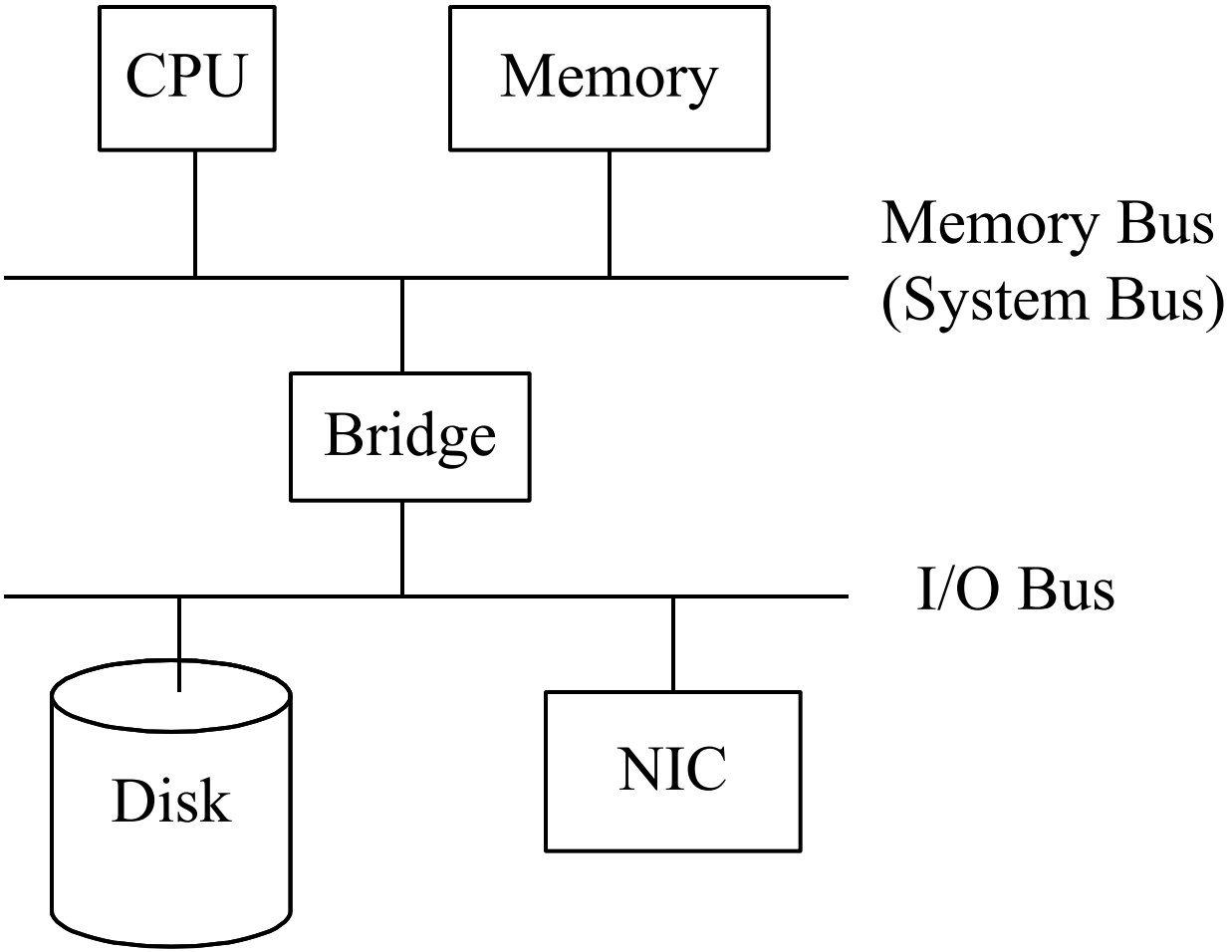
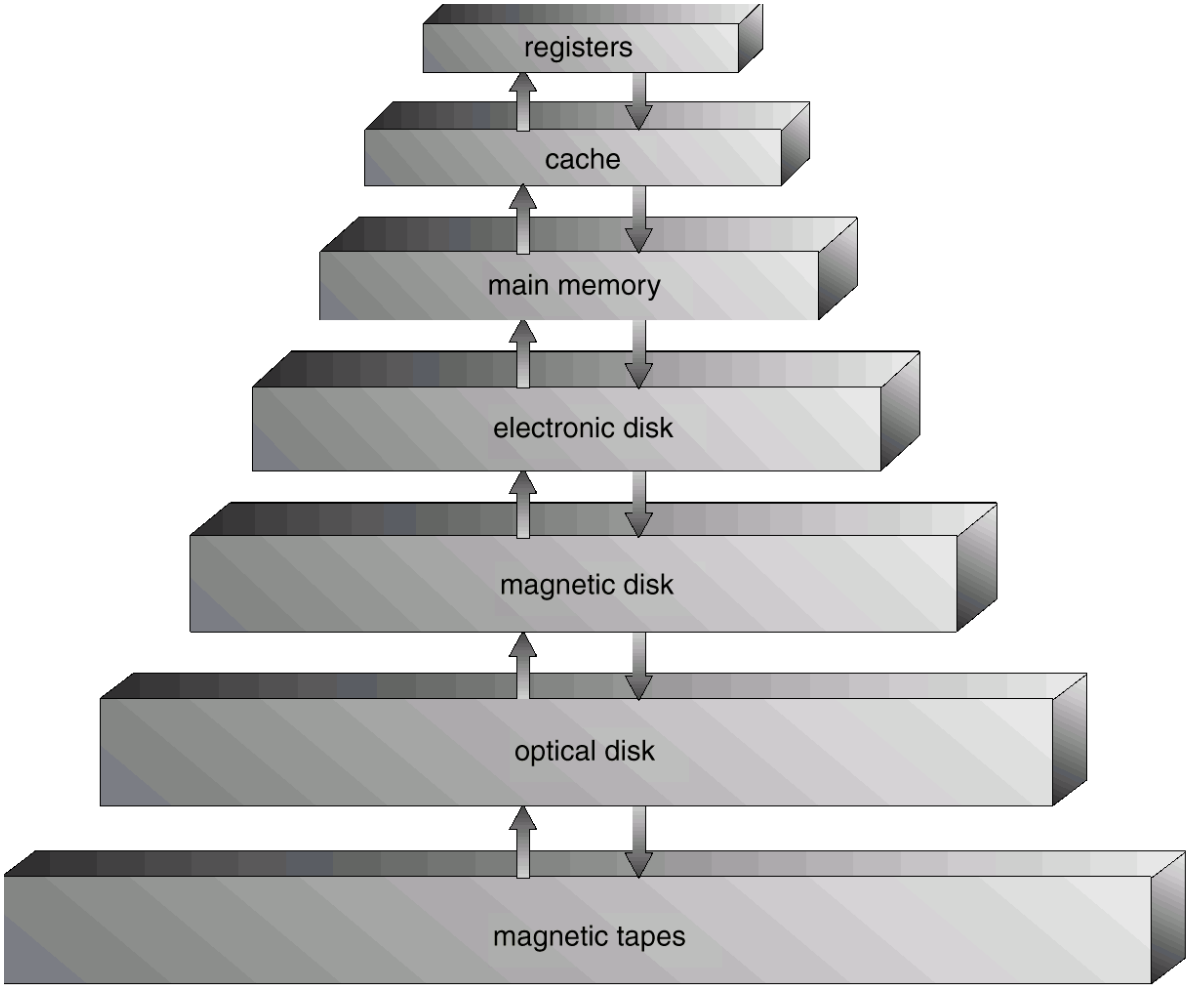
Hệ thống file: hình ảnh bộ nhớ phụ



