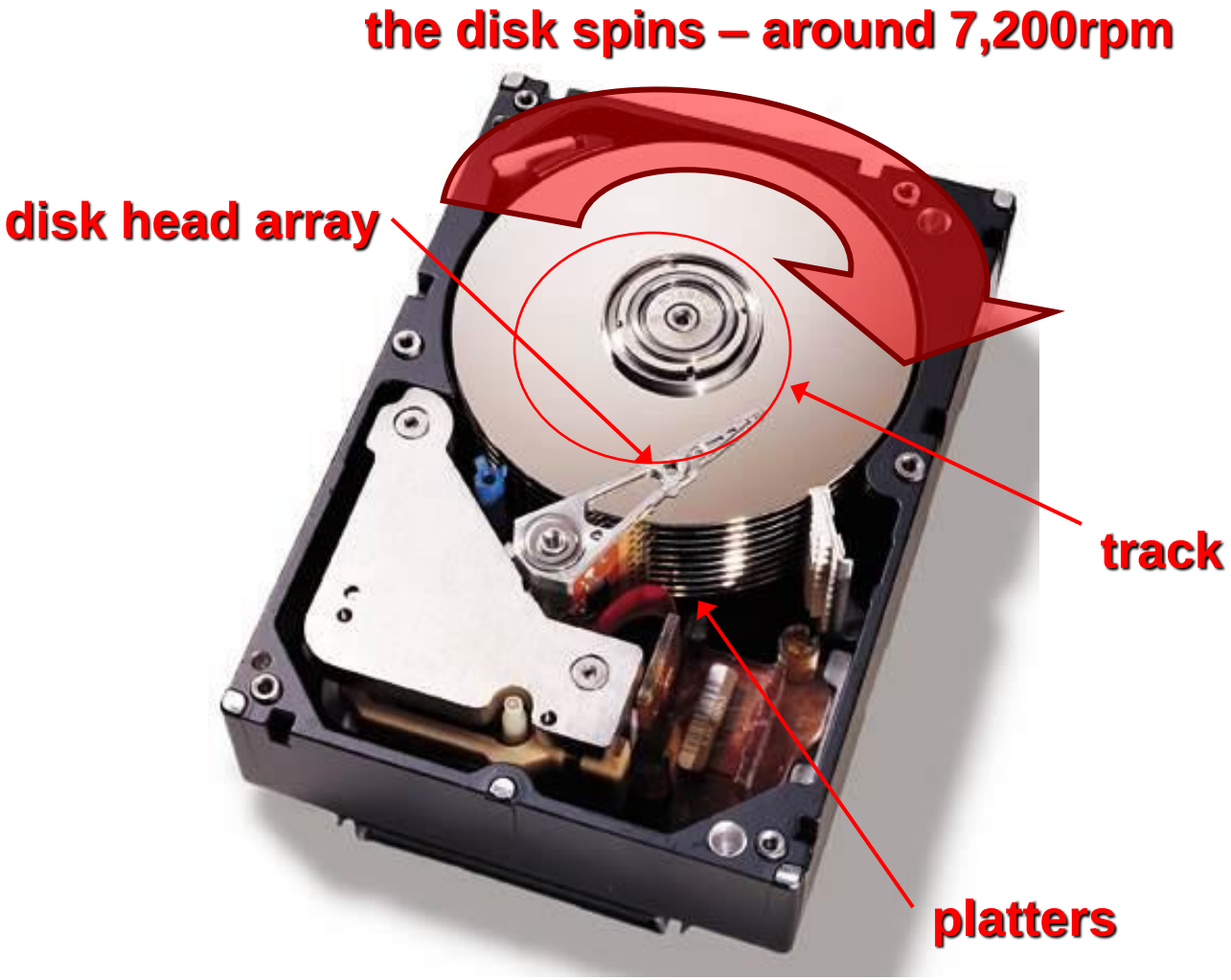
Hệ thống tập tin (đĩa cứng-hardisk)

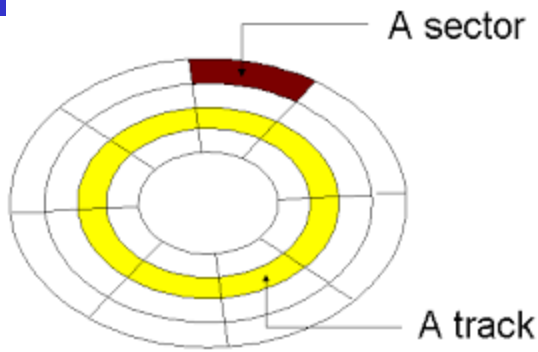
Đĩa cứng: Hệ thống tập tin

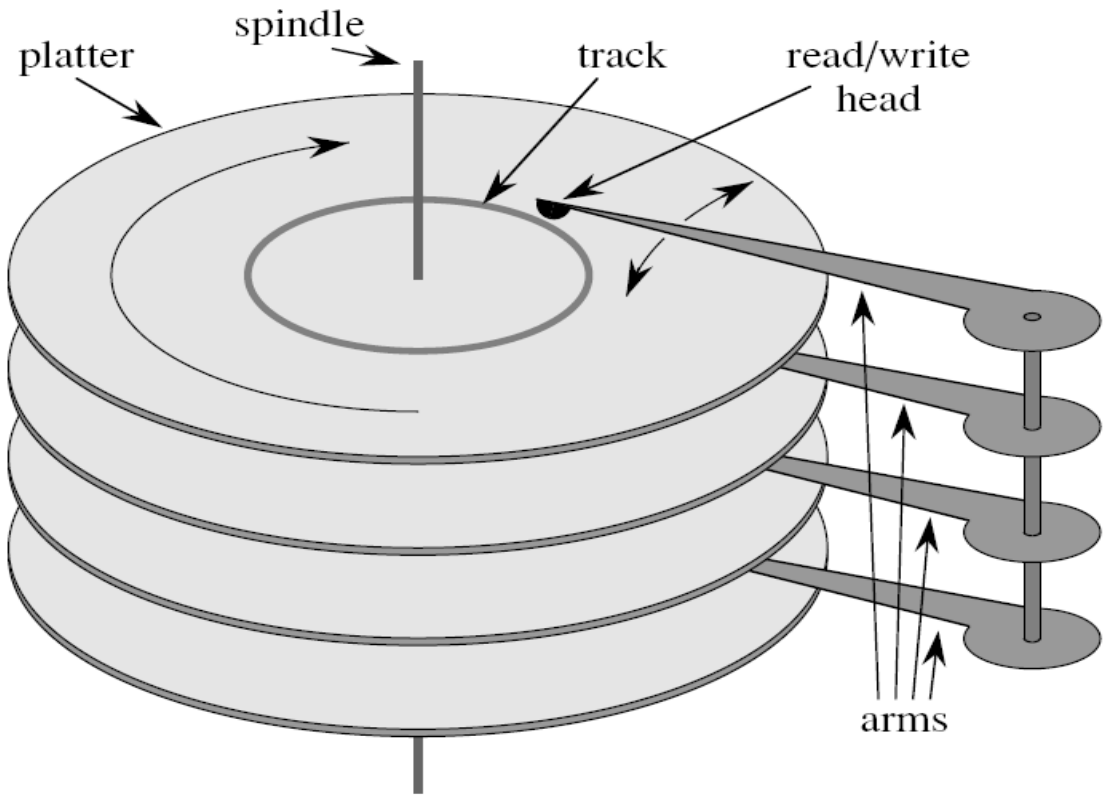
- Bên trong đĩa cứng
- Các giải thuật định thời truy cập đĩa
- Định dạng, phân vùng, raw disk
- RAID (Redundant Arrays of Independent (*Inexpensive*) Disks)

