ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA

NGUYÊN LÝ HỆ ĐIỀU HÀNH

Khoa Công nghệ thông tin
ThS. Nguyễn Thị Lệ Quyên

Nội dung môn học

- Giới thiệu
- Tiến trình và luồng
- ***Thi giữa kỳ***
- Quản lý bộ nhớ
- **Vào/ Ra**
- Hệ thống file
- Thực hành

STORAGE MANAGEMENT

1. Cấu trúc lưu trữ lớn (Mass-storage Structure)
2. Hệ thống Vào/ Ra (I/O system)

Nội dung

- **Cấu trúc lưu trữ lớn (Mass-storage Structure)**
  - Tổng quan
  - HDD Scheduling
  - NVM Scheduling
  - Phát hiện và sửa lỗi
  - Quản lý thiết bị lưu trữ
  - Quản lý không gian hoán đổi (swap-space management)
  - Storage Attachment
  - Storage-Area Networks and Storage Arrays
  - Summary
- Hệ thống Vào/ Ra (I/O system)

Tổng quan

- Ổ cứng HDD (hard disk drives), NVM (nonvolatile memory), băng từ, đĩa quang, lưu trữ đám mây, ...
- Cần nắm:
  - Mô tả cấu trúc vật lý của các thiết bị lưu trữ thứ cấp (secondary storage devices) và hiệu quả của các cấu trúc này
  - Giải thích các đặc tính hiệu suất của các thiết bị lưu trữ dung lượng lớn
  - Đánh giá các thuật toán lập lịch I/O
  - Thảo luận về các dịch vụ được cung cấp cho bộ nhớ lưu trữ dung lượng lớn, bao gồm RAID

Tổng quan

- Nội dung
  - HDD
  - NVM
  - Volatile Memory
  - Secondary Storage Connection Methods
  - Address Mapping

HDD

Figure 11.2 A 3.5-inch HDD with cover removed.

HDD

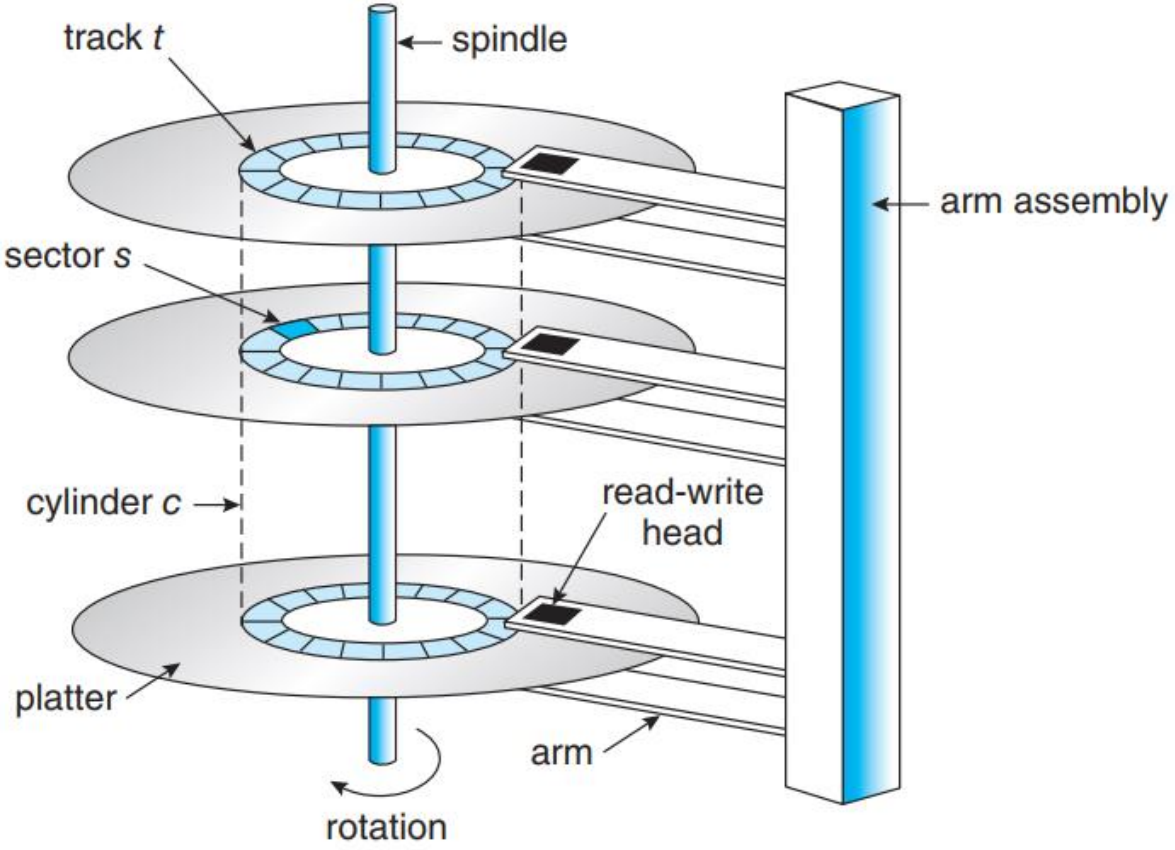


Figure 11.1 HDD moving-head disk mechanism.

HDD

- Động cơ của HDD quay với tốc độ cao. Hầu hết ổ đĩa quay 60 đến 250 lần mỗi giây, xác định bằng số vòng quay mỗi phút (RPM – round per minute)
  - Ổ đĩa thông thường quay ở tốc độ 5.400, 7.200, 10.000 và 15.000 RPM.
- Tốc độ quay liên quan đến tốc độ truyền (transfer rate – tốc độ truyền dữ liệu giữa ổ đĩa và máy tính)
- Thời gian truy cập ngẫu nhiên bao gồm 2 phần:
  - Thời gian cần thiết để di chuyển disk arm đến cylinder mong muốn, được gọi là thời gian tìm kiếm (seek time)
  - Thời gian cần thiết để sector mong muốn quay vào disk head, gọi là độ trễ quay (rotational latency)

HDD

- Các đĩa thông thường có thể truyền hàng chục đến hàng trăm MB dữ liệu mỗi giây và chúng có seek time và rotation latency vài ms.
- Disk head quay trên 1 lớp đệm khí cực mỏng và có nguy cơ tiếp xúc với bề mặt đĩa. Mặc dù đĩa được phủ 1 lớp bảo vệ ỏng nhưng disk head đôi khi sẽ làm hỏng bề mặt từ tính (gọi là head crash)
- Thông thường head crash không thể sửa chữa được, toàn bộ đĩa phải được thay thế và dữ liệu trên đĩa sẽ mất trừ khi được sao lưu sang bộ lưu trữ khác hoặc được bảo vệ bởi RAID.

NVM

Figure 11.3 A 3.5-inch SSD circuit board.

NVM

- NVM dựa trên bộ nhớ Flash thường được sử dụng trong 1 container giống như ổ đĩa, gọi là SSD (solid-state-disk)
- Trường hợp khác có dạng ổ USB (còn gọi là thumb drive hoặc flash drive) hoặc thanh DRAM.
- Ưu điểm
  - đáng tin cậy hơn HDD vì không có thành phần di chuyển
  - nhanh hơn HDD vì không có seek time và rotational latency
  - Tiêu thụ ít năng lượng hơn HDD
- Nhược điểm (đã được cải thiện theo thời gian)
  - Đắt tiền hơn
  - Dung lượng ít hơn

NVM

- Vì các thiết bị NVM nhanh hơn HDD nên standard bus interface có thể gây ra giới hạn về thông lượng.
  - Một số NVM được thiết kế để kết nối trực tiếp với bus hệ thống (vd PCIe).
- Một số hệ thống sử dụng thiết bị NVM thay thế trực tiếp HDD, một số hệ thống khác sử dụng như một 1 tầng bộ đệm mới, di chuyển dữ liệu giữa các đĩa từ, NVM và bộ nhớ chính để tối ưu hóa hiệu suất

NAND Flash Controller Algorithms

- Để theo dõi khối logic nào chứa dữ liệu hợp lệ, bộ điều khiển duy trì flas translation layer (FTL).

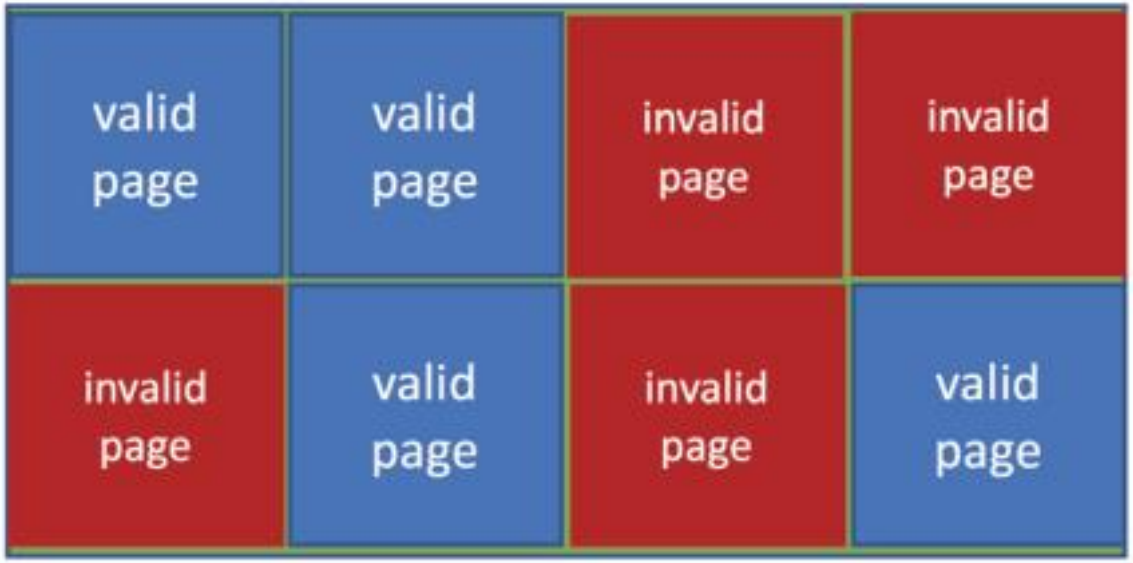


Figure 11.4 A NAND block with valid and invalid pages.