Phân cấp thiết bị lưu trữ



Bộ nhớ phụ

Bộ nhớ phụ:

Là bộ lưu trữ bên ngoài bộ nhớ chính

Không cho phép thực thi trực tiếp các instruction hoặc lưu trữ dữ liệu cho các instruction này.

Đặc điểm:

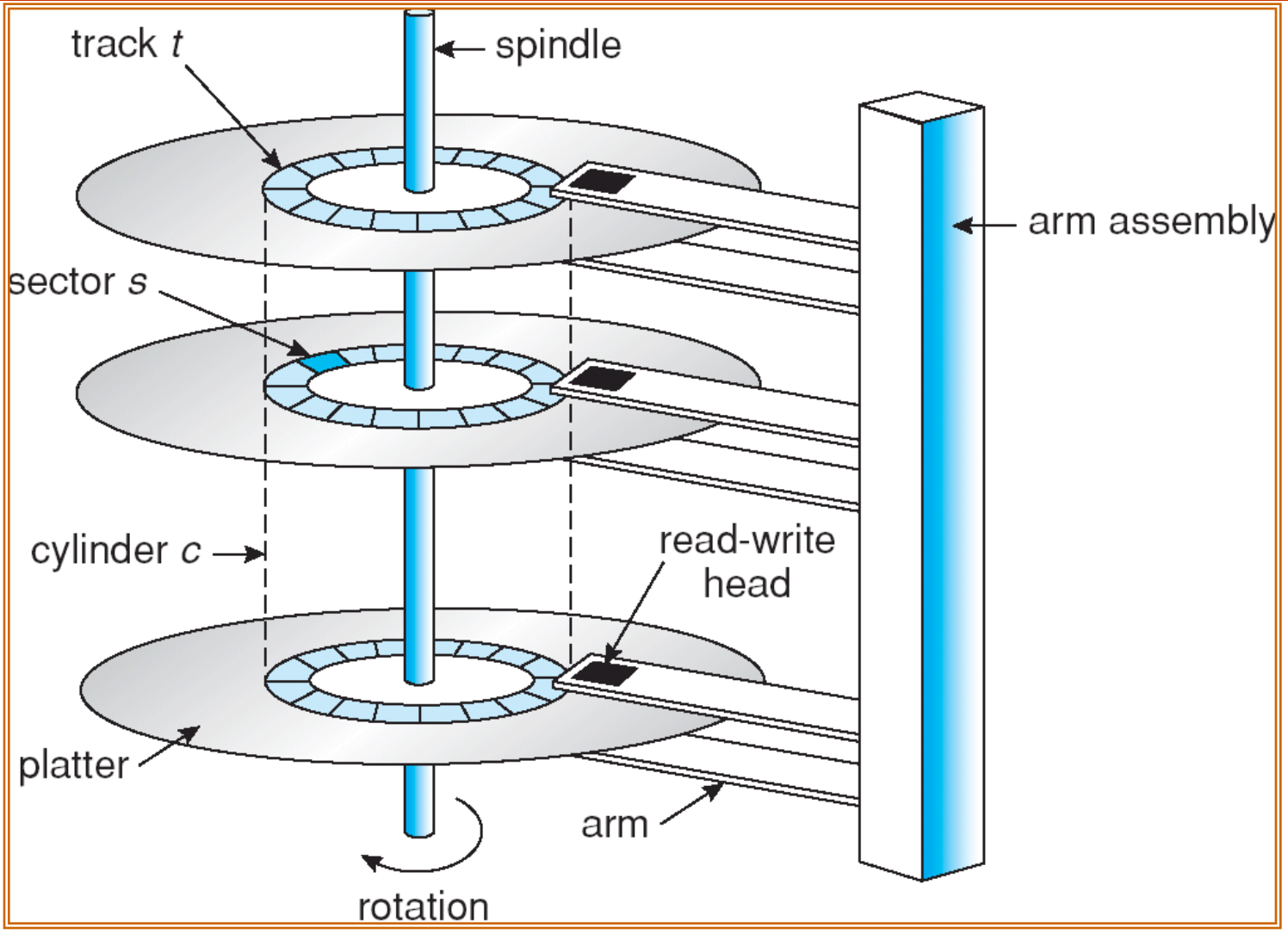
Kích thước lơn: ~ GB, TB

Rẻ

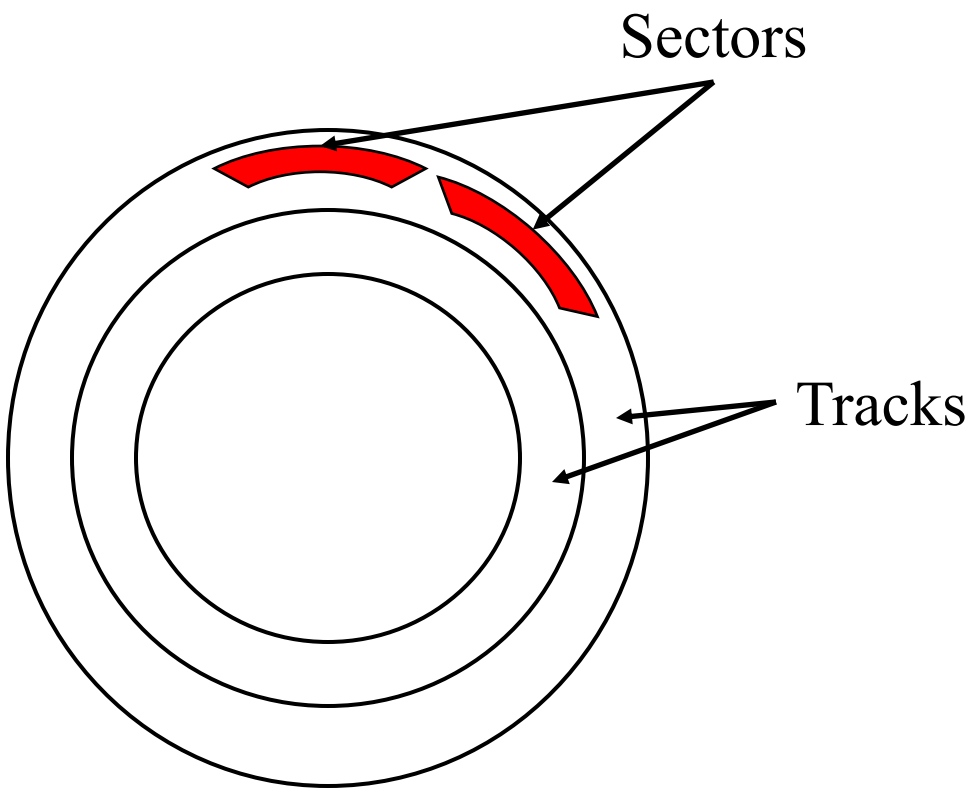
Dữ liệu được lưu trữ ngay cả khi không có nguồn cung cấp năng lượng

Chậm: thời gian truy cập khoảng milliseconds

Ổ đĩa


track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation

Ổ đĩa



Seek time: thời gian di chuyển đầu đọc tới track cần đọc

Rotational delay: thời gian để tìm sector cần đọc khi mà đầu đọc đã tìm được track

Transfer rate: tốc độ đọc/ghi dữ liệu

Thông thường:

Seek: ~8-10ms

Rotational delay: ~4.15ms với tốc độ 7200 rpm

Lập lịch ổ đĩa

Ổ đĩa chậm hơn rất nhiều so với bộ nhớ chính

Tốc độ I/O chiếm vị trí quan trọng trong việc nâng cao khả năng thực thi của máy tính

Thời gian truy cập (seek time+ rotational delay) >> thời gian copy một sector

Vì vậy thứ tự đọc các sector là ảnh hưởng nhiều đến thời gian I/O

Lập lịch ổ đĩa

Thường là dựa trên vị trí của các sector hơn là độ ưu tiên của các tiến trình

Thay đổi thứ tự đọc/ghi các sector giúp tăng tốc độ thực thi

Lập lịch ổ đĩa (tt.)

Giới thiệu một số thuật toán lập lịch cho thứ tự nhập/xuất.

Giả sử chúng ta có một hàng đợi trong khoảng (0-199).