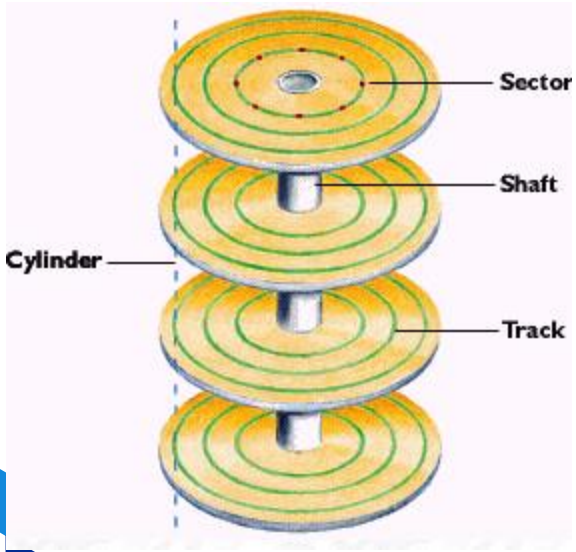
Giải phẫu bên trong đĩa



Bên trong đĩa cứng

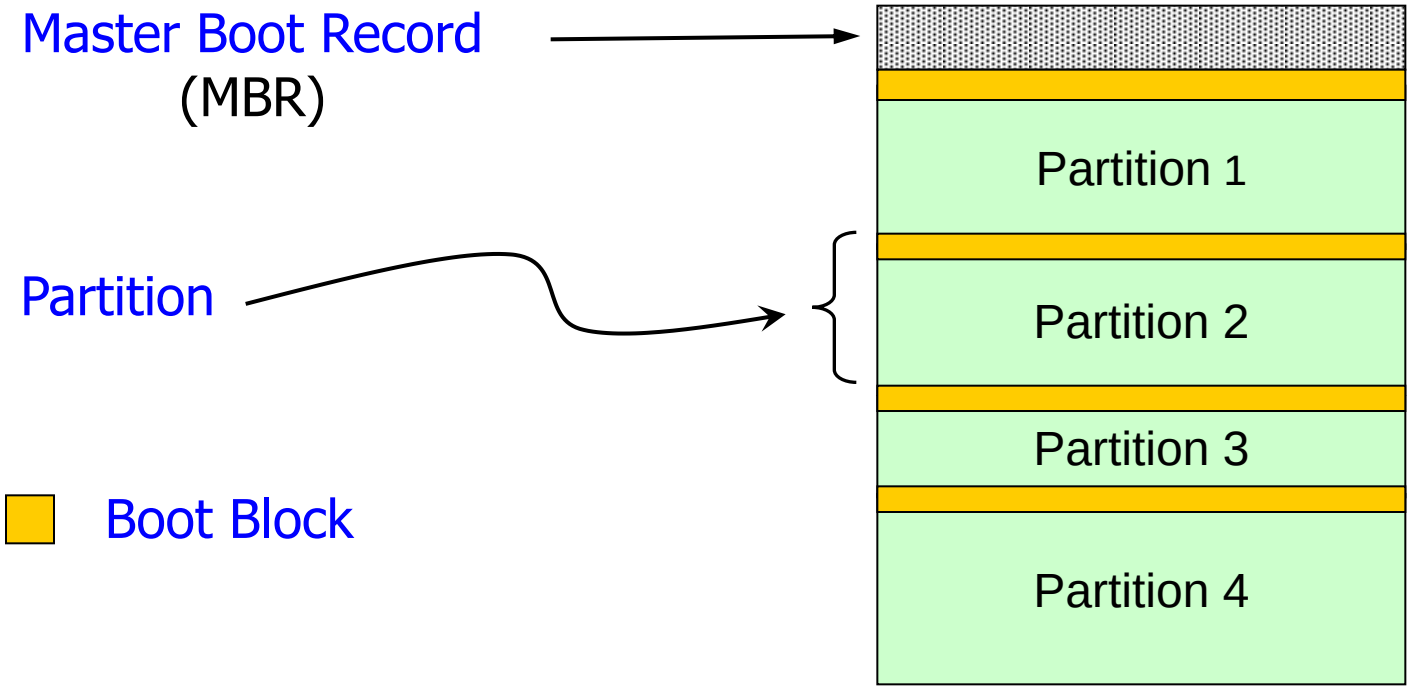






Tổ chức thông tin trên đĩa cứng

- Đĩa cứng trong hệ thống PC (lụn lý)



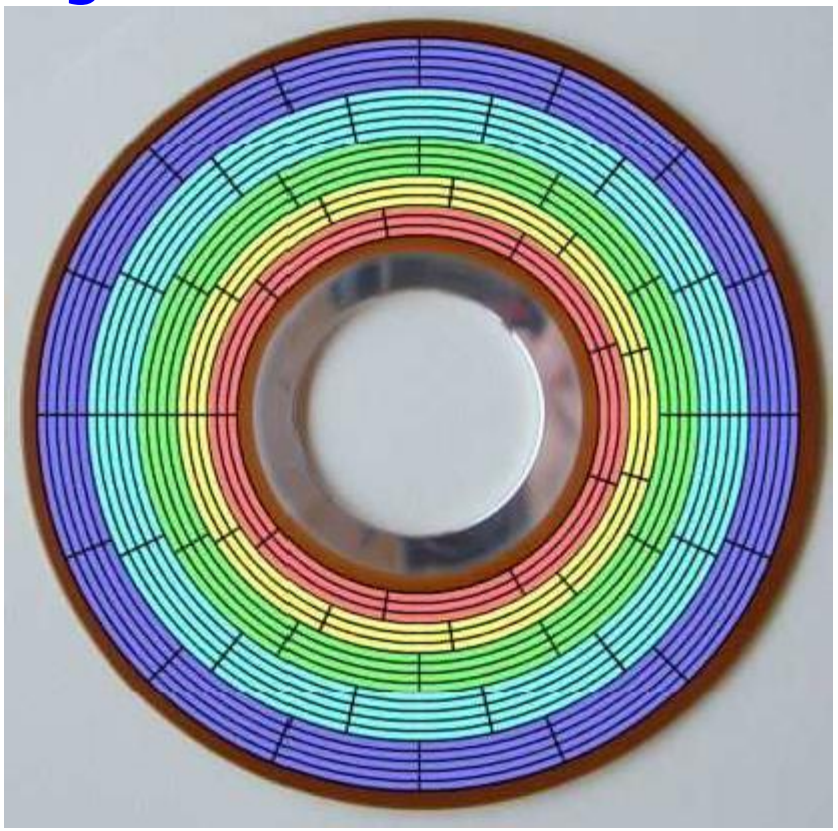
Các tham số của đĩa

- Thời gian đọc/ghi dữ liệu trên đĩa bao gồm:
  - **Seek time**: thời gian di chuyển đầu đọc để định vị đúng track/cylinder, phụ thuộc tốc độ/cách di chuyển của đầu đọc
  - **Rotational delay** (latency): 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