NAND Flash Controller Algorithms

- Khi SSD full thì xử lý yêu cầu ghi thêm dữ liệu ntn?
  - Có thể có khối chứa dữ liệu không hợp lệ → đợi quá trình xóa xảy ra, sau đó ghi
  - Không có khối chứa dữ liệu hợp lệ, thì tìm page riêng lẻ đang chứa dữ liệu không hợp lệ → đợi quá trình thu thập rác (garbage collection)
    - Good data sẽ được lưu trữ → Tính năng cung cấp quá mức (over-provisioning)
- Không gian over-provisioning giúp cân bằng độ hao mòn của thiết bị. Nếu 1 số khối bị xóa liên tục thì sẽ mòn nhanh hơn dẫn đến thiết bị có tuổi thọ ngắn. Controller sẽ tránh bằng cách sử dụng các thuật thoán khác nhau đặt dữ liệu trên các khối ít bị xóa để lần xóa tiếp theo xảy ra trên khối đó.

NAND Flash Controller Algorithms

- Về mặt bảo mật, giống như HDD, thiết bị NVM cung cấp mã sửa lỗi, được tính toán và lưu trữ cùng với dữ liệu trong quá trình đọc và ghi cùng với dữ liệu để phát hiện lỗi và sửa lỗi nếu có thể.
- Nếu một trang thường xuyên có lỗi có thể sửa được thì trang đó có thể bị đánh dấu là xấu và không được sử dụng trong những lần ghi tiếp theo. Nói chung, một thiết bị NVM, như ổ cứng HDD, có thể gặp lỗi nghiêm trọng khiến thiết bị bị hỏng hoặc không phản hồi các yêu cầu đọc hoặc ghi. Để cho phép khôi phục dữ liệu trong những trường hợp đó, tính năng bảo vệ RAID được sử dụng.

Volatile Memory

- Sử dụng DRAM làm thiết bị lưu trữ dữ liệu tạm thời tốc độ cao
  - Thiết bị NVM nhanh nhưng DRAM nhanh hơn nhiều và các thao tác I/O đối với ổ RAM là cách nhanh nhất để tạo, đọc, ghi và xóa file cũng như nội dung của chúng
- Nhiều chương trình sử dụng (hoặc hưởng lợi từ việc sử dụng) ổ RAM để lưu trữ các file tạm thời
  - VD1 các chương trình chia sẻ dữ liệu dễ dàng bằng cách ghi và đọc file từ ổ RAM
  - VD2 Linux khi khởi động sẽ tạo 1 hệ thống file gốc tạm thời (initrd) cho phép các phần khác của hệ thống có quyền truy cập vào hệ thống file gốc và nội dung của nó trước khi các phần của OS hiểu được thiết bị lưu trữ được tải

Secondary Storage Connection Methods

- Thiết bị lưu trữ thứ cấp được gắn vào máy tính bằng bus hệ thống hoặc bus I/O.
  - Các loại bus có sẵn như ATA (advanced technology attachment), SATA (serial ATA), eSATA, SAS (serial attached SCSI), USB (universal serial bus), và FC (fiber channel)
  - Đối với các thiết bị NVM sử dụng NVM express (NVMe). NVMe kết nối trực tiếp thiết bị với bus hệ thống PCI, tăng thông lượng, giảm độ trễ so với các phương thức kết nối khác

Address Mappings

- Các thiết bị lưu trữ được đánh địa chỉ dưới dạng mảng một chiều lớn gồm các khối logic
  - Khối logic là đơn vị truyền tải nhỏ nhất.
  - Mỗi khối logic ánh xạ tới một physical sector hoặc semiconductor page
  - Mảng 1 chiều của các khối logic được ánh xạ lên các sector hoặc page của thiết bị
  - VD sector 0 có thể là sector đầu tiên của track đầu tiên trên cylinder ngoài cùng của HDD
  - Đối với NVM thì ánh xạ từ 1 tuple gồm chip, khối và page tới 1 mảng các khối logic LBA (logical block address)

Address Mappings

- Rất khó thực hiện việc chuyển đổi địa chỉ trên HDD vì:
  - Hầu hết ổ đĩa đều có 1 số sector bị lỗi, nhưng việc ánh xạ sẽ che giấu bằng cách thay thế các sector dự phòng từ nơi khác trên ổ đĩa → địa chỉ logic vẫn tuần tự nhưng địa chỉ vật lý thay đổi
  - Số lượng sector trên track không phải là hằng số trên một số ổ đĩa
  - Các nhà sản xuất đĩa quản lý LBA để ánh xạ địa chỉ vật lý nội bộ, do đó trong các ổ đĩa hiện tại có rất ít mối quan hệ giữa LBA và các physical sector

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - **HDD Scheduling**
  - NVM Scheduling
  - Phát hiện và sửa lỗi
  - Quản lý thiết bị lưu trữ
  - Quản lý không gian hoán đổi (swap-space management)
  - Storage Attachment
  - RAID Structure
  - Summary
- Hệ thống Vào/ Ra (I/O system)

HDD Scheduling

- Có thể cải thiện cả thời gian truy cập bộ nhớ và băng thông truyền dữ liệu bằng cách quản lý thứ tự phục vụ các yêu cầu I/O lưu trữ.
- Bất cứ khi nào tiến trình cần I/O đến hoặc từ ổ đĩa, nó sẽ đưa 1 system call tới OS. Yêu cầu chỉ định 1 số thông tin:
  - Hoạt động này là input hay output
  - Phần xử lý file đang mở cho biết file sẽ hoạt động trên đó
  - Địa chỉ bộ nhớ để truyền là gì
  - Lượng dữ liệu cần truyền
- Nếu có sẵn ổ đĩa và bộ điều khiển mong muốn, yêu cầu sẽ được thực hiện ngay lập tức. Ngược lại, mọi yêu cầu dịch vụ mới sẽ được đặt vào hàng đợi yêu cầu đang chờ xử lý đối với ổ đĩa đó.

HDD Scheduling

- FCFS Scheduling
- SCAN Scheduling
- Selection of a Disk-Scheduling Algorithm

FCFS Scheduling

- Disk queue với yêu cầu cho I/O

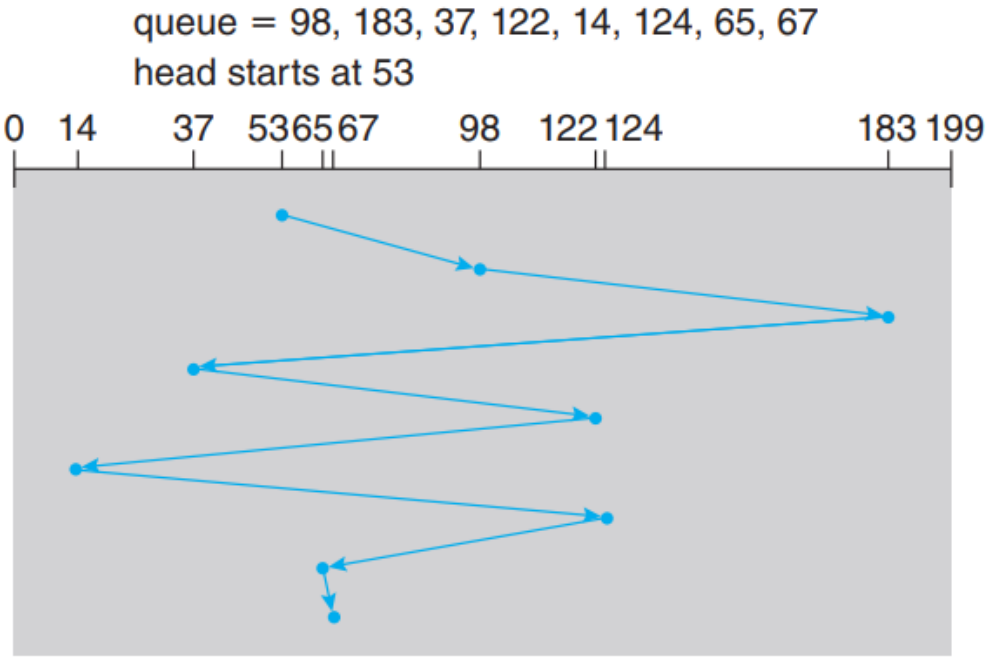


Figure 11.6 FCFS disk scheduling.

