏa fairness

Bài tập

- **Vùng cylinder của đĩa đi từ 0 đến 199. Cho các yêu cầu truy cập đĩa tại các cylinder sau: 98, 180, 35, 120, 10, 128, 66, 70. Đầu đọc đang ở cylinder số 40 và trước đó phục vụ cylinder 23. Hãy tính số cylinder cần phải duyệt qua khi dùng các giải thuật:**
  - **FCFS,**
  - **SSTF,**
  - **SCAN,**
  - **C-SCAN,**
  - **LOOK,**
  - **C-LOOK.**

THAM KHẢO

Quản lý đĩa: Định dạng (formatting)

- Định dạng cấp thấp: định dạng vật lý, chia đĩa thành nhiều sector
  - Mỗi sector có cấu trúc dữ liệu đặc biệt: header – data – trailer

  - Header và trailer chứa các thông tin dành riêng cho disk controller như chỉ số sector và error-correcting code (ECC)
  - Khi controller ghi dữ liệu lên một sector, trường ECC được cập nhật với giá trị được tính dựa trên dữ liệu được ghi
  - Khi đọc sector, giá trị ECC của dữ liệu được tính lại và so sánh với trị ECC đã lưu để kiểm tra tính đúng đắn của dữ liệu

Quản lý đĩa: Phân vùng (partitioning)

- Phân vùng: chia đĩa thành nhiều vùng (partition), mỗi vùng là một chuỗi block liên tục
  - Mỗi partition được xem như một "đĩa luận lý" riêng biệt
- Định dạng luận lý cho partition: tạo một hệ thống file (FAT, ext2, NTFS…), bao gồm
  - Lưu các cấu trúc dữ liệu khởi đầu của hệ thống file lên partition
  - Tạo cấu trúc dữ liệu quản lý không gian trống và không gian đã cấp phát (DOS: FAT; UNIX: superblock và i-node list)

Ví dụ định dạng một partition

Quản lý đĩa: Raw disk

- Raw disk: partition không có hệ thống file
- I/O lên raw disk được gọi là raw I/O
  - đọc hay ghi trực tiếp các block
  - không dùng các dịch vụ của file system (buffer cache, file locking, prefetching, cấp phát không gian trống, định danh file, và thư mục)
- Ví dụ
  - Một số hệ thống cơ sở dữ liệu chọn dùng raw disk để có hiệu suất đĩa cao hơn

Quản lý không gian tráo đổi (swap space)

- Swap space
  - không gian đĩa được sử dụng để mở rộng không gian địa chỉ ảo trong kỹ thuật bộ nhớ ảo
  - Mục tiêu quản lý: hiệu suất cao cho hệ thống quản lý bộ nhớ ảo (vd demand paging)
  - Hiện thực
    - chiếm partition riêng, vd *swap partition* của Linux
    - hoặc qua một file system, vd file pagefile.sys của Windows
    - Thường kèm theo caching hoặc dùng phương pháp cấp phát liên tục

RAID Introduction

- Disks act as bottlenecks for both system performance and storage reliability
- A disk array consists of several disks which are organized to increase performance and improve reliability
  - Performance is improved through data striping
  - Reliability is improved through redundancy
- Disk arrays that combine data striping and redundancy are called Redundant Arrays of Independent Disks, or RAID
- There are several RAID schemes or levels

  - Slide from CMPT 354
    http://sleepy.cs.surrey.sfu.ca/cmpt/courses/archive/fall2005spring2006/cmpt354/notes

Data Striping

- A disk array gives the user the abstraction of a single, large, disk
  - When an I/O request is issued, the physical disk blocks to be retrieved have to be identified
  - How the data is distributed over the disks in the array affects how many disks are involved in an I/O request
- Data is divided into equal size partitions called striping units
  - The size of the striping unit varies by the RAID level
- The striping units are distributed over the disks using a round robin algorithm

KEY POINT – disks can be read in parallel, increasing the transfer rate

Striping Units – Block Striping

- Assume that a file is to be distributed across a 4 disk RAID system and that
  - Purely for the sake of illustration, blocks are only one byte! [here striping-unit size = block size]

Notional File – a series of bits, numbered so that we can distinguish them

1	2	3	4	5	6	7	8	9	10	11	12	13	12	15	16	17	18	19	20	21	22	23	24	…

Now distribute these bits across the 4 RAID disks using BLOCK striping:

1	2	3	4	5	6	7	8	33	34	35	36	37	38	39	40	65	66	67	68	69	70	71	72	…

9	10	11	12	13	14	15	16	41	42	43	44	45	46	47	48	73	74	75	76	77	78	79	80	…

17	18	19	20	21	22	23	24	49	50	51	52	53	54	55	56	81	82	83	84	85	86	87	88	…

25	26	27	28	29	30	31	32	57	58	59	60	61	62	63	64	89	90	91	92	93	94	95	96	…

Striping Units – Bit Striping

- Now here is the same file, and 4 disk RAID using bit striping, and again:
  - Purely for the sake of illustration, blocks are only one byte!

Notional File – a series of bits, numbered so that we can distinguish them

1	2	3	4	5	6	7	8	9	10	11	12	13	12	15	16	17	18	19	20	21	22	23	24	…

Now distribute these bits across the 4 RAID disks using BIT striping:

1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65	69	73	77	81	85	89	93	…

2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70	74	78	82	86	90	94	…

3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63	67	71	75	79	83	87	91	95	…

4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	…

Striping Units Performance

- A RAID system with D disks can read data up to D times faster than a single disk system
  - As the D disks can be read in parallel
  - For large reads* there is no difference between bit striping and block striping
    - *where some multiple of D blocks are to be read
  - Block striping is more efficient for many unrelated requests
    - With bit striping all D disks have to be read to recreate a single block of the data file
    - In block striping each disk can satisfy one of the requests, assuming that the blocks to be read are on different disks
- Write performance is similar but is also affected by the parity scheme

Reliability of Disk Arrays

- The mean-time-to-failure (MTTF) of a hard disk is around 50,000 hours, or 5.7 years
- In a disk array the MTTF (of a single disk in the array) increases
  - Because the number of disks is greater
- The MTTF of a disk array containing 100 disks is 21 days (= 50,000/100 hours)
  - Assuming that failures occur independently and
  - The failure probability does not change over time
  - Pretty implausible assumptions ☺
- Reliability is improved by storing redundant data

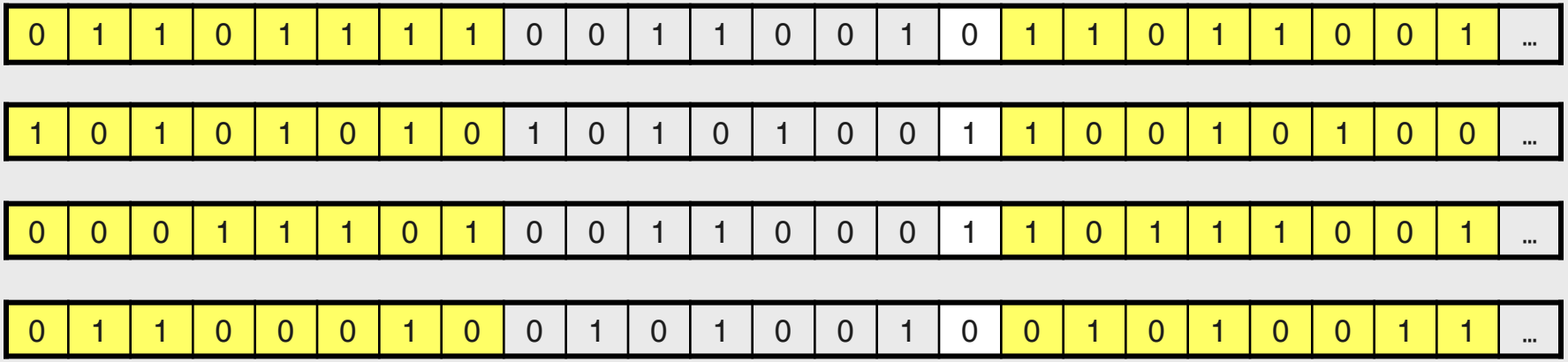
Redundancy

- Reliability of a disk array can be improved by storing redundant data
- If a disk fails, the redundant data can be used to reconstruct the data lost on the failed disk
  - The data can either be stored on a separate check disk or
  - Distributed uniformly over all the disks
- Redundant data is typically stored using a parity scheme
  - There are other redundancy schemes that provide greater reliability