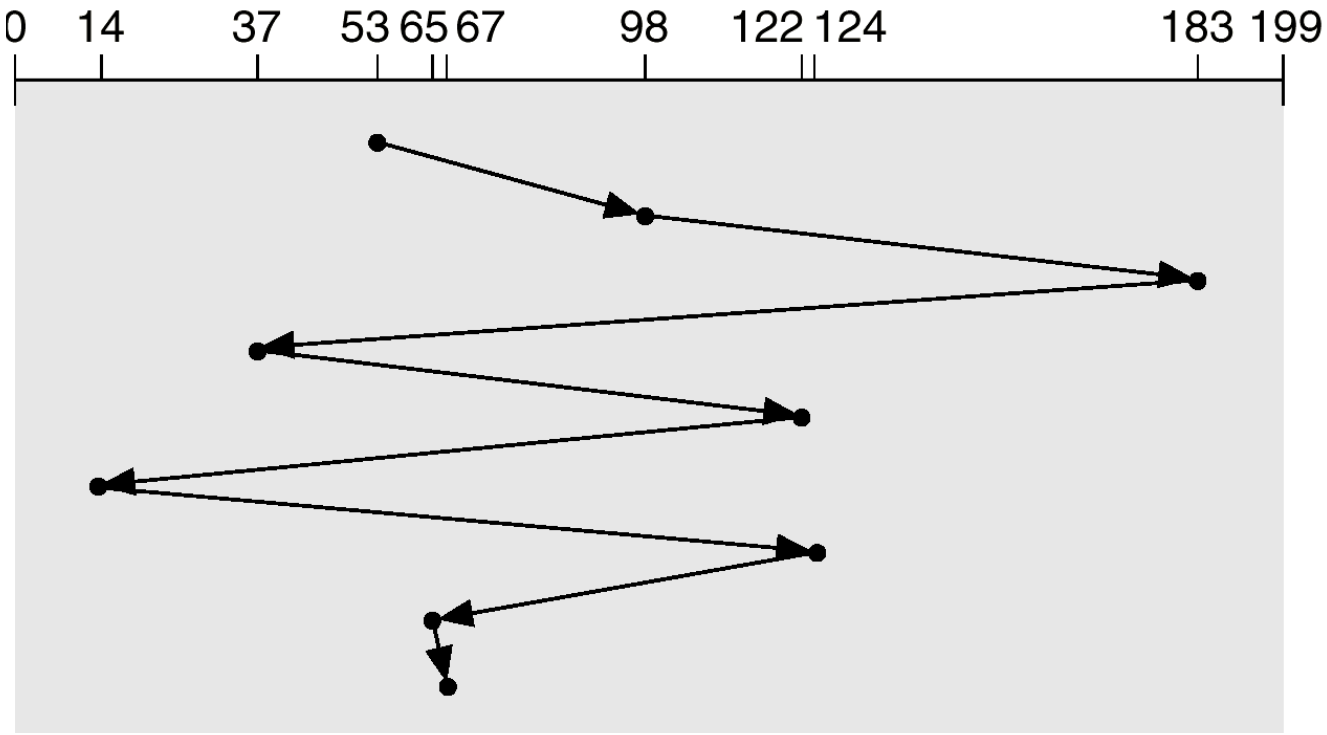
98, 183, 37, 122, 14, 124, 65, 67

Đầu đọc ở vị trí 53

FCFS

Tổng di chuyển của đầu đọc 640 cylinders.

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



SSTF (shortest seek-time first)

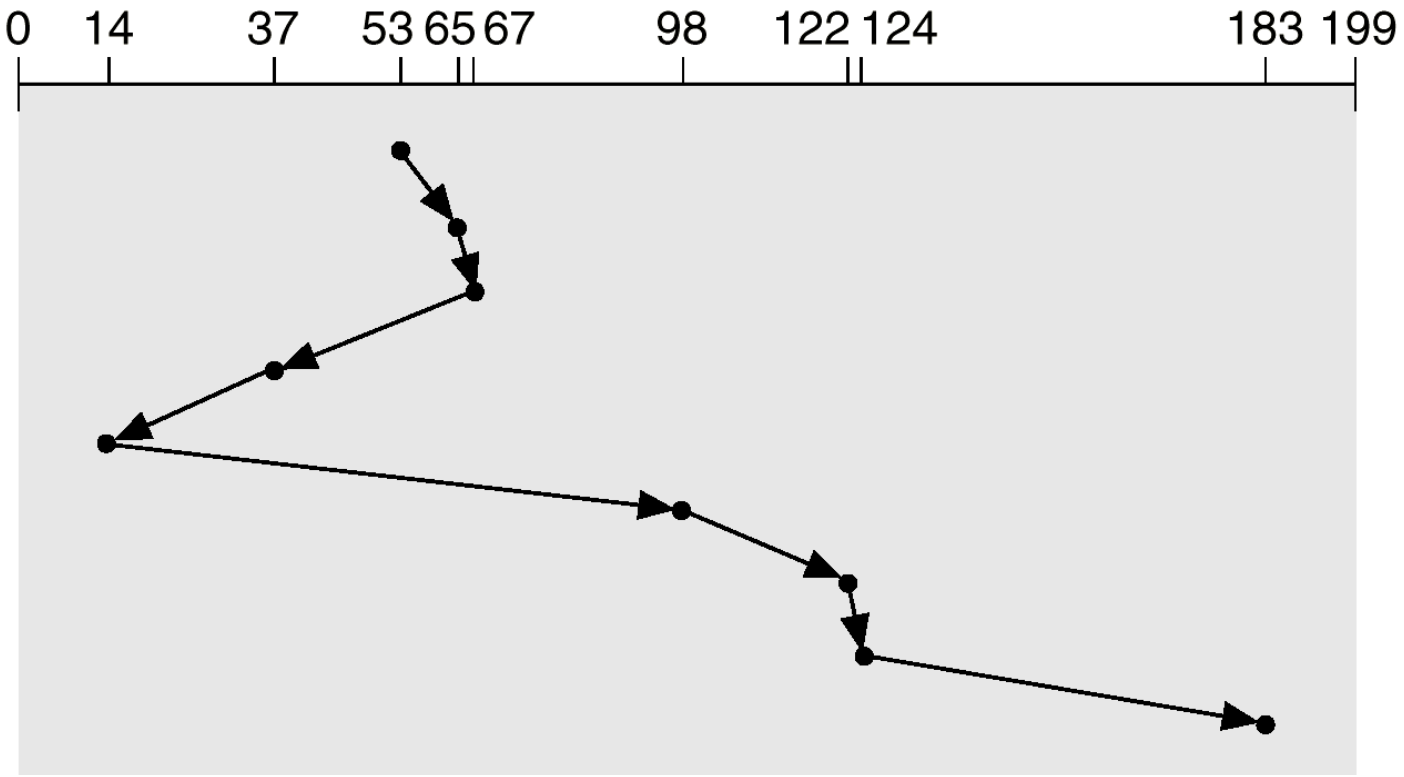
Chọn sector có seek time nhỏ nhất từ vị trí đầu đọc hiện tại.