SCAN Scheduling

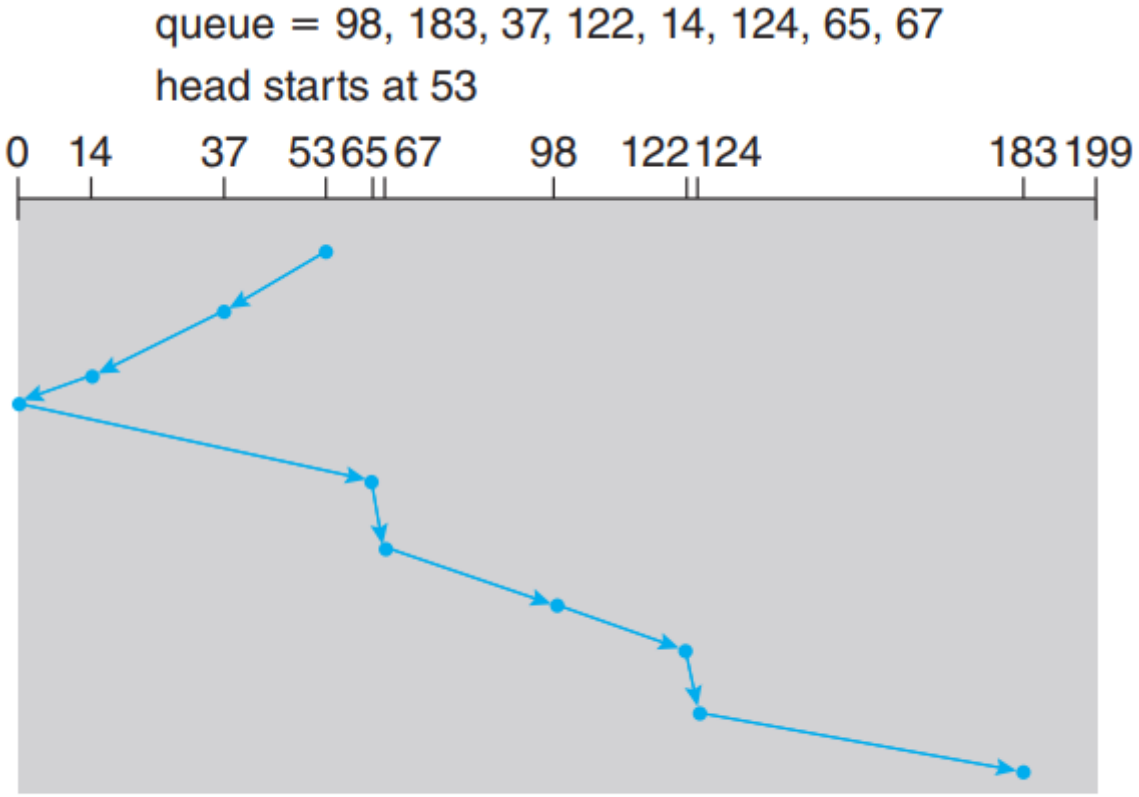


Figure 11.7 SCAN disk scheduling.

C-SCAN Scheduling

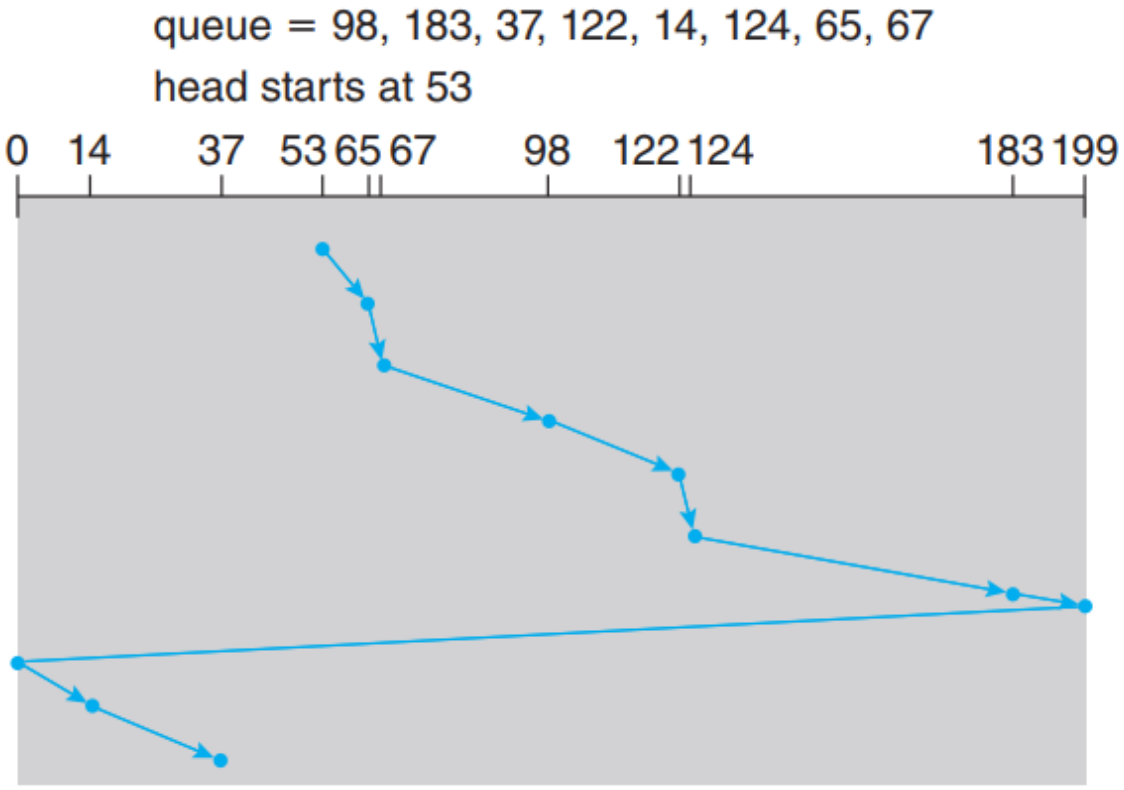


Figure 11.8 C-SCAN disk scheduling.

Selection of a Disk-Scheduling Algorithm

- SCAN và C-SCAN hoạt động tốt hơn đối với các hệ thống đặt tải nặng lên đĩa vì chúng ít có khả năng gây ra sự cố starvation (nhưng vẫn có) → thúc đẩy Linux tạo ra deadline scheduler.
- Các hàng đợi được sắp xếp theo thứ tự LBA, về cơ bản là triển khai C-SCAN.
-
-

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - HDD Scheduling
  - **NVM Scheduling**
  - Phát hiện và sửa lỗi
  - Quản lý thiết bị lưu trữ
  - Quản lý không gian hoán đổi (swap-space management)
  - Storage Attachment
  - RAID Structure
  - Summary
- Hệ thống Vào/ Ra (I/O system)

NVM Scheduling

- Các thiết bị NVM thường sử dụng FCFS đơn giản
  - VD Linux NOOP scheduler sử dụng FCFS nhưng sửa đổi để hợp nhất với các yêu cầu liền kề.
- I/O có thể xảy ra tuần tự hoặc ngẫu nhiên
  - Truy cập tuần tự là tối ưu cho các thiết bị cơ học như ổ cứng và băng vì dữ liệu cần đọc hoặc ghi nằm gần đầu đọc/ ghi.
  - Truy cập ngẫu nhiên, được đo bằng IOPS (input/output operations per second), gây ra chuyển động của đầu đĩa HDD
    - HDD tạo ra hàng trăm IOPS nhưng SSD tạo ra trăm ngàn IOPS

NVM Scheduling

- 1 cách để cải thiện tuổi thọ và hiệu suất của thiết bị NVM theo thời gian là yêu cầu hệ thống file thông báo cho thiết bị khi file bị xóa để thiết bị có thể xóa các khối mà các file đó được lưu trữ trên đó
- Xét 1 thiết bị NVM với tải read và write ngẫu nhiên. Giả sử tất cả các khối đã được write nhưng vẫn còn chỗ trống.
- Garbage collection phải diễn ra để lấy lại không gian bị chiếm bởi dữ liệu không hợp lệ

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - HDD Scheduling
  - NVM Scheduling
  - **Phát hiện và sửa lỗi**
  - Quản lý thiết bị lưu trữ
  - Quản lý không gian hoán đổi (swap-space management)
  - Storage Attachment
  - RAID Structure
  - Summary
- Hệ thống Vào/ Ra (I/O system)

Phát hiện và sửa lỗi

- Phát hiện lỗi xác định nếu vấn đề xảy ra
  - ví dụ 1 bit trong DRAM tự động thay đổi từ 0 thành 1, nội dung của gói mạng đã thay đổi trong quá trình truyền hoặc 1 khối dữ liệu đã thay đổi giữa thời điểm nó được write và khi nó được read.
- Bằng cách phát hiện sự cố, hệ thống có thể tạm dừng hoạt động trước khi lỗi lan truyền, báo cáo lỗi cho người dùng hoặc quản trị viên hoặc cảnh báo thiệt bị có thể bắt đầu bị lỗi hoặc đã bị lỗi
- Các hệ thống bộ nhớ từ lâu đã phát hiện ra một số lỗi nhất định bằng cách sử dụng các bit chẵn lẻ (checksum)
- Cách khác là sử dụng CRC (cyclic redundancy check) – sử dụng hash function để xác định lỗi trên nhiều bit

Phát hiện và sửa lỗi

- ECC (error-correction code) không chỉ phát hiện mà còn sửa lỗi.
  - Việc chỉnh sửa được thực hiện bằng cách sử dụng thuật toán và dung lượng lưu trữ bổ sung
  - Các code khác nhau tuy theo dung lượng lưu trữ bổ sung mà chúng cần và số lỗi chúng có thể sửa
  - Nếu có quá nhiều thay đổi xảy ra và ECC không thể sữa lỗi thì sẽ báo hiệu 1 lỗi cứng không thể sửa được. Bộ điều khiển tự động thực hiện xử lý ECC bất cứ khi nào một sector hoặc page được đọc hoặc ghi
- Việc phát hiện và sửa lỗi thường là điểm khác biệt giữa sản phẩm tiêu dùng và sản phẩm doanh nghiệp. Ví dụ: ECC được sử dụng trong một số hệ thống để sửa lỗi DRAM và bảo vệ đường dẫn dữ liệu.