Loại đĩa cứng mới hiện nay

- Đĩa loại mới phân bố lại mật độ dữ liệu: lưu trữ mật độ Thông tin (bit)/vùng

  Đĩa chia ra thành vùng có số lượng sectors/vùng khác nhau (ngoài nhiều hơn trong)



Định danh đĩa (Addressing)

- OS sẽ quản lý
  - Loại giao tiếp (IDE/SCSI, etc), đĩa nào, số sector….
- Làm sao xác định tiếp sectors, tracks, etc?
  - Loại đĩa cũ: xác định bởi cylinder/head/sector (CHS)
  - Loại đĩa mới: chỉ số "block" luận lý
    - LBA = logical block address
- Chỉ số sector được sử dụng như thế nào?
  - Phần mềm quản lý hệ thống file sẽ chuyển đổi định danh block luận lý sang vật lý tương ứng trên đĩa
  - Thuật ngữ
    - Đối với người sử dụng đĩa: "khối" hay "Sector" là như nhau
    - Đối với người sử dụng hệ thống file: "khối" có dung lượng cố định, gồm 1 hay nhiều "sectors"

Định danh & Định thời đĩa

- Mục tiêu của giải thuật định thời đĩa:
  - Quản lý hàng đợi các yêu cầu truy xuất đĩa
  - Dịch vụ các yêu cầu hợp lý
    - Ví dụ: đầu từ dịch đến vị trí gần nhất
- Mục tiêu định danh luận lý đĩa
  - Che dấu phần chuyển đổi vật lý (Track?, Sector? …ở đâu trên đĩa)
- Vấn đề:
  - Các hệ điều hành cũ: Quan tâm kỹ đến sắp đặt không gian trên đĩa
  - Các hệ điều hành mới: Quan tâm đến các sectors liền kề cần được sắp xếp gần nhau
  - Thực tế: OSE rvẫn phải quan tâm đến sắp đặt không gian trên đĩa như loại cũ
  - Môn học liên quan đến các hệ điều hành cũ/thực tế

Tăng hiệu suất truy cập đĩa

Các giải pháp

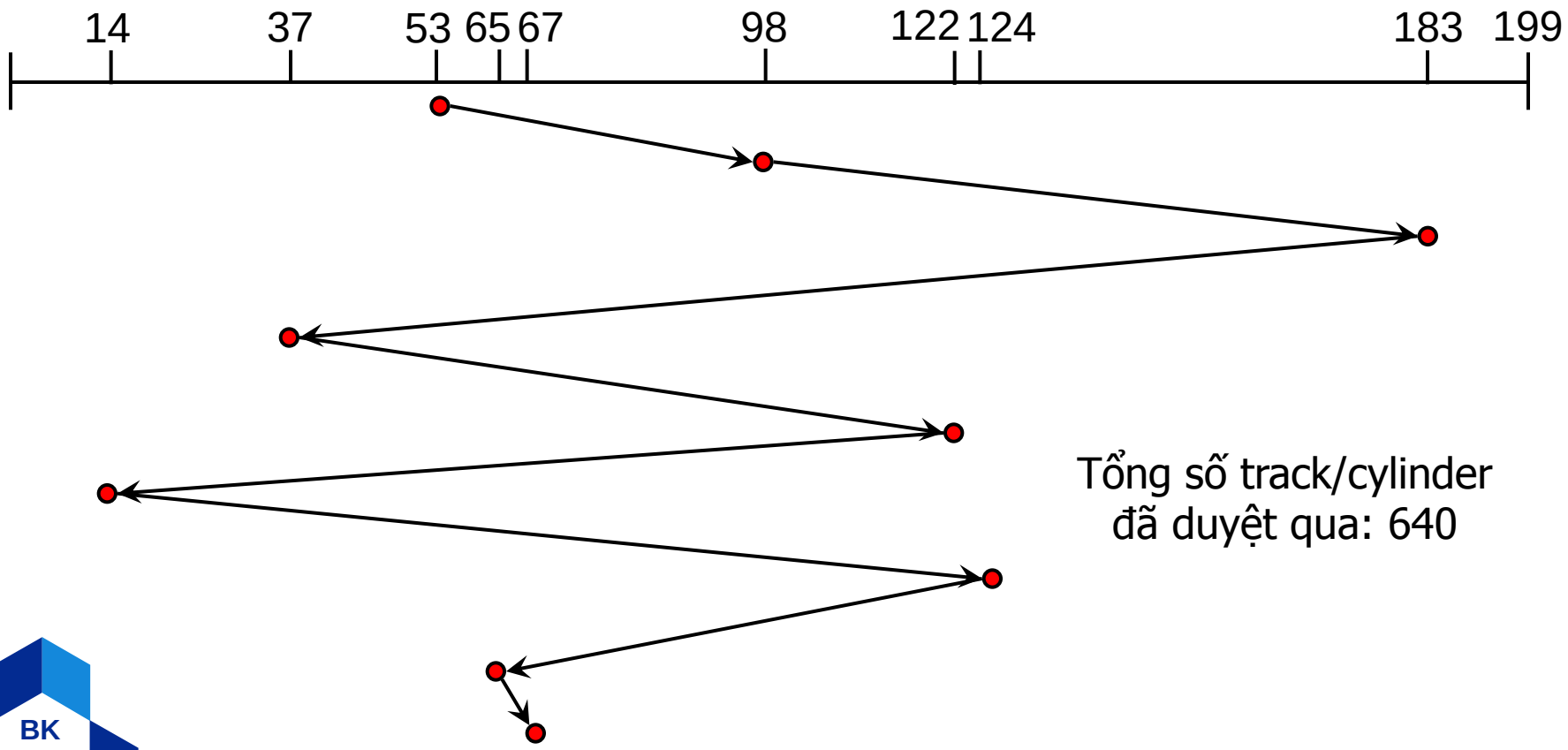
- Giảm kích thước đĩa
- Tăng tốc độ quay của đĩa
- Định thời các tác vụ truy xuất đĩa (disk scheduling) để hạn chế di chuyển đầu đọc
- Bố trí ghi dữ liệu trên đĩa hợp ly
  - các dữ liệu có liên quan nằm trên các track gần nhau
  - interleaving
- Bố trí các file thường sử dụng vào vị trí thích hợp
- Chọn kích thước của logical block
- Read ahead

Định thời truy cập đĩa

- Ý tưởng chính
  - Sắp xếp lại trật tự của các yêu cầu đọc/ghi đĩa sao cho giảm thiểu thời gian di chuyển đầu đọc (seek time)
- Các giải thuật định thời truy cập đĩa
  - First Come, First Served (FCFS)
  - Shortest-Seek-Time First (SSTF)
  - SCAN
  - C-SCAN (Circular SCAN)
  - C-LOOK
- Ví dụ: định thời chuỗi yêu cầu đọc/ghi đĩa tại
  - cylinder 98, 183, 37, 122, 14, 124, 65, 67
  - Đầu đọc đang ở cylinder số 53