Lập lịch SSTF là một dạng của lập lịch SJF; có thể gây starvation cho một số yêu cầu.

Tổng di chuyển đầu đọc của thiết bị là 236 cylinders.

SSTF (tt.)

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



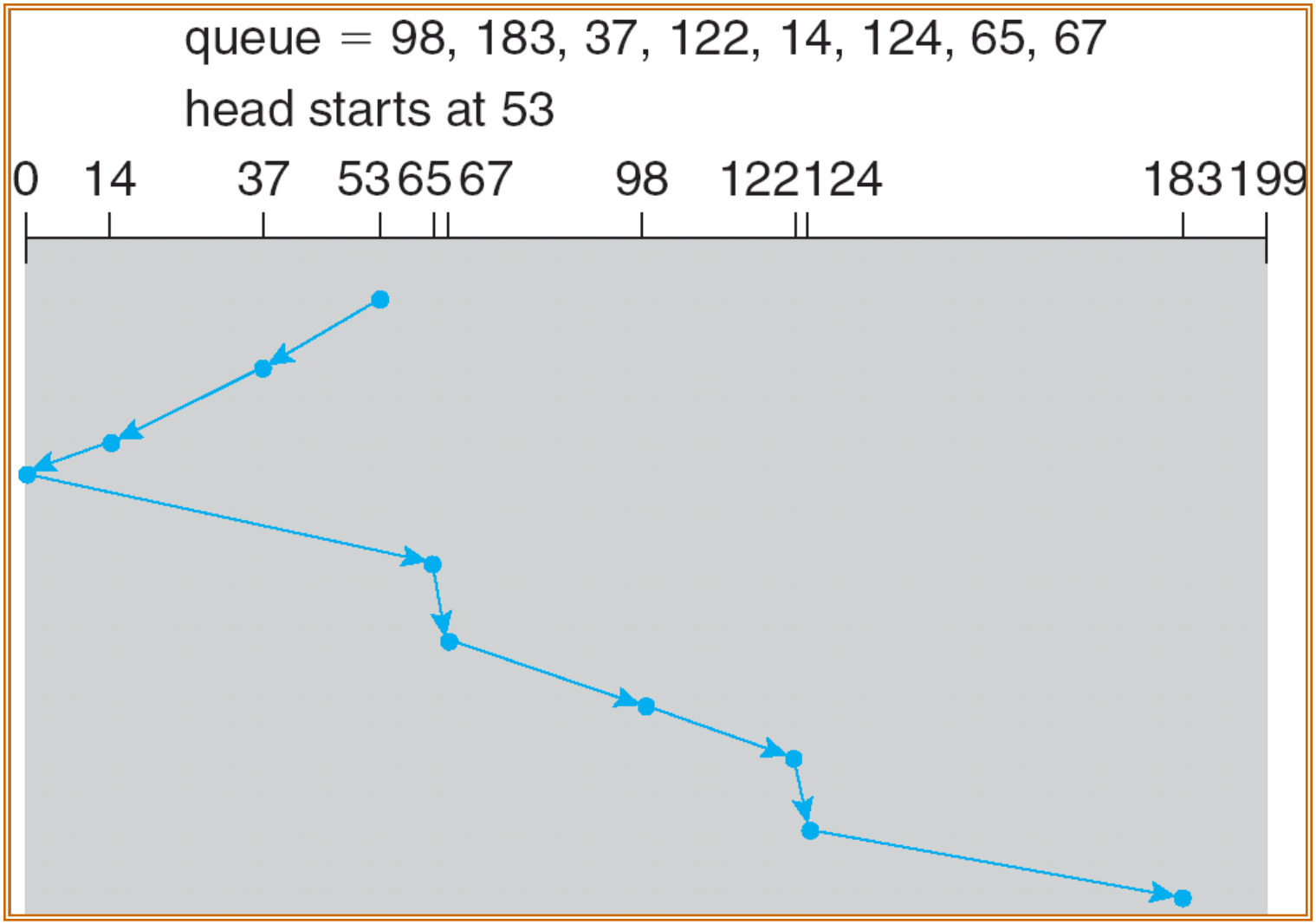
SCAN

Đầu đọc bắt đầu tại một đầu của ổ đĩa, và di chuyển tới đầu còn lại, phục vụ các yêu cầu trong lúc di chuyển, khi tới đầu kia thì quay ngược lại và tiếp tục phục vụ các yêu cầu.

Ta có thể gọi là *thuật toán thang máy*.

Tổng di chuyển đầu đọc là 208 cylinders.

SCAN (tt.)