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - HDD Scheduling
  - NVM Scheduling
  - Phát hiện và sửa lỗi
  - **Quản lý thiết bị lưu trữ**
  - Quản lý không gian hoán đổi (swap-space management)
  - Storage Attachment
  - RAID Structure
  - Summary
- Hệ thống Vào/ Ra (I/O system)

Quản lý thiết bị lưu trữ

- Định dạng ổ đĩa, phân vùng và kích thước ổ đĩa
- Boot Block
- Bad Blocks

Định dạng ổ đĩa, phân vùng và kích thước ổ đĩa

- 1 thiết bị lưu trữ mới là 1 bản trống: nó chỉ là 1 đĩa vật liệu ghi từ tình hoặc một tập hớp các ô lưu trữ bán dẫn chưa được khởi tạo
- Trước khi lưu trữ, nó sẽ được chia thành các sector mà bộ điều khiển có thể đọc và ghi
- NVM page phải được khởi tạo và FTL được tạo ra. Quá trình này gọi là low-level formatting, hoặc physical formatting.
- Low-level formatting sẽ lấp đầy thiết bị bằng CTDL đặc biệt cho từng vị trí lưu trữ. CTDL cho 1 sector hoặc page thường bao gồm tiêu đề (header), vùng dữ liệu (data area) và trailer.
  - Header và trailer chứa thông tin được bộ điều khiển sử dụng, chẳng hạn như số sector/ page và 1 mã phát hiện hoặc sửa lỗi

Định dạng ổ đĩa, phân vùng và kích thước ổ đĩa

- Hầu hết ổ đĩa đều được được định dạng cấp thấp tại nhà máy như một phần của quy trình sản xuất → định dạng này cho phép nhà sản xuất kiểm tra thiết bị và khởi tạo ánh xạ từ số khối logic đến các sector hoặc page không có lỗi
- Thông thường có thể chọn trong một số kích thước sector, chẳng hạn 512 bytes và 4KB
- Định dạng ổ đĩa có kích thước sector lớn thì số lượng sector trên mỗi track (rãnh) sẽ ít, cũng có nghĩa là ít header và trailer hơn được ghi trên mỗi rãnh và có nhiều không gian hơn cho dữ liệu người dùng.
- Một số OS chỉ có thể xử lý một kích thước sector cụ thể

Định dạng ổ đĩa, phân vùng và kích thước ổ đĩa

- Trước khi có thể sử dụng ổ đĩa để lưu trữ file, OS vẫn cần ghi lại CTDL của chính nó lên thiết bị trong 3 bước:
  - Phân vùng thiết bị thành 1 hoặc nhiều nhóm khối dữ liệu hoặc page dữ liệu. OS có thể coi mỗi phân vùng như thể nó là 1 thiết bị riêng biệt
  - Tạo và quản lý volume.
  - Định dạng logic hoặc tạo hệ thống file. Ở bước này, OS lưu trữ CTDL hệ thống file ban đầu vào thiết bị. Các CTDL nầy có thể bao gồm các ánh xạ của không gian trống và không gian đã được cấp phát cũng như một thư mục trống ban đầu

Định dạng ổ đĩa, phân vùng và kích thước ổ đĩa

- Thông tin phân vùng cũng cho biết liệu phân vùng có chứa hệ thống file có khả năng khởi động hay không (chứa OS).
- Phân vùng có nhãn khởi động được sử dụng để thiết lập thư mục gốc của hệ thống tập tin

Định dạng ổ đĩa, phân vùng và kích thước ổ đĩa

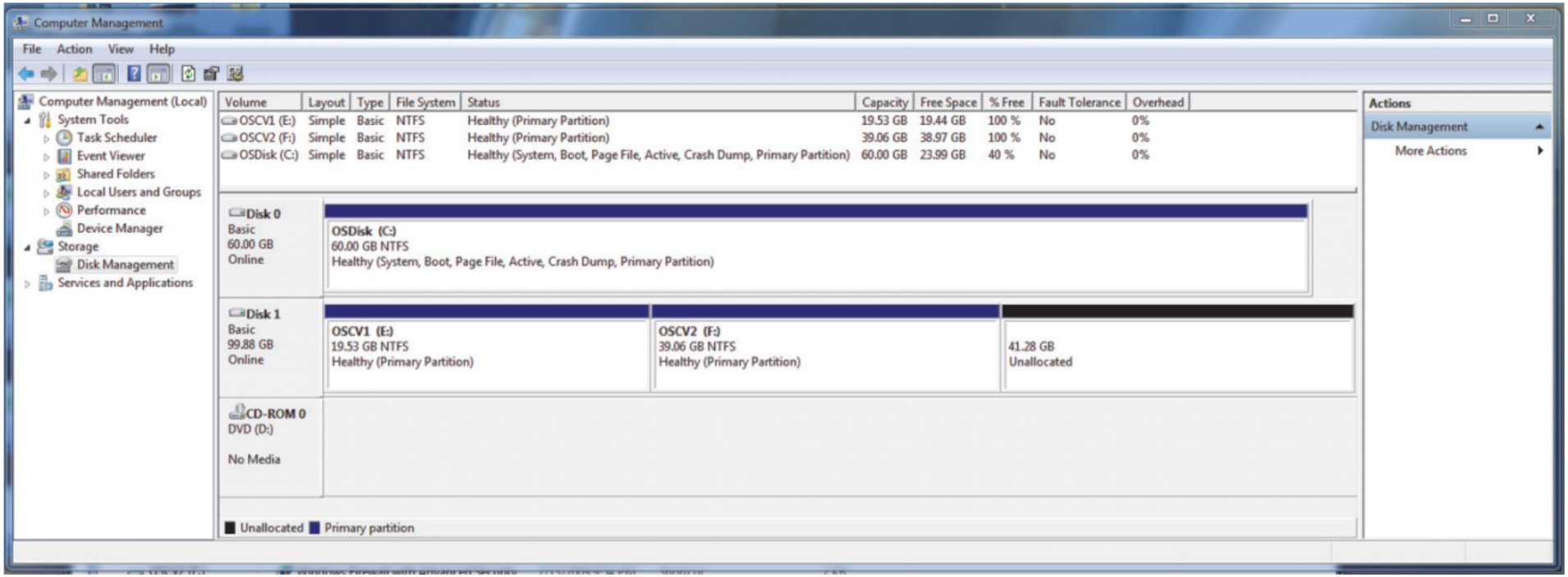


Figure 11.9 Windows 7 `Disk Management` tool showing devices, partitions, volumes, and file systems.

Boot Block

- Để máy tính bắt đầu chạy, nó phải có chương trình ban đầu để chạy
- Trình tải bootstrap ban đầu này có xu hướng đơn giản. Đối với hầu hết máy tính, nó được lưu trữ trong NVM flash memory firmware trên motherboard hệ thống và được ánh xạ tới một vị trí bộ nhớ đã biết
- Trình tải bootstrap nhỏ này đủ thông minh để mang trình bootstrap đầy đủ từ bộ lưu trữ thứ cấp. Full bootstrap program được lưu trữ trong “boot blocks” tại một vị trí cố định trên thiết bị.

Boot Block

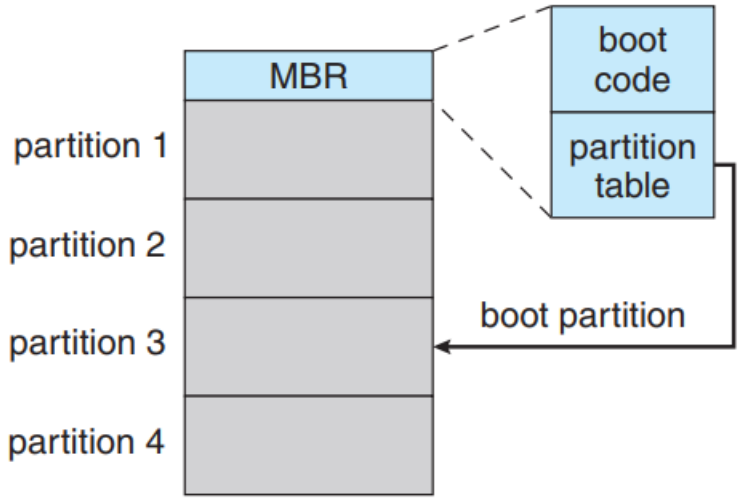


Figure 11.10 Booting from a storage device in Windows.

- VD trên Windows
  - 1 trong các phân vùng là boot partition – chứa OS và driver thiết bị
  - Hệ thống Windows đặt boot code ở block logic đầu tiên trên ổ cứng hoặc page đầu tiên của thiết bị NVM, gọi là MBR (master boot record).
  - Boot bắt đầu bằng cách chạy code trong firmware của hệ thống
  - Trong MBR chứa 1 bảng liệt kê các phân vùng và cờ đánh dấu phân vùng nào boot hệ thống.

Bad Blocks

- Đĩa có các bộ phận chuyển động và dung sai nhỏ nên chúng dễ bị hỏng
  - Nếu hỏng hoàn toàn thì thay mới và nội dung của nó được khôi phục từ phần backup
  - Vấn đề thường gặp hơn là 1 hoặc nhiều sector bị lỗi, nhiều đĩa thậm chí còn có các bad block từ nhà máy
- Trên các ổ đĩa cũ (vd ổ đĩa có bộ điều khiển IDE) các khối lỗi xấu được xử lý theo cách thủ công.
  - Quét đĩa để tìm các bad block trong khi đĩa đang được định dạng
  - Gắn cờ cho bad block để hệ thống file không sử dụng chúng
  - Nếu các khối bị hỏng trong quá trình sử dụng, có 1 chương trình đặc biệt được chạy thủ công để tìm kiếm các bad block và khóa chúng đi.

Bad Blocks

- Các đĩa phức tạp hơn sẽ thông minh hơn trong việc khôi phục bad block. Bộ điều khiển duy trì một danh sách bad block trên đĩa. Danh sách này được khởi tạo trong quá trình định dạng low-level tại nhà máy và được cập nhật trong suốt thời gian sử dụng của đĩa.
- Định dạng low-level cũng dành riêng các sector dự phòng mà OS không nhìn thầy được. Bộ điều khiển có thể được yêu cầu thay thế từng bad sector bằng một trong các sector dự phòng. Cơ chế này được gọi là sector sparing hoặc forwarding.