Parity Scheme

- For each bit on the data disks there is a related parity bit on a check disk
  - If the sum of the bits on the data disks is even the parity bit is set to zero
  - If the sum of the bits is odd the parity bit is set to one
- The data on any one failed disk can be recreated bit by bit

Here is the 4 disk RAID system showing the actual bit values



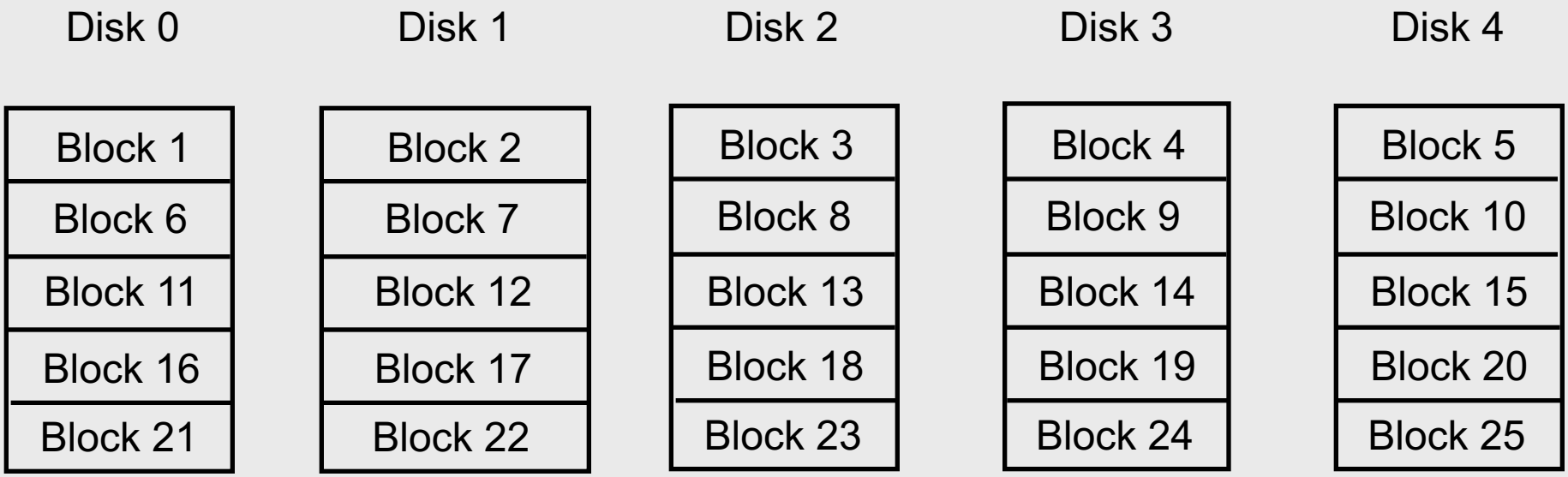
Here is a fifth CHECK DISK with the parity data

Parity Scheme and Reliability

- In RAID systems the disk array is partitioned into reliability groups
  - A reliability group consists of a set of data disks and a set of check disks
  - The number of check disks depends on the reliability level that is selected
- Given a RAID system with 100 disks and an additional 10 check disks the MTTF can be increased from 21 days to 250 years!