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

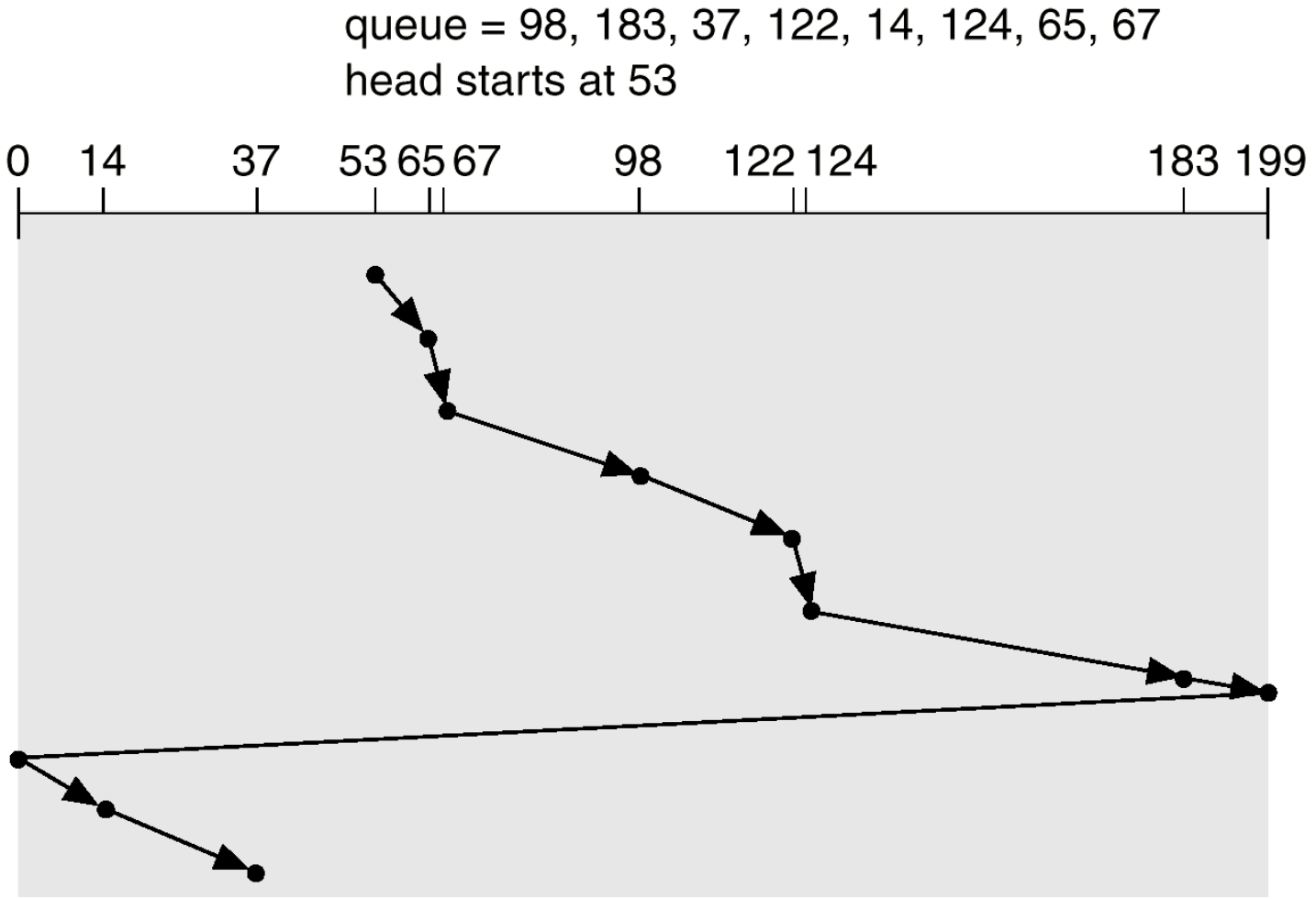
C-SCAN

Cung cấp thời gian đợi trung bình đều hơn so với SCAN.

Đầu đọc bắt đầu tại một đầu của ổ đĩa, và di chuyển tới đầu còn lại, phục vụ các yêu cầu trong lúc di chuyển, khi tới đầu kia thì quay ngược lại vị trí bắt đầu của ổ đĩa, không phục vụ các yêu cầu trong lúc di chuyển ngược lại.

Xem các cylinders là một danh sách vòng, nối cylinder cuối với cylinder đầu tiên.

C-SCAN (tt.)