Bad Blocks

- Một transaction bad sector điển hình có thể như sau:
  - OS đọc block logic 87
  - Bộ điều khiển tính toán ECC và nhận thấy nó là bad sector. Nó báo cáo tới OS như 1 lỗi I/O
  - Bộ điều khiển thiết bị thay thế bad sector bằng sector dự phòng
  - Sau đó, bất cứ khi nào hệ thống yêu cầu block logic 87, yêu cầu đó sẽ được bộ điều khiển dịch sang địa chỉ sector thay thế

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - HDD Scheduling
  - NVM Scheduling
  - Phát hiện và sửa lỗi
  - Quản lý thiết bị lưu trữ
  - **Quản lý không gian hoán đổi (swap-space management)**
  - Storage Attachment
  - RAID Structure
  - Summary
- Hệ thống Vào/ Ra (I/O system)

Quản lý không gian hoán đổi

- Swap space được sử dụng theo nhiều cách khác nhau bởi các OS khác nhau, tùy thuộc vào thuật toán quản lý bộ nhớ được sử dụng.
- Chú ý rằng việc đánh giá cao và cấp phát dung lượng lớn cho swap space có thể an toàn hơn vì nếu một hệ thống hết swap space có thể buộc phải hủy bỏ các tiến trình hoặc có thể bị hỏng hoàn toàn.
  - Có thể lãng phí không gian lưu trữ thứ cấp mà có thể được sử dụng cho file, nhưng không gây hại gì khác
- Nhiều OS – bao gồm Linux – cho phép sử dụng nhiều swap space, bao gồm cả file và phân vùng swap chuyên dụng. Các swap space này thường được đặt trên các thiết bị lưu trữ riêng biệt để tải được lên hệ thống I/O bằng cách paging và swaping có thể được trải rộng trên băng thông I/O của hệ thống.

Quản lý không gian hoán đổi

- Swap space có thể nằm 1 trong 2 nơi:
  - Có thể được tạo ra từ hệ thống file thông thường
    - Ưu điểm: các thủ tục hệ thống file thông thường có thể được sử dụng để tạo, đặt tên, cấp phát không gian cho nó
  - Có thể nằm trong 1 phân vùng riêng biệt
    - Ưu điểm: có trình quản lý riêng sử dụng các thuật toán tối ưu hóa cho tốc độ thay vì hiệu quả lưu trữ.
- Một số OS linh hoạt có thể swap cả trong không gian file và phân vùng.
  - VD Linux: chính sách và cách triển khai là riêng biệt, cho phép quản trị viên quyết định sử dụng loại swap nào

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - HDD Scheduling
  - NVM Scheduling
  - Phát hiện và sửa lỗi
  - Quản lý thiết bị lưu trữ
  - Quản lý không gian hoán đổi (swap-space management)
  - **Storage Attachment**
  - RAID Structure
  - Summary
- Hệ thống Vào/ Ra (I/O system)

Storage Attachment

- Máy tính truy cập bộ nhớ thứ cấp theo ba cách:
  - Host-attached storage
  - Network-attached storage
  - Cloud storage
  - Storage-Area Networks and Storage Arrays

Host-attached storage

- Bộ nhớ gắn trên máy chủ là bộ nhớ được truy cập thông qua các cổng I/O cục bộ. Các cổng này sử dụng một số công nghệ, phổ biến nhất là SATA, một hệ thống điển hình có một hoặc vài cổng SATA
- Để cho phép hệ thống truy cập vào nhiều bộ nhớ hơn, thiết bị lưu trữ riêng lẻ, thiết bị trong khung hoặc nhiều ổ đĩa trong khung có thể được kết nối qua cổng và cáp USB FireWire hoặc Thunderbolt.
- Các máy trạm và máy chủ cao cấp thường cần nhiều bộ nhớ hơn hoặc cần chia sẻ bộ nhớ, vì vậy sử dụng các kiến trúc I/O phức tạp hơn, chẳng hạn như FC (fiber channel)
- Có nhiều loại thiết bị lưu trữ phù hợp để sử dụng làm thiết bị lưu trữ với máy chủ: HDD, thiết bị NVM, CD, DVD, Blu-ray, ...

Network-Attached Storage

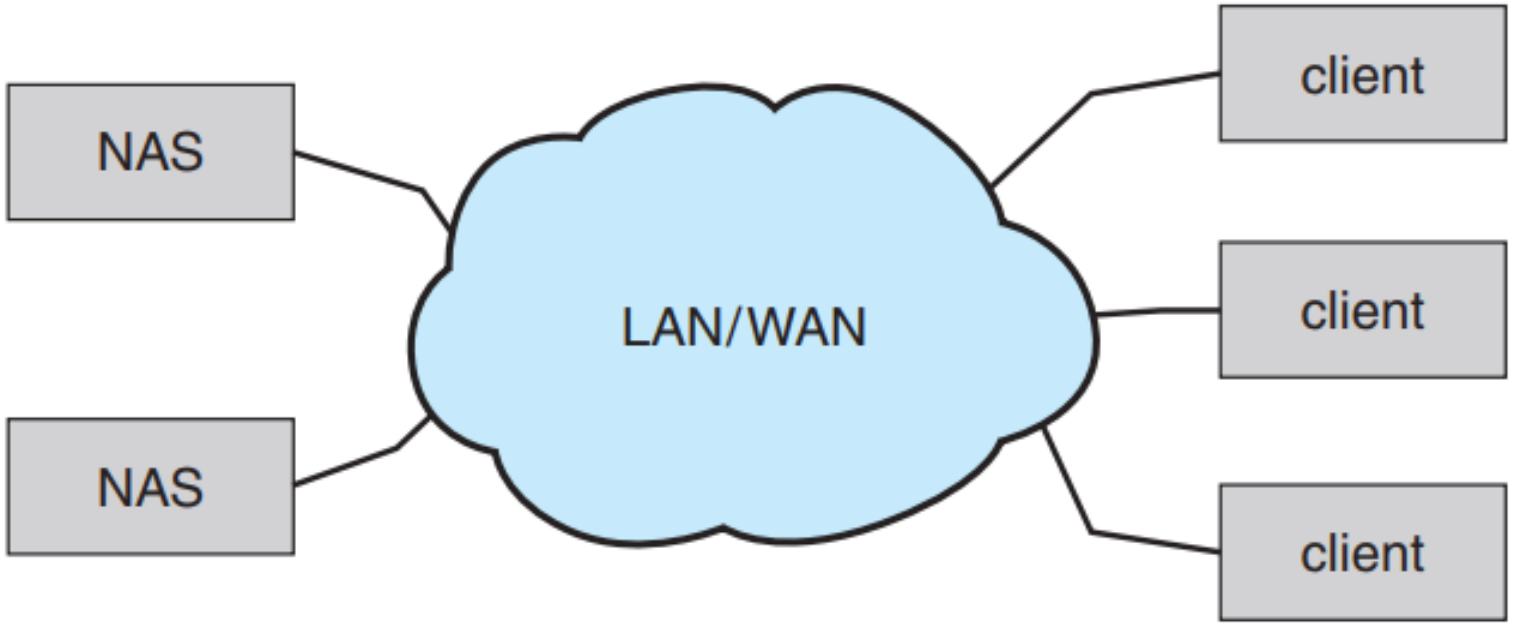


Figure 11.12 Network-attached storage.

Network-Attached Storage

- Bộ nhớ lưu trữ gắn mạng (NAS) cung cấp quyền truy cập vào bộ lưu trữ trên mạng.
- Thiết bị NAS có thể là hệ thống lưu trữ có mục đích đặc biệt hoặc hệ thống máy tính chung cung cấp khả năng lưu trữ cho các máy chủ khác trên mạng.
- Client truy cập thông qua remote-procedure all interface như NFS cho hệ thống UNIX và Linux hoặc CIFS cho máy Windows. Các cuộc gọi thủ tục từ xa (RPC) được thực hiện qua TCP hoặc UDP qua mạng IP – thường là cùng 1 LAN

Cloud Storage

- Tương tự với bộ nhớ lưu trữ gắn mạng, lưu trữ đám mây cung cấp quyền truy cập vào bộ nhớ lưu trữ trên mạng.
- Không giống NAS, bộ lưu trữ được truy cập qua Internet hoặc mạng WAN
- 1 điểm khác nữa là cách lưu trữ được truy cập và hiển thị cho người dùng
  - NAS được truy cập như 1 hệ thống file khác nếu sử dụng giao thức CIFS hoặc NFS hoặc như 1 thiết bị khối thô nếu sử dụng iSCSI
  - Lưu trữ đám mây dựa trên APU và các chương trình sử dụng API để truy cập vào bộ lưu trữ