First Come First Served (FCFS)

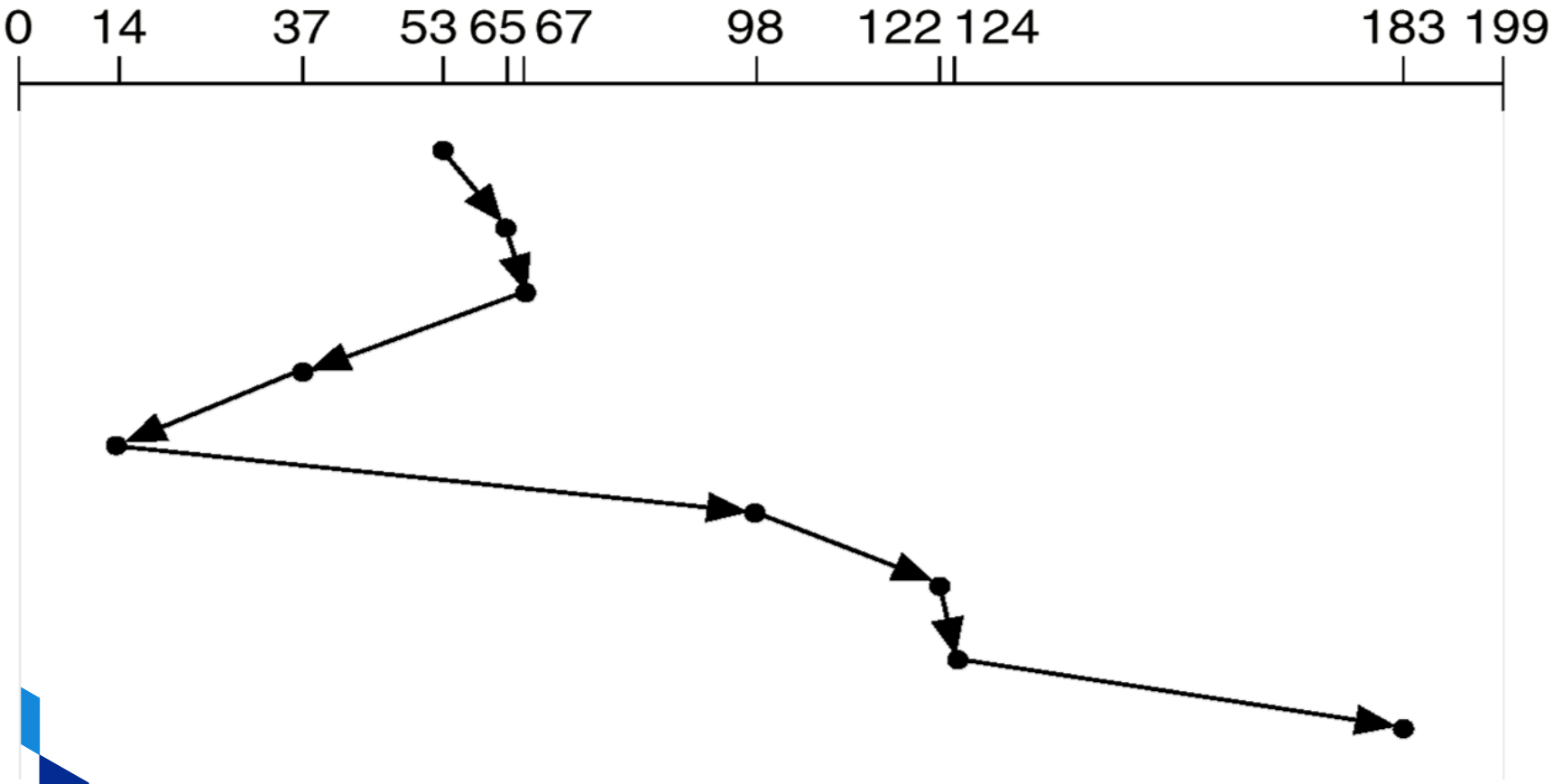
Hàng đợi: 98, 183, 37, 122, 14, 124, 65, 67
Đầu đọc đang ở cylinder số 53



Tổng số track/cylinder đã duyệt qua: 640

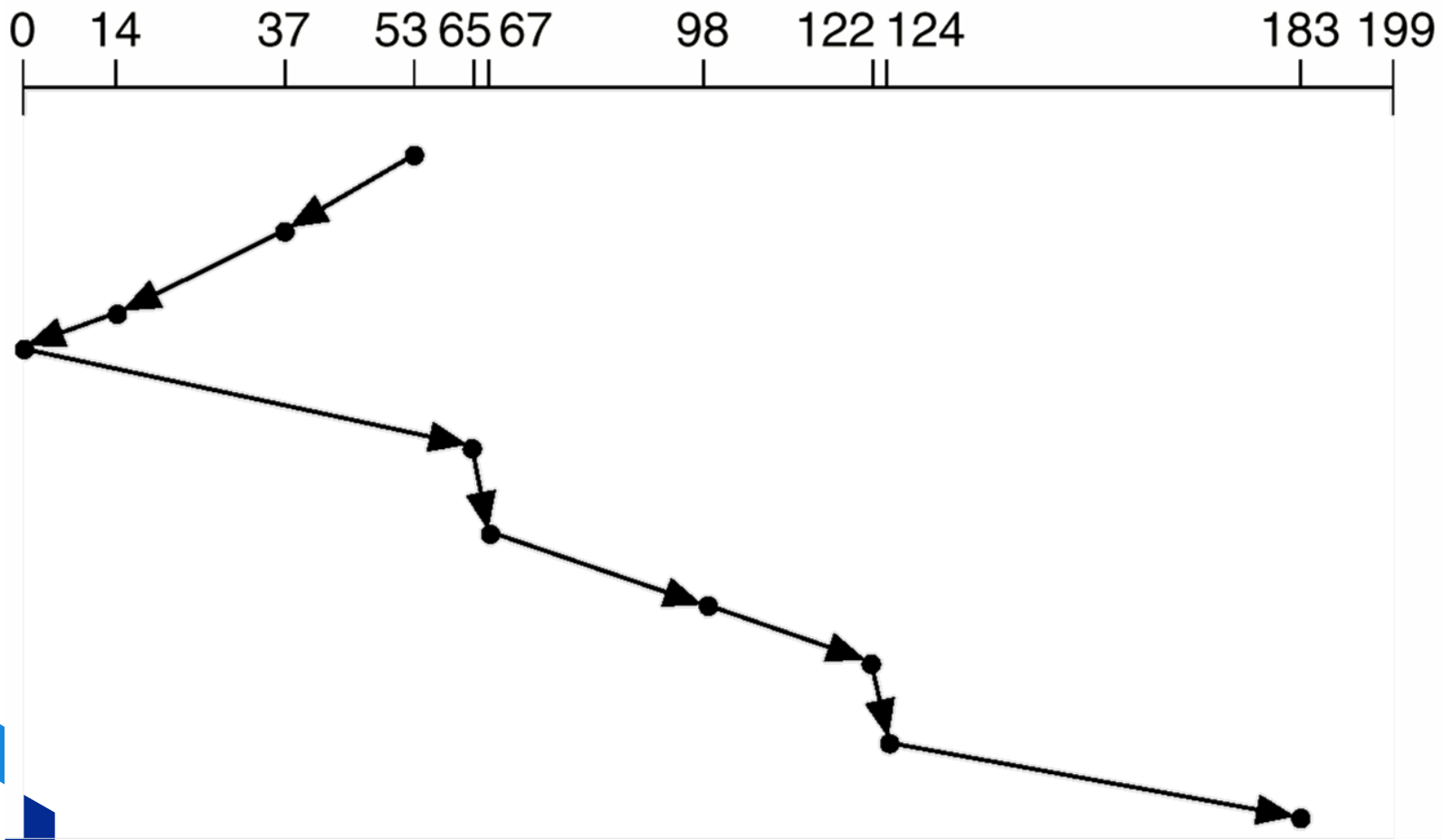
Shortest-Seek-Time First (SSTF)