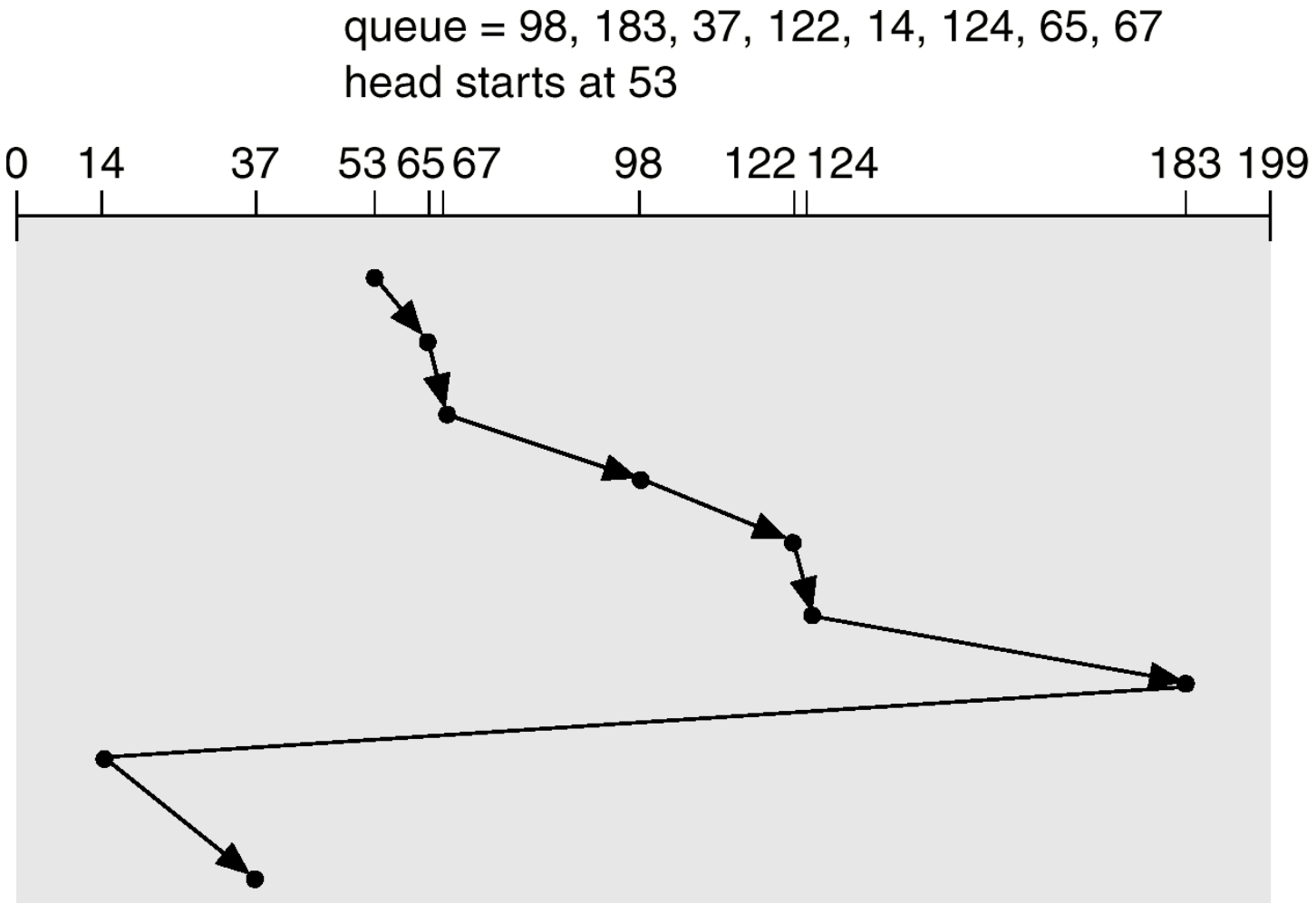

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK

Một phiên bản của C-SCAN

Đầu đọc chỉ đi chuyển đến yêu cầu ở xa nhất về một phía, sau đó chuyển hướng, không cần phải di chuyển về đến đầu cuối của ở đĩa.

C-LOOK (tt.)


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

Chọn lựa thuật toán lập lịch ở đĩa

SSTF có triết lý tự nhiên

SCAN và C-SCAN tốt cho các hệ thống thường truy suất khối lượng dữ liệu lớn trên đĩa.

Tốc độ thực thi còn tùy thuộc vào thứ tự và loại yêu cầu.

Các yêu cầu từ ổ đĩa chịu ảnh hưởng từ các phương thức cấp phát bộ nhớ để lưu trữ file.

Thuật toán lập lịch đĩa nên cài đặt riêng biệt với HĐH, cho phép HĐH có thể thay thế thuật toán khi cần thiết.

Nên chọn thuật toán SSTF hoặc LOOK làm thuật toán mặc định.

Quản lý ổ đĩa

Formatting cấp thấp, hay là *Format vật lý* — chia ổ đĩa thành các sector mà disk controller có thể đọc và ghi.

Để có thể lưu được các tập tin, HĐH cần phải lưu cấu trúc dữ liệu của nó lên ổ đĩa.

- *Partition* ổ đĩa thành một hoặc nhiều nhóm cylinders.
- *Logical formatting* hay "tạo hệ thống tập tin".

Boot block dùng để khởi động hệ thống.

- Bootstrap được lưu trên ROM.
- Bootstrap nạp *Bootstrap loader* program.

Các phương thức như là *sector dự trữ* dùng để xử lý các blocks hư.

Quản lý Swap-Space

Swap-space — sử dụng không gian trên ở đĩa như là phần mở rộng của bộ nhớ chính.

Swap-space có thể được lưu trên partition chứa file thông thường, cũng có thể trên một partition riêng biệt.

Quản lý swap-space

- 4.3BSD tạo swap space khi tiến trình bắt đầu; lưu *text segment* (mã chương trình) và *data segment*.
- Kernel dùng *swap maps* để lưu vết sử dụng swap-space.
- Solaris 2 tạo swap space chỉ khi trang bị chép ra khỏi bộ nhớ chính, (không phải lúc các trang được tạo.)

Bài toán về độ tin cậy khi lưu trên đĩa

Một vài cải tiến kĩ thuật để giúp nhiều ở đĩa có thể phối hợp sử dụng cùng nhau.

RAID là một kĩ thuật quan trọng đang được sử dụng.