Storage-Area Networks and Storage Arrays

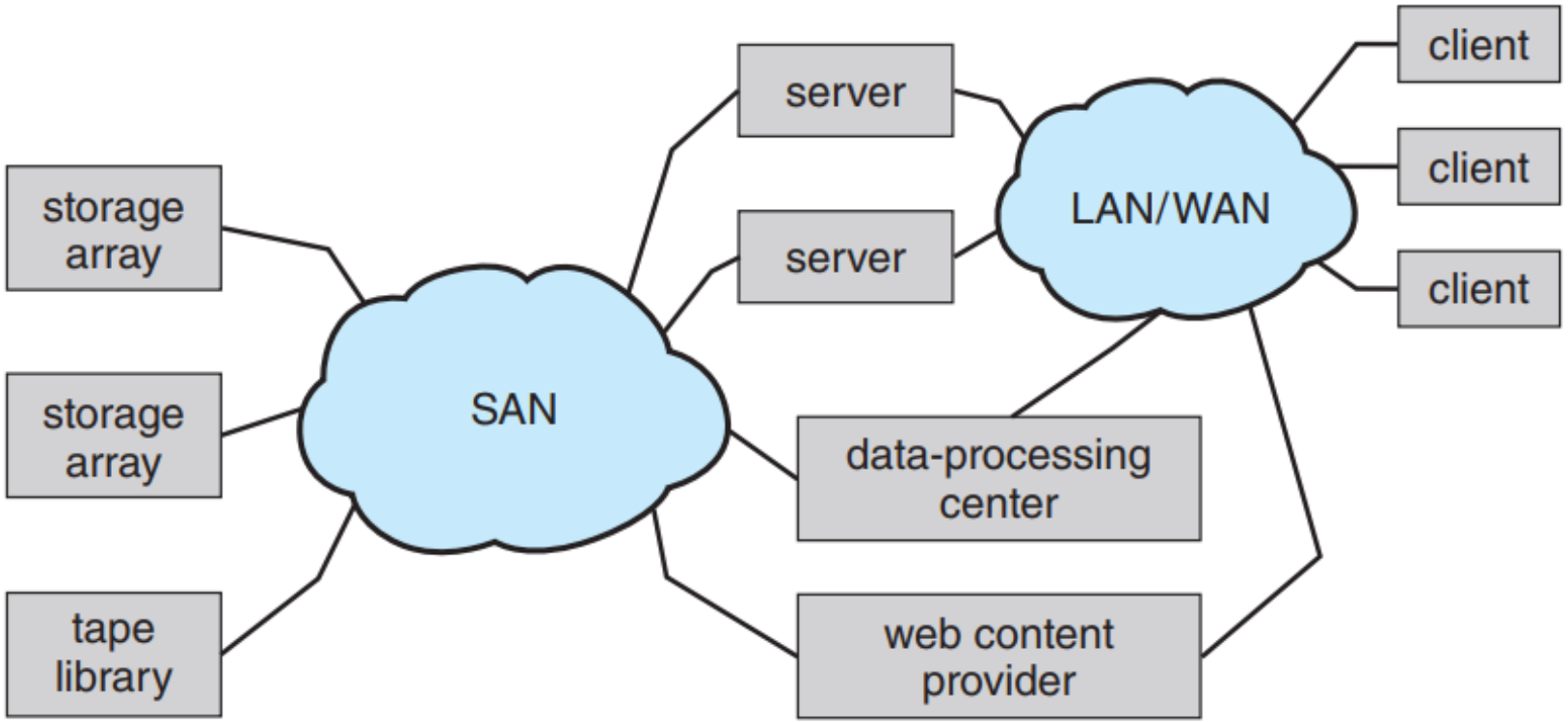


Figure 11.13 Storage-area network.

Storage-Area Networks and Storage Arrays

- Mạng vùng lưu trữ (SAN) là mạng riêng (sử dụng giao thức lưu trữ thay vì giao thức mạng) kết nối máy chủ và thiết bị lưu trữ
- Sức mạnh của SAN nằm ở tính linh hoạt của nó
  - Nhiều máy chủ và nhiều mảng lưu trữ có thể gắn vào cũng một SAN và bộ nhớ có thể được cấp phát động cho các máy chủ
  - Mảng lưu trữ có thể là ổ đĩa được bảo vệ hoặc không được bảo vệ RAID
- Bộ chuyển mạch SAN cho phép hoặc cấm truy cập giữa máy chủ và bộ lưu trữ. VD nếu 1 máy chủ hết dung lượng ổ đĩa, SAN có thể được cấu hình để cấp phát thêm dung lượng lưu trữ cho máy chủ đó. SAN giúp các cụm máy chủ có thể chia sẻ cùng 1 bộ lưu trữ và các mảng lưu trữ có thể bao gồm nhiều kết nối máy chủ trực tiếp

Storage-Area Networks and Storage Arrays

Figure 11.14 A storage array.

Storage-Area Networks and Storage Arrays

- Mảng lưu trữ là một thiết bị được xây dựng có mục đích bao gồm các cổng SAN, cổng mạng hoặc cả 2. Nó cũng chứa các ổ đĩa để lưu trữ dữ liệu và bộ điều khiển để quản lý bộ lưu trữ và cho phép truy cập vào bộ lưu trữ trên mạng.
- Bộ điều khiển bao gồm CPU, bộ nhớ và phần mềm triển khai các tính năng của mảng, có thể bao gồm các giao thức mạng, giao diện người dùng, bảo vệ RAID, snapshots, sao chép, nén, chống trùng lặp và mã hóa.
- Một số mảng lưu trữ bao gồm SSD. Một mảng có thể chỉ chứa SSD, mang lại hiệu suất tối đa nhưng dung lượng nhỏ hơn hoặc có thể bao gồm sự kết hợp giữa SSD và HDD, với phần mềm mảng (hoặc quản trị viên) chọn phương tiện tốt nhất cho mục đích sử dụng nhất được hoặc sử dụng SSD làm bộ đệm và HDD như bộ nhớ số lượng lớn.

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - HDD Scheduling
  - NVM Scheduling
  - Phát hiện và sửa lỗi
  - Quản lý thiết bị lưu trữ
  - Quản lý không gian hoán đổi (swap-space management)
  - Storage Attachment
  - **RAID Structure**
  - Summary
- Hệ thống Vào/ Ra (I/O system)

RAID Structure (Redundant arrays of independent disk)

- Cải thiện độ tin cậy thông qua dự phòng
- Cải thiện hiệu suất thông qua song song hóa
- RAID Levels
- Chọn RAID Level
- Vấn đề với RAID

Cải thiện độ tin cậy thông qua dự phòng

- Lưu trữ thông tin bổ sung (thường không cần thiết) có thể được sử dụng trong trường hợp đĩa bị hỏng để xây dựng lại thông tin bị mất
- RAID có thể được áp dụng cho các thiết bị NVM
- Cách tiếp cận đơn giản nhất (tốn kém nhất) tạo dự phòng là sao chép mọi ổ đĩa – kỹ thuật phản chiếu
  - 1 đĩa logic gồm 2 ổ đĩa vật lý, mọi thao tác đều được thực hiện trên cả 2 ổ đĩa
- Nếu 1 trong các ổ đĩa bị lỗi, dữ liệu có thể đọc từ ổ đĩa kia. Dữ liệu chỉ bị mất nếu ổ thứ 2 bị lỗi trước khi ổ đầu tiên được thay thế

Cải thiện độ tin cậy thông qua dự phòng

- MTBF của đĩa được phản chiếu – trong đó lỗi là do mất dữ liệu – phụ thuộc vào 2 yếu tố.
  - MTBF – Thời gian trung bình giữa các lần lỗi của từng ổ đĩa
  - MTTR - Thời gian trung bình để sửa chữa – thời gian cần thiết (trung bình) để thay thế một ổ đĩa bị lỗi và khôi phục dữ liệu trên đó
- Giả sử 2 ổ đĩa độc lập, nếu MTBF của 1 ổ đĩa là 100.000 giờ và MTTR là 10 giờ, thì thời gian trung bình dẫn đến mất dữ liệu của hệ thống ổ đĩa là $\frac{100000^2}{2\times10} = 500 \times 10^6$ giờ, hoặc 57,000 years.

Cải thiện hiệu suất thông qua song song hóa

- Vì sử dụng tính năng dự phòng, nên tốc độ xử lý các yêu cầu đọc sẽ tăng gấp đôi
- Với nhiều ổ đĩa, có thể cải thiện tốc độ truyền tải bằng cách phân chia dữ liệu trên các ổ đĩa.
  - Đơn giản nhất là phân chia dữ liệu của từng byte lên nhiều ổ đĩa (phân chia cấp độ bit). VD có 8 ổ đĩa, ghi bit i của mỗi byte vào ổ i → có thể được coi là 1 ổ đơn với các sector có kích thước gấp 8 lần → tốc độ truy cập gấp 8 lần
  - Phân chia cấp độ khối, các khối của fiel sẽ được phân chia trên nhiều ổ đĩa, với n ổ đĩa, khối i của file sẽ được chuyển đến ổ (i % n) + 1

Cải thiện hiệu suất thông qua song song hóa

- Tính song song trong hệ thống lưu trữ đạt được thông qua việc phân chia có 2 mục tiêu chính:
  - Tăng thông lượng (throughput) của nhiều truy cập nhỏ (là truy cập page) bằng cách cân bằng tải
  - Giảm thời gian phản hồi của các truy cập lớn
- Đọc (read) có thể được thực hiện song song → tăng tốc độ
- Ghi (write) không thể được thực hiện song song
  - Việc khi yêu cầu đọc khối dữ liệu, sửa đổi, ghi lại, cập nhật khối chẳn lẽ - chu trình đọc – sửa – ghi. Do đó 1 lần ghi yêu cầu 4 lần truy cập vào ổ đĩa, 2 lần để đọc 2 khối cũ, 2 lần để ghi 2 khối mới

RAID Levels

- Kỹ thuật phản chiếu cung cấp độ tin cậy cao nhưng nó đắt tiền.
- Phân chia cấp độ khối cung cấp tốc độ truyền dữ liệu cao nhưng không cải thiện độ tin cậy.
- Kết hợp phân chia với các bit chẵn lẻ được đề xuất nhằm cung cấp dự phòng với chi phí thấp.