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



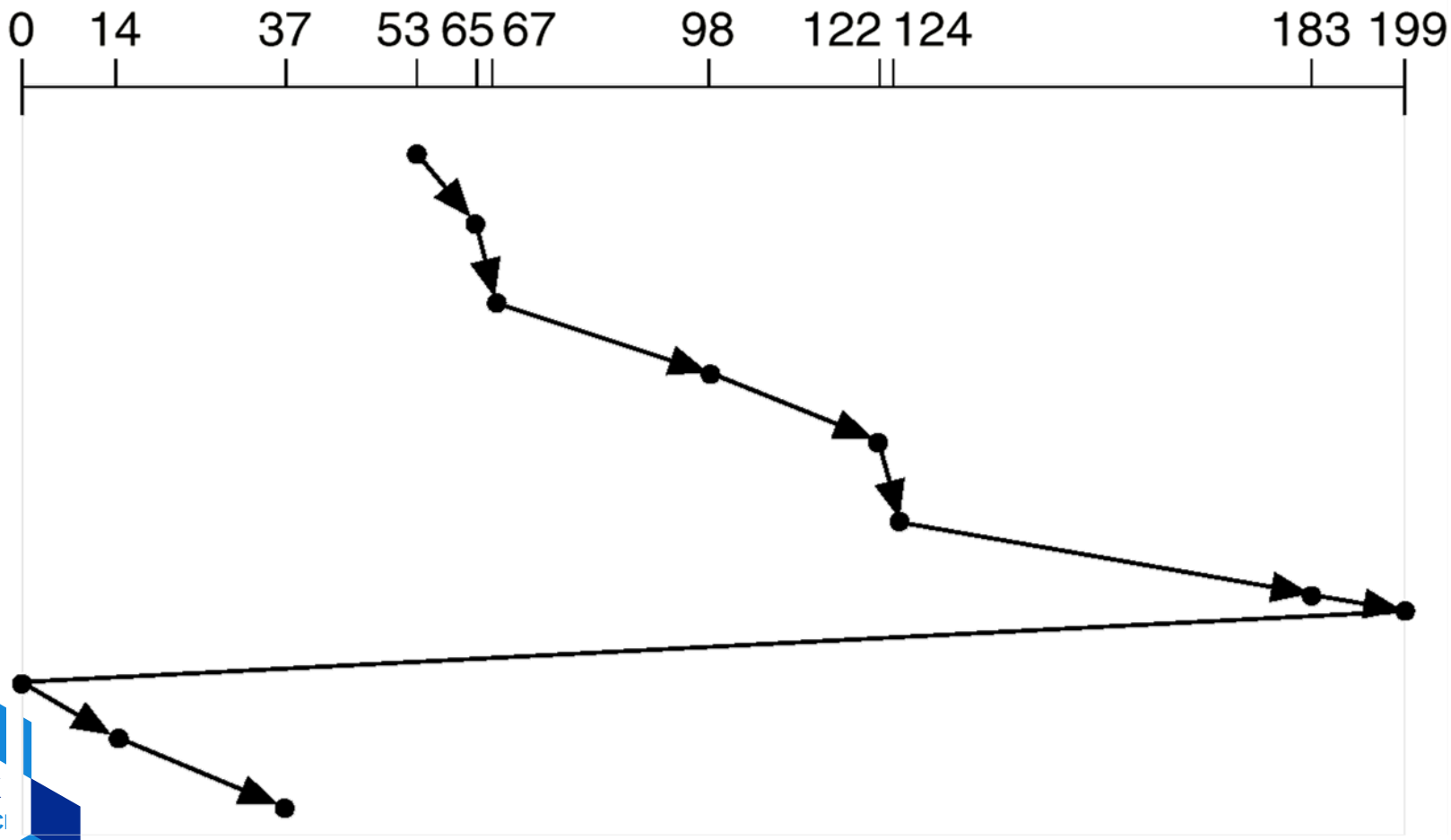
SCAN (elevator algorithm)

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



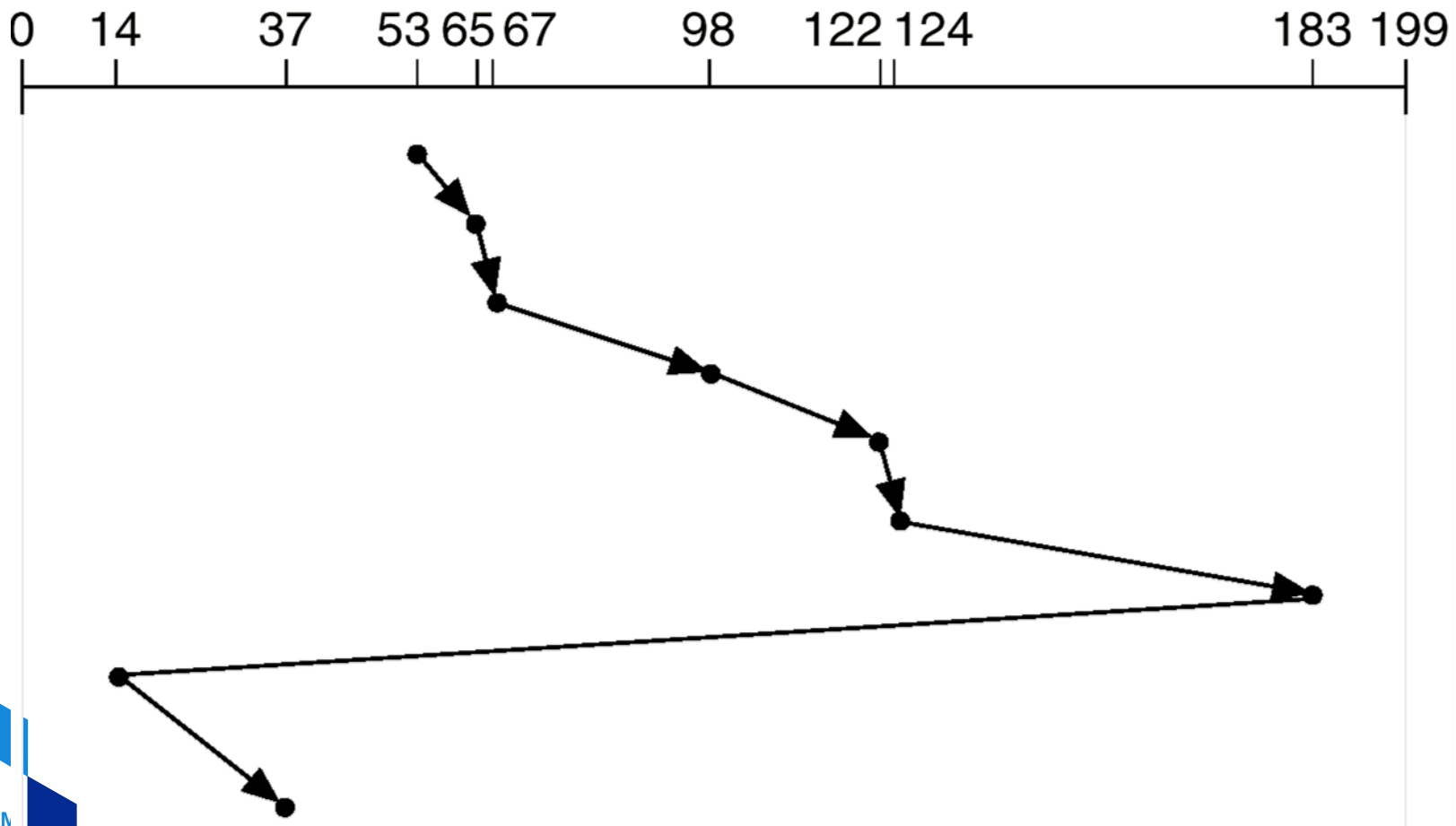
C-SCAN (Circular SCAN)