RAID

Redundant Array of Inexpensive Disks (RAID)

HĐH sử dụng một bộ các ổ đĩa như là một ổ đĩa logic

Thay thế ổ đĩa dung lượng lớn bằng nhiều ổ đĩa dung lượng nhỏ để tăng tốc độ I/O

Dữ liệu được phân phối trên các ổ đĩa sao cho có thể truy xuất dữ liệu đồng thời từ nhiều ổ đĩa

Hỗ trợ sao lưu để tăng khả năng đối phó việc mất mát dữ liệu, khi xác suất bị lỗi sẽ tăng khi chúng ta sử dụng nhiều ổ đĩa

Tăng tính sẵn sàng, vì khó có xác suất hệ thống hỏng nhiều đĩa

Sáu cấp độ RAID đại diện cho các thiết kế khác nhau

RAID Level 0

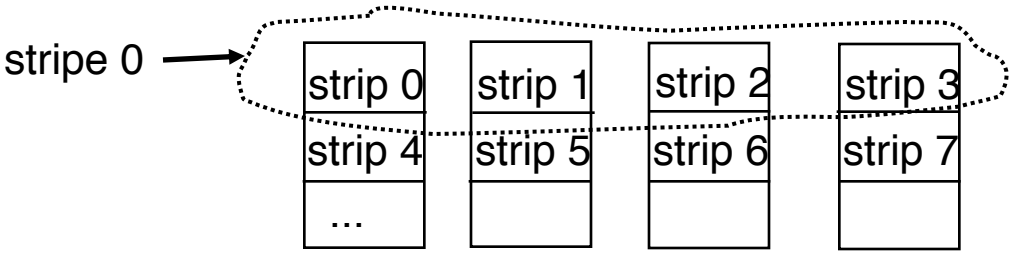
Không có sao lưu

Dữ liệu được lưu trữ dàn trãi trên các đĩa

- Tổng dung lượng của các đĩa được chia thành các strip
- Strips được đánh số xoay vòng trên các đĩa (round-robin)
- Một tập các strip liên tiếp nhau mà mỗi strip nằm trên một đĩa được gọi là một stripe (một lớp)

Tại sao tăng băng thông I/O?

Thứ tự truy cập nào mang lại hiệu quả tốt nhất?



RAID Level 1

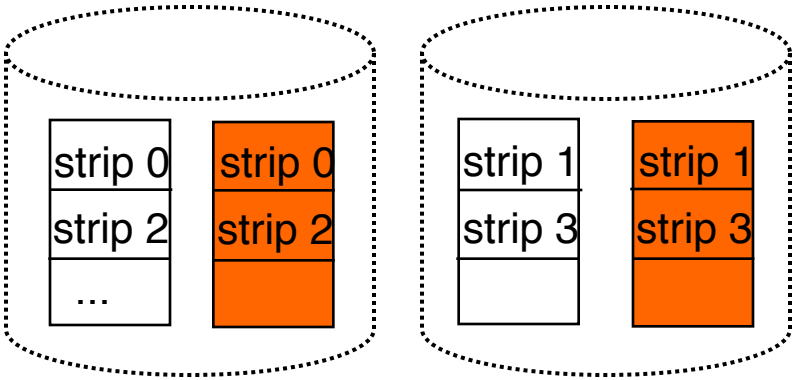
Sao lưu bằng cách nhân bản dữ liệu

Mỗi đĩa có một đĩa sao lưu, lưu y hệt dữ liệu của nó

Có thể đọc từ bất cứ đĩa nào trong 2 đĩa này (tăng tốc độ đọc nếu yêu cầu đọc nhiều)

Phải ghi trên cả 2 đĩa, tuy nhiên có thể ghi song song

Khôi phục đơn giản, nhưng chi phí cao



RAID Levels 2 và 3

Truy cập song song: tất cả các đĩa liên quan đến 1 yêu cầu I/O

Kích thước của mỗi lần read/write = số đĩa * strip size

RAID 2: error correcting code được tính dựa trên các bit tương ứng trên các ổ đĩa dữ liệu và lưu trên parity disks

Hamming code: có thể sửa lỗi 1 bit và nhận dạng 2 bit bị lỗi

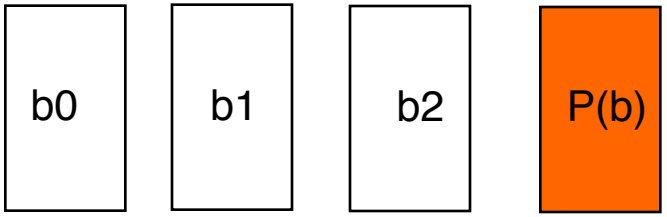
Tốn ít chi phí hơn RAID 1 nhưng chi phí vẫn cao – không áp dụng trong thực tế

RAID 3: chỉ dùng một đĩa lưu parity bits

$P(i) = X2(i) \oplus X1(i) \oplus X0(i)$

Khi một đĩa bị hư, vẫn khôi phục được dữ liệu

Chỉ cho một đĩa hư tại một thời điểm



$X2(i) = P(i) \oplus X1(i) \oplus X0(i)$

RAID Levels 4 and 5

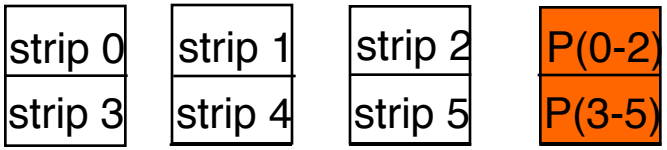
RAID 4

Dùng parity strip tương tự như RAID 3 (nhưng RAID 3 dùng parity bit)

Truy cập độc lập – vì vậy nhiều I/O có thể thực hiện song song

Truy cập độc lập (riêng lẻ) ➜ write = 2 reads + 2 writes

Parity disk có thể bị vấn đề cổ chai (bottleneck)



RAID 5

Giống RAID 4 nhưng parity strips được lưu phân phối trên các đĩa