RAID Levels

- RAID cấp 0 có phân chia ở cấp độ khối nhưng không có dự phòng
- RAID cấp 1 có dự phòng
- RAID cấp 4 còn được gọi là tổ chức mã sửa lỗi (ECC) kiểu bộ nhớ
- RAID cấp 5 có phân tán xen kẽ ECC với khối dữ liệu
- RAID cấp 6 – Sơ đồ dự phòng P+Q
- RAID đa chiều cấp 6 – lưu trữ phức tạp khuếch đại RAID cấp 6 (chứa hàng trăm ổ đĩa)

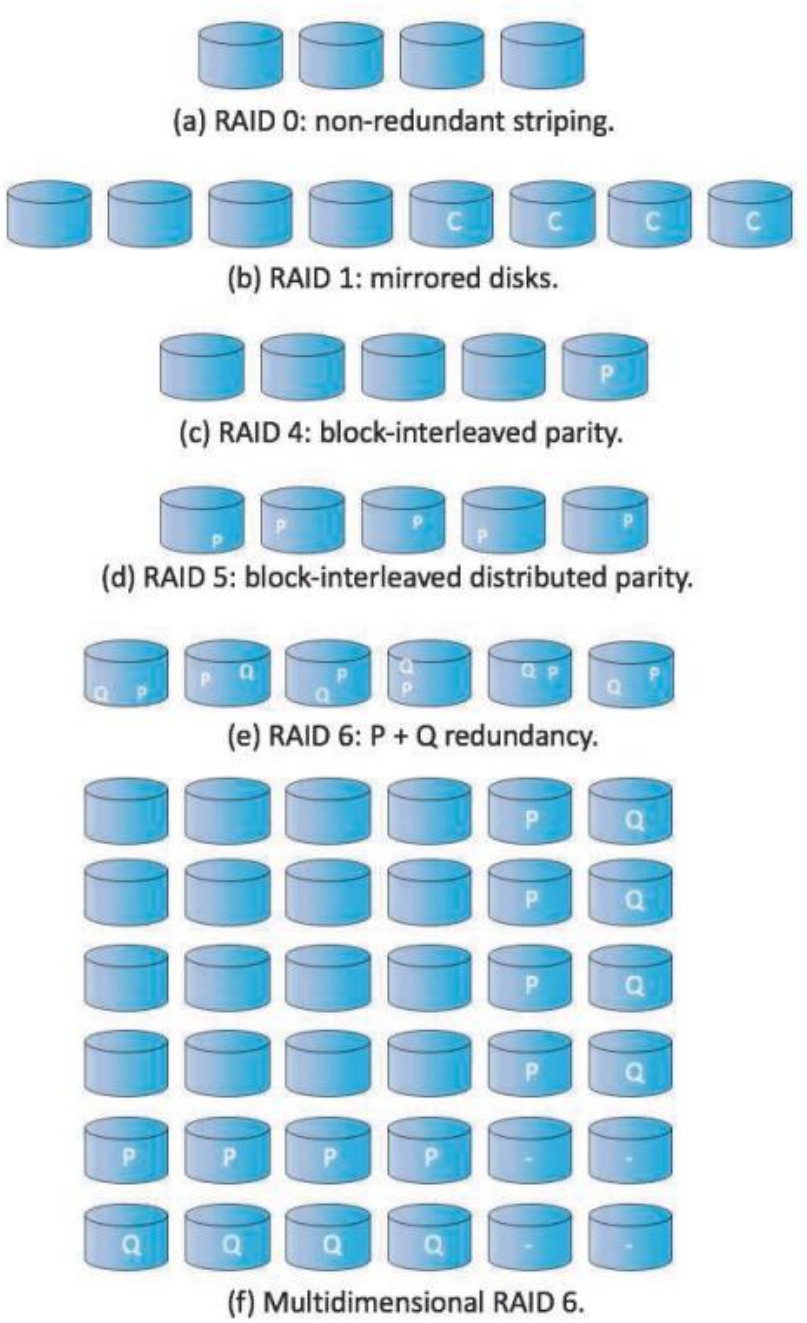


Figure 11.15 RAID levels.

RAID Levels

- Ý tưởng về ECC có thể được sử dụng trực tiếp trong mảng lưu trữ thông qua việc phân chia các khối trên ổ đĩa
  - VD: block đầu tiên được lưu trong ổ đĩa 1, khối thứ 2 trong ổ đĩa 2, ... Cho đến khi khối thứ N được lưu trong ổ đĩa N; kết quả tính toán sửa lỗi của các khối đó được lưu trữ trên ổ N+1.
- Khi 1 trong các ổ đĩa bị lỗi, ECC sẽ tính toán phát hiện ra và ngăn dữ liệu được chuyển đến tiến trình yêu cầu.

RAID Levels

- RAID cấp 0+1 kết hợp hiệu suất của RAID 0 và độ tin cậy của RAID 1
  - Cung cấp hiệu suất tốt hơn RAID 5
  - Phổ biến trong môi trường mà cả hiệu suất và độ tin vậy đều quan trọng
  - Đắt do tăng gấp đôi số lượng đĩa cần thiết để lưu trữ
- RAID cấp 1+0 trong đó các ổ đĩa được phản chiếu theo cặp

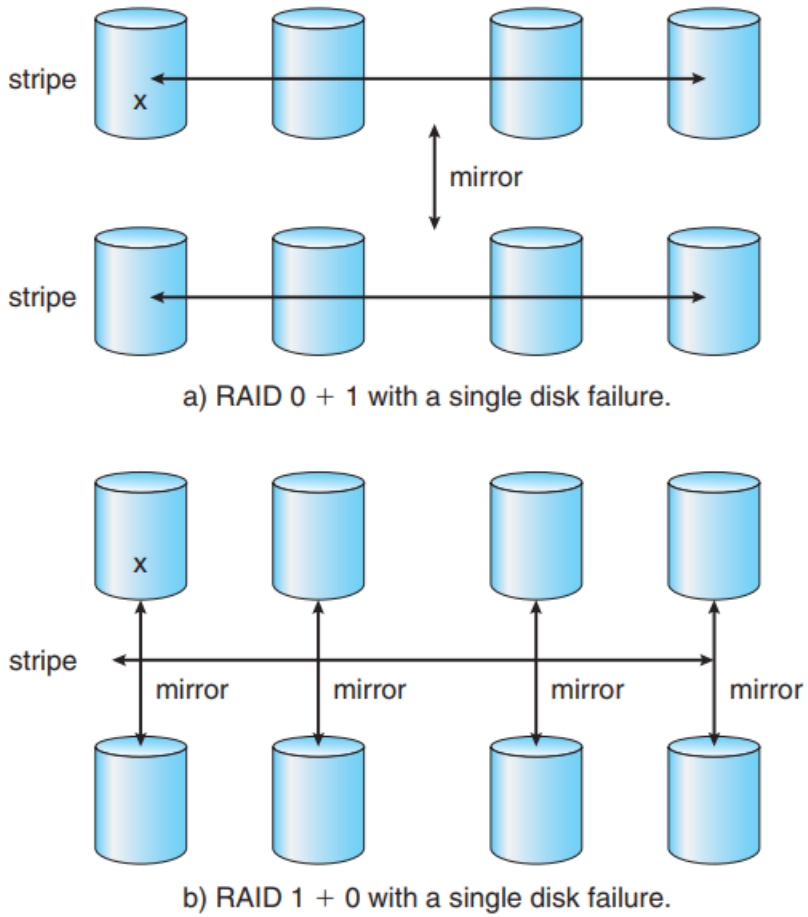


Figure 11.16 RAID 0 + 1 and 1 + 0 with a single disk failure.

Chọn RAID Level

- Điều cần cân nhắc khi chọn Level RAID là xây dựng lại hiệu suất khi ổ đĩa gặp lỗi sau 1 thời gian sử dụng
  - Việc xây dựng lại là dễ dàng nhất đối với RAID cấp 1 vì dữ liệu có thể sao chép từ ổ đĩa khác
  - Đối với các ổ đĩa khác cần phải truy cập vào tất cả các ổ đĩa khác trong mảng để xây dựng lại dữ liệu trong ổ đĩa bị lỗi. VD RAID cấp 5 của bộ ổ đĩa lớn mất hàng giờ để khôi phục lại

Chọn RAID Level

- RAID cấp 0 được sử dụng trong các ứng dụng hiệu suất cao trong đó việc mất dữ liệu không nghiêm trọng
- RAID cấp 1 phổ biến cho các ứng dụng yêu cầu độ tin cậy cao và khả năng phục hồi nhanh
- RAID cấp 0+1 và 1+0 được sử dụng khi cả hiệu suất và độ tin cậy đều quan trọng
- Do chi phí cao nên RAID cấp 1 và RAID cấp 5 thường được ưu tiên để lưu trữ khối lượng dữ liệu vừa phải.
- RAID cấp 6 và RAID cấp 6 đa chiều là những định dạng phổ biến nhất trong mảng lưu trữ vì cung cấp hiệu suất và khả năng bảo vệ tốt mà không tốn nhiều không gian

Chọn RAID Level

- Các nhà thiết kế hệ thống RAID và quản trị viên lưu trữ cũng phải đưa ra một số quyết định khác.
  - Nên có bao nhiêu ổ đĩa trong 1 nhóm RAID nhất định? → Có nhiều ổ đĩa tốc độ truyền dữ liệu sẽ cao hơn nhưng hệ thống sẽ đắt hơn
  - Có ao nhiêu bit cần được bảo vệ bởi mỗi bit chẳn lẽ? → Nhiều bit hơn được bảo vệ bởi 1 bit chẵn lẻ thì chi phí không gian do các bit chẵn lẽ sẽ thấp hơn nhưng khả năng ổ đĩa thứ 2 bị lỗi trước khi ổ đĩa đầu tiên được sửa sẽ lớn hơn và điều đó dẫn đến mất dữ liệu

Vấn đề với RAID

- Việc khi không đầy đủ (torn writes) nếu không được khôi phục đúng cách, có thể dẫn đến dữ liệu bị hỏng
  - Một số tiến trình có thể vô tình ghi lên cấu trúc của hệ thống file
  - RAID bảo vệ khỏi các lỗi phương tiện vật lý nhưng không bảo vệ khỏi các lỗi phần cứng và phần mềm khác. Lỗi bộ điều khiển RAID phần cứng hoặc lỗi trong mã RAID phần mềm có thể dẫn đến mất toàn bộ dữ liệu

Vấn đề với RAID

- Việc triển khai RAID thiếu tính linh hoạt
  - Xem xét 1 mảng lưu trữ có 20 ổ đĩa được chia thành 4 bộ 5 ổ đĩa, mỗi bộ năm ổ đĩa là 1 bộ RAID cấp 5. Kết quả có 4 tập riêng biệt, mỗi tập chứa 1 hệ thống file.
  - Điều gì xảy ra khi 1 hệ thống file quá lớn để vừa với bộ RAID cấp 5 gồm 5 ổ đĩa?
  - Nếu một hệ thống file khác cần rất ít dung lượng thì sao?