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



C-LOOK

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



Quản lý đĩa: Định dạng (formatting)

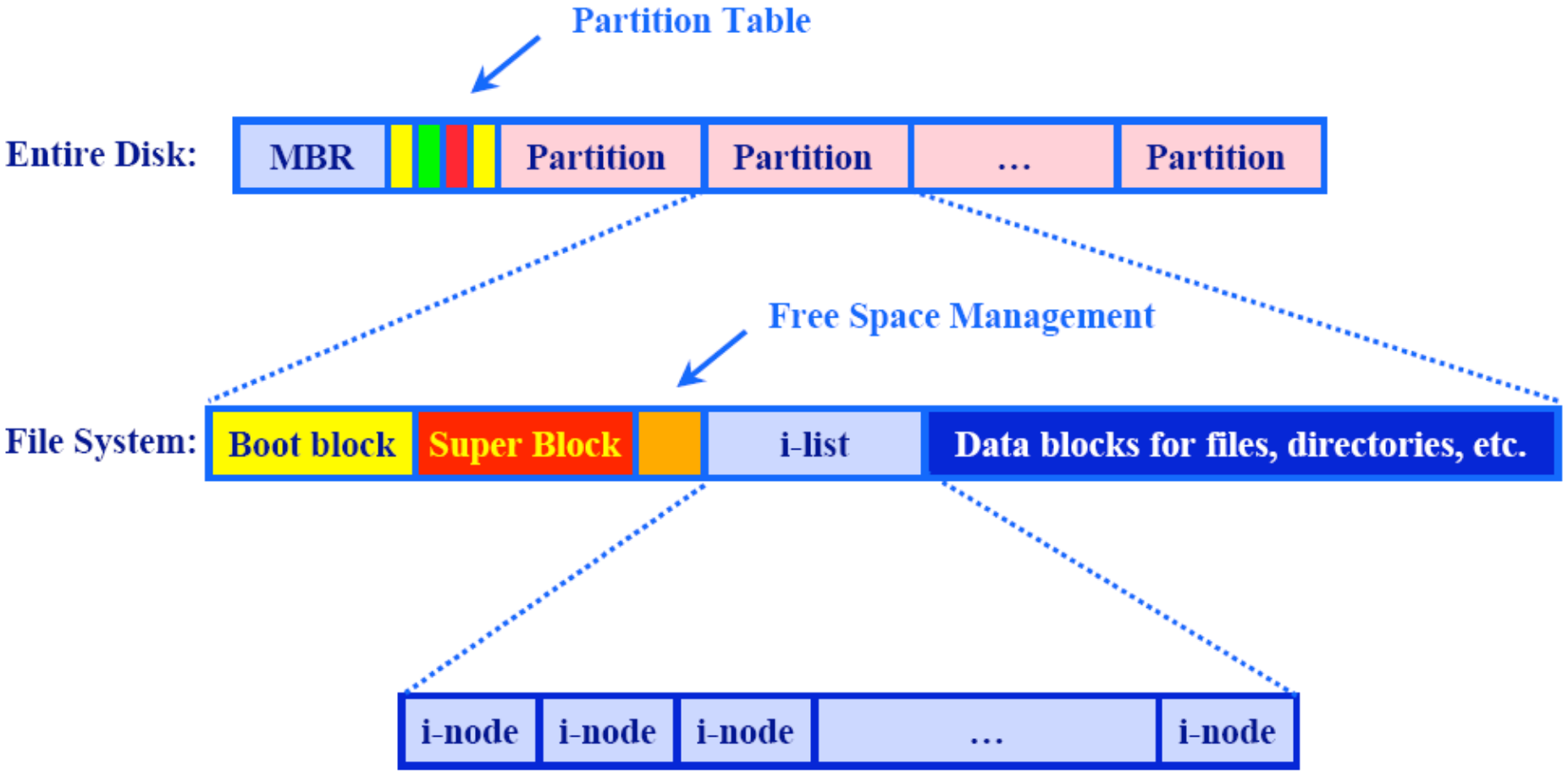
- Định dạng cấp thấp: định dạng vật lý, chia đĩa thành nhiều sector
  - Mỗi sector có cấu trúc dữ liệu đặc biệt: header – data – trailer

  - 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 gồm nhiều 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,…)
  - 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: inode table)

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 như 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

Quản lý không gian tráo đổi (swap space)

- Swap space
  - không gian đĩa được sử dụng để mở rộng không gian nhớ trong kỹ thuật bộ nhớ ảo
  - Mục tiêu quản lý: cung cấp hiệu suất cao nhất cho hệ thống quản lý bộ nhớ ảo
  - 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

Quản lý các khối bị lỗi

- Tồn tại một số khối (sectors) bị lỗi:
  - Ngay sau khi xuất xưởng: tự sửa bằng cách thay thế với các sectors, tracks dự trữ.
  - Phát hiện sau một thời gian sử dụng trong hệ thống (OS):
    - Ví dụ:
      - Block 87 (logic block) không truy xuất được
      - Điều khiển đĩa phát hiện EEC không đúng, báo Os
      - Os ghi nhận để lần sau khi reboot thông báo điều khiển đĩa thay thế
      - Sau đó vị trí block 87 đã được cập nhật lại