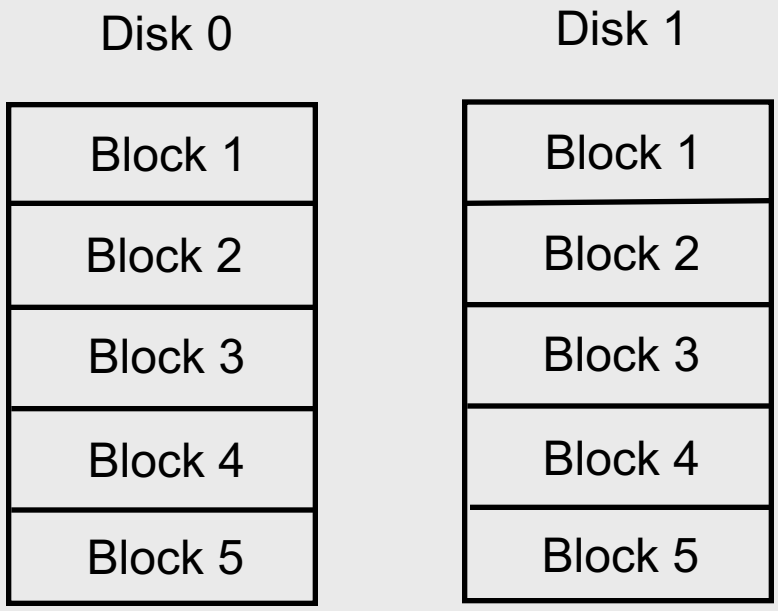
RAID Level 0: Nonredundant

- Uses data striping to increase the transfer rate
  - Good read performance
    - Up to D times the speed of a single disk
- No redundant data is recorded
  - The best write performance as redundant data does not have to be recorded
  - The lowest cost RAID level but
  - Reliability is a problem, as the MTTF increases linearly with the number of disks in the array
- With 5 data disks, only 5 disks are required


Disk 0
Disk 1
Disk 2
Disk 3
Disk 4
Block 1
Block 6
Block 11
Block 16
Block 21
Block 2
Block 7
Block 12
Block 17
Block 22
Block 3
Block 8
Block 13
Block 18
Block 23
Block 4
Block 9
Block 14
Block 19
Block 24
Block 5
Block 10
Block 15
Block 20
Block 25

RAID Level 1: Mirrored

- For each disk in the system an identical copy is kept, hence the term mirroring
  - No data striping, but parallel reads of the duplicate disks can be made, otherwise read performance is similar to a single disk
- Very reliable but the most expensive RAID level
  - Poor write performance as the duplicate disk has to be written to
    - These writes should not be performed simultaneously in case there is a global system failure
- With 4 data disks, 8 disks are required

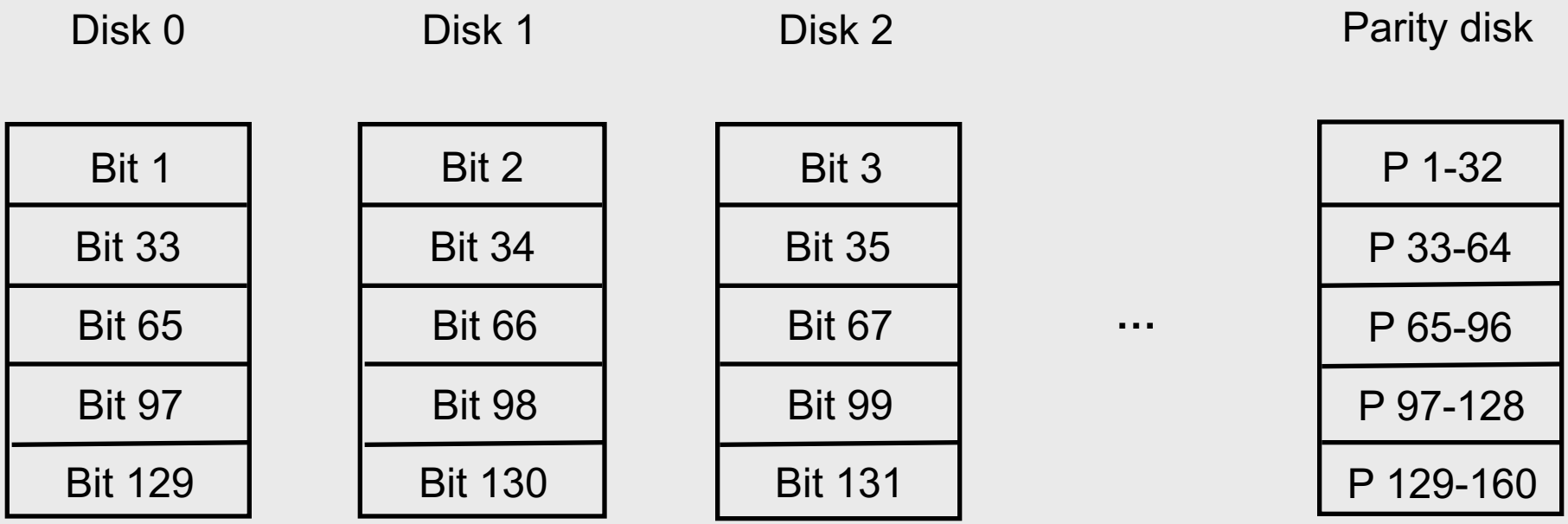

Disk 0
Block 1
Block 2
Block 3
Block 4
Block 5
Disk 1
Block 1
Block 2
Block 3
Block 4
Block 5

RAID Level 2: Memory-Style ECC

- Not common because redundancy schemes such as bit-interleaved parity provide similar reliability at better performance and cost.