RAID *(Redudant Arrays of Independent Disk)*

- Khi mật độ yêu cầu truy cập đĩa cao: nghẽn, hoặc "cổ chai" → hạn chế hiệu năng và tính ổn định của hệ *thống*
- Giải pháp: kết hợp nhiều đĩa (array) truy xuất song hành:
  - Hiệu năng cải thiện: chia mảnh dữ liệu và chứa trên nhiều đĩa (data striping)
  - Reliability is improved through redundancy
- Tăng độ tin cậy: lưu trữ dư thừa thông tin (Redundant Arrays of Independent Disks, or RAID)
- Có nhiều phương pháp để đáp ứng theo tiêu chí lưu dữ thông tin (schemes or levels)

Phân mảnh dữ liệu *(Data Striping)*

- Tuy gồm nhiều đĩa, nhưng cho người sử dụng cảm giác chỉ một đĩa, nhưng dung lượng lớn
  - Khi có yêu cầu truy xuất thì sẽ tiến hành thủ tục định danh các khối vật lý chứa trên đĩa
  - Cách phân bố lưu trữ trên các đĩa như thế nào thì sẽ xác định các đĩa liên quan đến yêu cầu truy xuất
- Dữ liệu sẽ được phân mảnh đều trên các vùng lưu trữ, gọi là striping units (đơn vị phân mảnh)
  - Dung lượng mỗi đơn vị phân mảnh phụ thuộc vào mức RAID (RAID level)
- Các đơn vị phân mảnh được lưu trữ phân tán trên các đĩa theo giải thuật xoay vòng (Round Robin)

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

Phân mảnh khối – 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	…

Phân mảnh bit – 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	…

Hiệu suất phân mảnh

- Hệ thống RAID có D đĩa: tốc độ tăng tối đa là D lần
  - Vì cùng lúc D đĩa được truy xuất song hành
  - Khi đọc với khối lớn dữ liệu: không có sự khác biệt giữa phân mảnh khối và phân mảnh bit
    - Khi mà có yêu cầu đọc D blocks
  - Phân mảnh khối hiệu quả hơn khi truy cập nhiều yêu cầu truy cập không liên quan với nhau
    - Đối với phân mảnh bit, tất cả D đĩa đều phải truy xuất để có được yêu cầu 1 block của file dữ liệu
    - Trong khi với phân mảnh khối, thì mỗi đĩa có thể thỏa mãn 1 yêu cầu, vì các khối khác nhau được lưu trên các đĩa khác nhau
- Hiệu suất ghi thì như nhau, nhưng cũng bị ảnh hưởng bởi phương thức lưu chẫn/lẻ.

Độ tin cậy

- Thời gian làm việc trung bình (mean-time-to-failure = MTTF) của 1 đĩa cứng khoảng 50,000 giờ (~5.7 năm)
- Hệ thống gồm nhiều đĩa: MTTF tăng, vì số đĩa nhiều hơn

$$(1\text{-}p)^n$$

- Ngoài ra độ tin cậy cũng được cải thiện vì có lưu trữ thông tin dự trữ

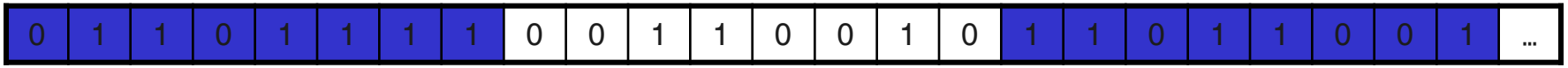
Độ dư dự trữ (Redundancy)

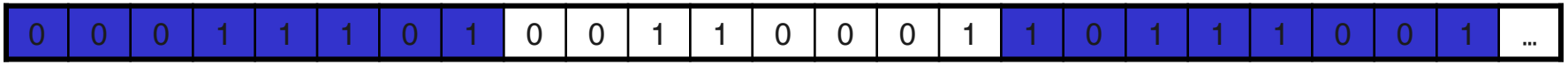
- Độ tin cậy của hệ thống nhiều đĩa sẽ được cải thiện bởi việc lưu trữ thông tin dự trữ
- Khi truy xuất bị lỗi, các thông tin dự trữ sẽ được sử dụng để khôi phục thông tin bị thất lạc
  - Dự liệu dự trữ có thể được lưu trên một đĩa riêng biệt, hoặc
  - Phân bố đều trên các đĩa
- Dữ liệu dự trữ thường được lưu trữ dưới dạng bit chẵn lẻ
  - Ngoài còn có các cách khác để đảm bảo độ tin cậy tốt hơn

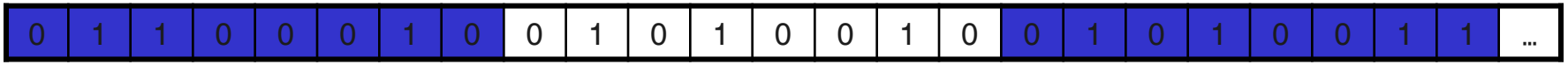
Phương thức Parity

- Mỗi bit dữ liệu liên quan đến bit chẵn/lẻ chứa trên đĩa kiểm tra
  - Nếu tổng các bit 1 của dữ liệu là 0 (chẵn) thì bit chẵn/lẻ là 0
  - Nếu tổng các bit 1 của dữ liệu là 1 (lẻ) thì bit chẵn/lẻ sẽ là 1
- Dữ liệu trên bất cứ đĩa nào bị lỗi đều có thể phục hồi từng bit một