RAID Level 3: Bit-Interleaved Parity

- Uses bit striping
  - Good read performance for large requests
    - Up to D times the speed of a single disk
    - Poor read performance for multiple small requests
- Uses a single check disk with parity information
  - Disk controllers can easily determine which disk has failed, so the check disks are not required to perform this task
  - Writing requires a read-modify-write cycle
    - Read D blocks, modify in main memory, write D + C blocks

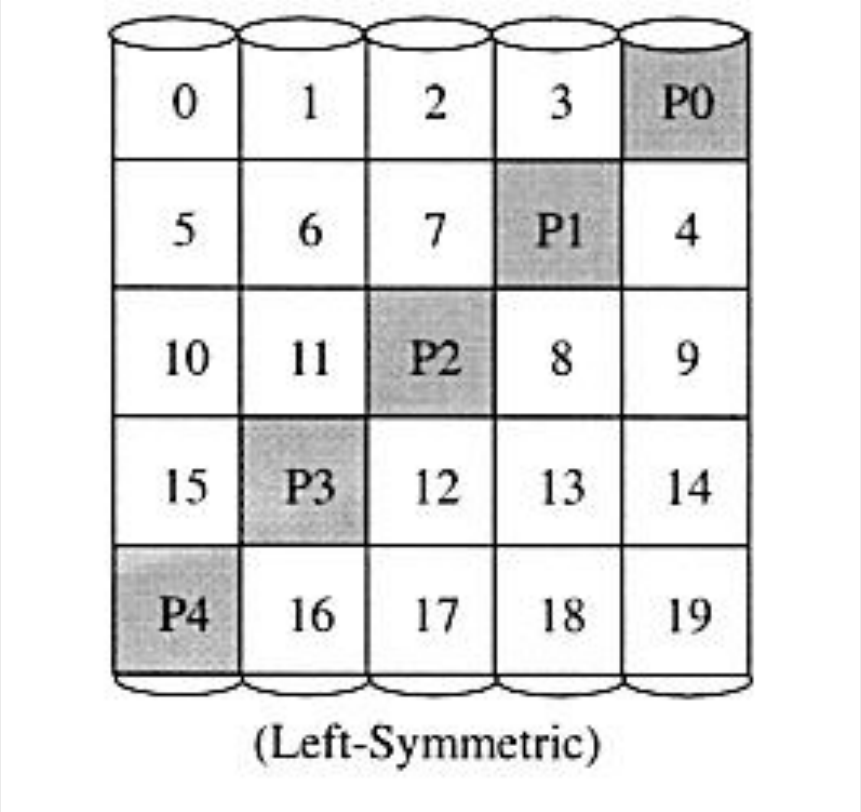

Disk 0
Disk 1
Disk 2
Parity disk
Bit 1
Bit 33
Bit 65
Bit 97
Bit 129
Bit 2
Bit 34
Bit 66
Bit 98
Bit 130
Bit 3
Bit 35
Bit 67
Bit 99
Bit 131
...
P 1-32
P 33-64
P 65-96
P 97-128
P 129-160

RAID Level 4: Block-Interleaved Parity

- Block-interleaved, parity disk array is similar to the bit-interleaved, parity disk array except that data is interleaved across disks in blocks of arbitrary size rather than in bits

RAID Level 5: Block-Interleaved Distributed Parity

- Uses block striping
  - Good read performance for large requests
    - Up to D times the speed of a single disk
    - Good read performance for multiple small requests that can involve all disks in the scheme
- Distributes parity information over all of the disks
  - Writing requires a read-modify-write cycle
    - But several write requests can be processed in parallel as the bottleneck of a single check disk has been removed
- Best performance for small and large reads and large writes
- With 4 disks of data, 5 disks are required with the parity information distributed across all disks



- Each square corresponds to a stripe unit. Each column of squares corresponds to a disk.
- P0 computes the parity over stripe units 0, 1, 2 and 3; P1 computes parity over stripe units 4, 5, 6 and 7; etc.