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





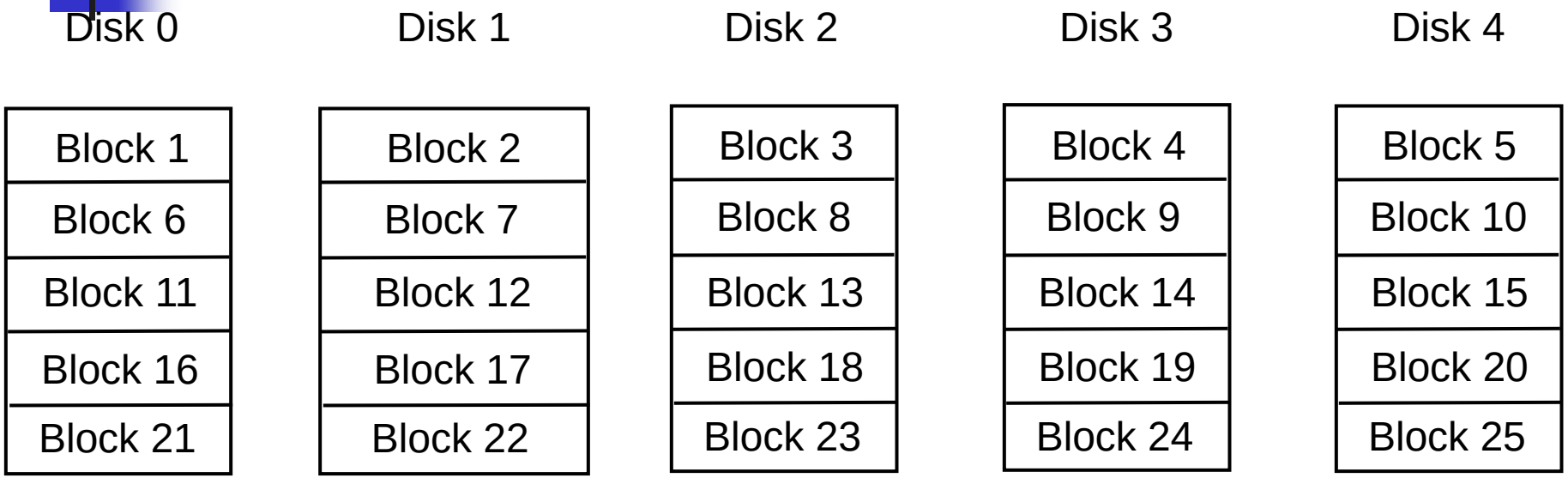


Here is a fifth CHECK DISK with the parity data

Parity Scheme and Reliability

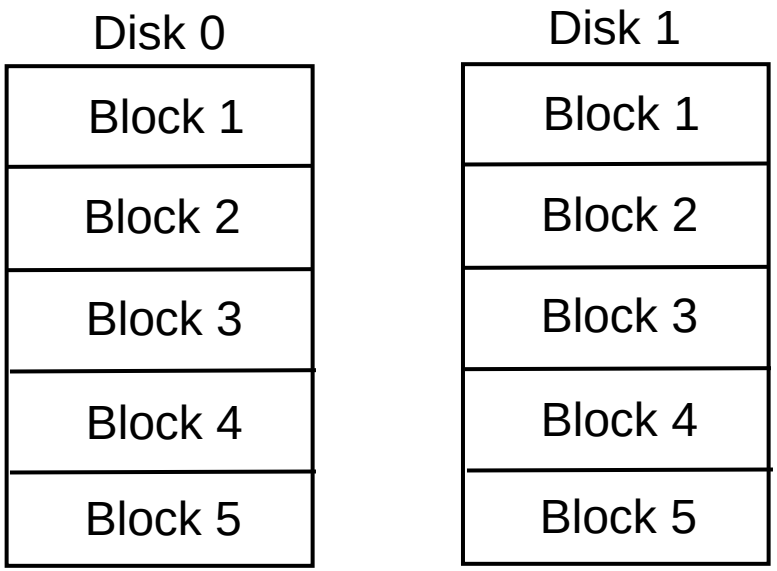
- In RAID systems the disk array is partitioned into reliability groups
  - A reliability group consists of a set of data disks and a set of check disks
  - The number of check disks depends on the reliability level that is selected
- Given a RAID system with 100 disks and an additional 10 check disks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

RAID0: 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

RAID1: Mirrored

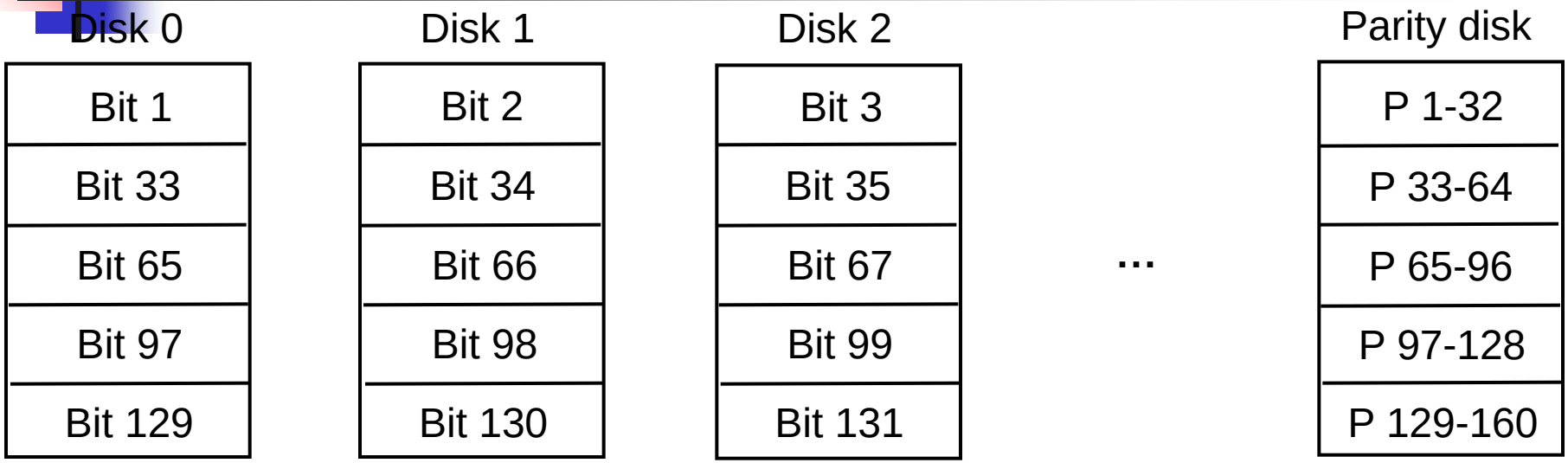


- For each disk in the system an identical copy is kept, hence the term mirroring
  - No data striping, but parallel reads of the duplicate disks can be made, otherwise read performance is similar to a single disk
- Very reliable but the most expensive RAID level
  - Poor write performance as the duplicate 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

RAID2: Memory-Style ECC

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

RAID3: Bit-Interleaved Parity



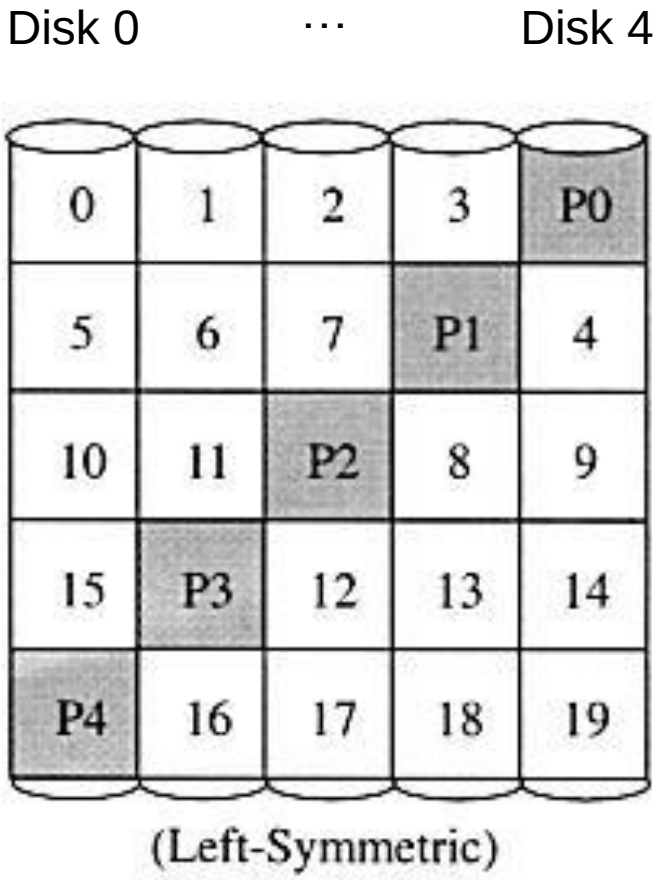
- Uses bit striping
  - Good read performance for large requests
    - Up to D times the speed of a single disk
    - Poor read performance for multiple small requests
- Uses a single check disk with parity information
  - Disk controllers can easily determine which disk has failed, so the check disks 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

RAID4: 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.