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
  - Tổng quan
  - HDD Scheduling
  - NVM Scheduling
  - Phát hiện và sửa lỗi
  - Quản lý thiết bị lưu trữ
  - Quản lý không gian hoán đổi (swap-space management)
  - Storage Attachment
  - RAID Structure
  - **Summary**
- Hệ thống Vào/ Ra (I/O system)

Summary

- HDD và NVM là các đơn vị I/O lưu trữ thứ cấp chính trên hầu hết các máy tính. Bộ lưu trữ thứ cấp hiện đại được cấu trúc dưới dạng mảng một chiều lớn các khối logic
- Các ổ đĩa thuộc một trong hai loại có thể được gắn vào hệ thống máy tính theo một trong ba cách: (1) thông qua các cổng I/O cục bộ trên máy tính chủ, (2) kết nối trực tiếp với bo mạch chủ hoặc (3) thông qua mạng truyền thông hoặc bộ lưu trữ Kết nối mạng.
- Yêu cầu I/O lưu trữ thứ cấp được tạo bởi hệ thống file và hệ thống bộ nhớ ảo. Mỗi yêu cầu chỉ định địa chỉ trên thiết bị được tham chiếu dưới dạng số khối logic.

Summary

- Các thuật toán lập lịch đĩa có thể cải thiện băng thông hiệu quả của ổ cứng, thời gian phản hồi trung bình và sự khác biệt về thời gian phản hồi. Các thuật toán như SCAN và C-SCAN được thiết kế để thực hiện những cải tiến như vậy thông qua các chiến lược sắp xếp hàng đợi đĩa. Hiệu suất của các thuật toán lập lịch đĩa có thể khác nhau rất nhiều trên các đĩa cứng. Ngược lại, do các đĩa thể rắn không có bộ phận chuyển động nên hiệu suất thay đổi rất ít giữa các thuật toán lập lịch và thường sử dụng chiến lược FCFS đơn giản.
- Việc lưu trữ và truyền dữ liệu rất phức tạp và thường xuyên gây ra lỗi. ECC cố gắng phát hiện những vấn đề như vậy để cảnh báo hệ thống về hành động khắc phục và tránh lan truyền lỗi. Sửa lỗi có thể phát hiện và sửa chữa các vấn đề, tùy thuộc vào lượng dữ liệu sửa lỗi có sẵn và lượng dữ liệu bị hỏng.

Summary

- Các thiết bị lưu trữ được phân chia thành một hoặc nhiều khối không gian. Mỗi phân vùng có thể chứa một ổ đĩa (volume) hoặc là một phần của ổ đĩa đa thiết bị (multidevice volume). Hệ thống file được tạo ra theo khối lượng (volumes).
- Hệ điều hành quản lý các khối của thiết bị lưu trữ. Các thiết bị mới thường được định dạng sẵn. Thiết bị được phân vùng, hệ thống tệp được tạo và các khối khởi động được phân bổ để lưu trữ chương trình khởi động của hệ thống nếu thiết bị chứa hệ điều hành. Cuối cùng, khi một khối hoặc trang bị hỏng, hệ thống phải có cách khóa khối đó hoặc thay thế nó một cách hợp lý bằng một khối dự phòng.

Summary

- Do dung lượng lưu trữ cần thiết trên các hệ thống lớn và do các thiết bị lưu trữ bị lỗi theo nhiều cách khác nhau nên các thiết bị lưu trữ thứ cấp thường bị dư thừa thông qua thuật toán RAID. Các thuật toán này cho phép sử dụng nhiều ổ đĩa cho một hoạt động nhất định và cho phép hoạt động liên tục và thậm chí tự động khôi phục khi ổ đĩa bị lỗi. Các thuật toán RAID được tổ chức thành các cấp độ khác nhau; mỗi cấp độ cung cấp một số sự kết hợp giữa độ tin cậy và tốc độ truyền tải cao.

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- **Hệ thống Vào/ Ra (I/O system)**
  - Tổng quan
  - I/O Hardware
  - Application I/O Interface
  - Kernel I/O Subsystem
  - Transforming I/O requests to hardware operations
  - STREAMS
  - Hiệu suất
  - Summary

Cần nắm

- Khám phá cấu trúc của hệ thống con I/O (I/O subsystem) của OS
- Thảo luận về nguyên tắc và độ phức tạp của phần cứng I/O
- Giải thích các khía cạnh hiệu suất của phần cứng và phần mềm I/O

Tổng quan

- Các thiết bị I/O khác nhau về chức năng và tốc độ (như chuột, ổ cứng, ổ đĩa flash, ...) nên cần có nhiều phương pháp khác nhau để điều khiển chúng → Các phương thức này tạo thành 1 hệ thống con I/O của kernel
- Công nghệ sản xuất thiết bị I/O có 2 xu hướng trái ngược:
  - Sự tiêu chuẩn hóa ngày càng tăng → giúp kết hợp các thế hệ thiết bị cải tiến vào các máy tính và OS hiện có
  - Số loại thiết bị I/O ngày càng tăng → thiết bị mới không giống thiết bị trước đó nên việc kết hợp là một thách thức

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- Hệ thống Vào/ Ra (I/O system)
  - Tổng quan
  - **I/O Hardware**
  - Application I/O Interface
  - Kernel I/O Subsystem
  - Transforming I/O requests to hardware operations
  - STREAMS
  - Hiệu suất
  - Summary

Phần cứng I/O

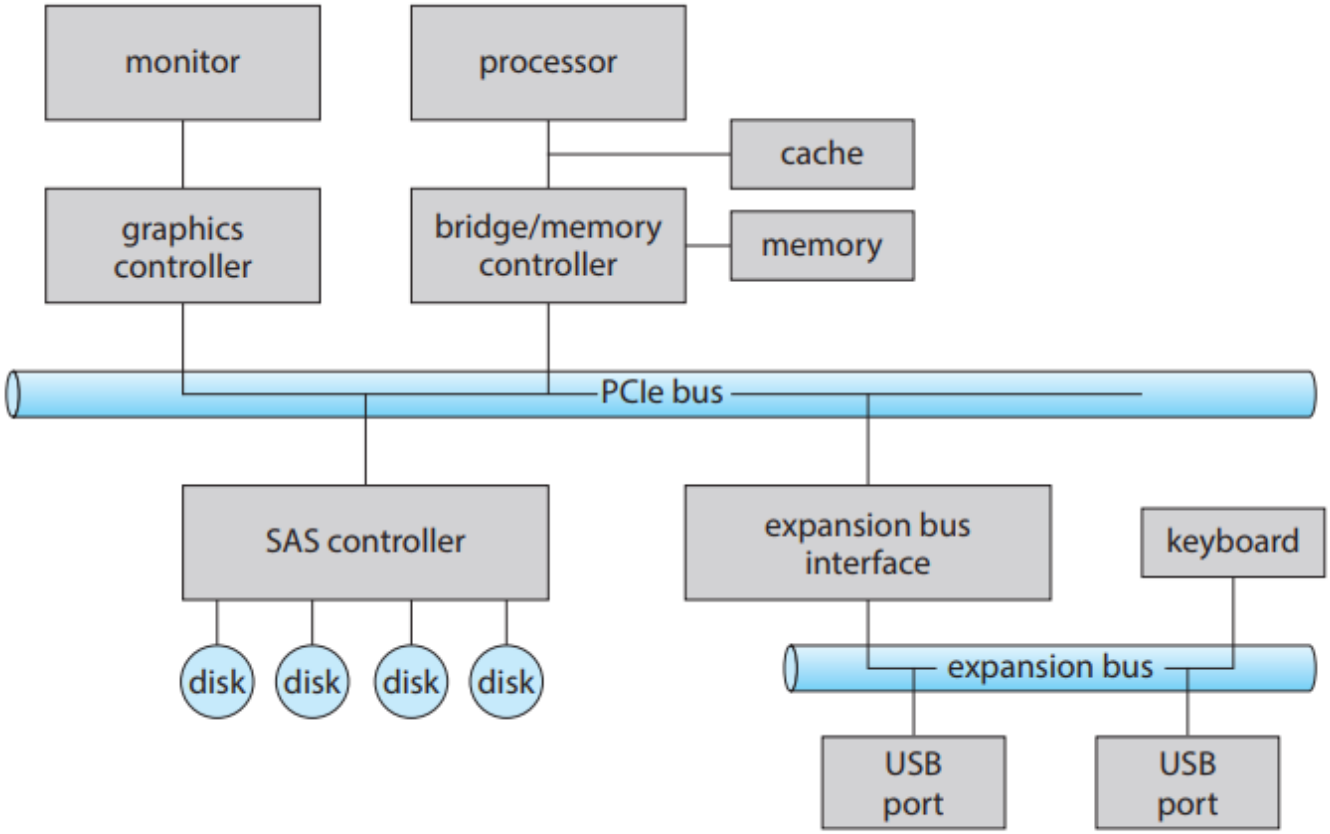


Figure 12.1 A typical PC bus structure.

Phần cứng I/O

- Máy tính vận hành rất nhiều loại thiết bị
- Thiết bị giao tiếp với máy tính thông qua điểm kết nối hoặc cổng (port) – VD cổng nối tiếp (serial port).
- Nếu thiết bị dùng chung bộ dây thì kết nối được gọi là bus
- Bộ điều khiển (controller) là tập hợp các thiết bị điện tử có thể vận hành port, bus hoặc thiết bị

Nội dung Phần cứng I/O

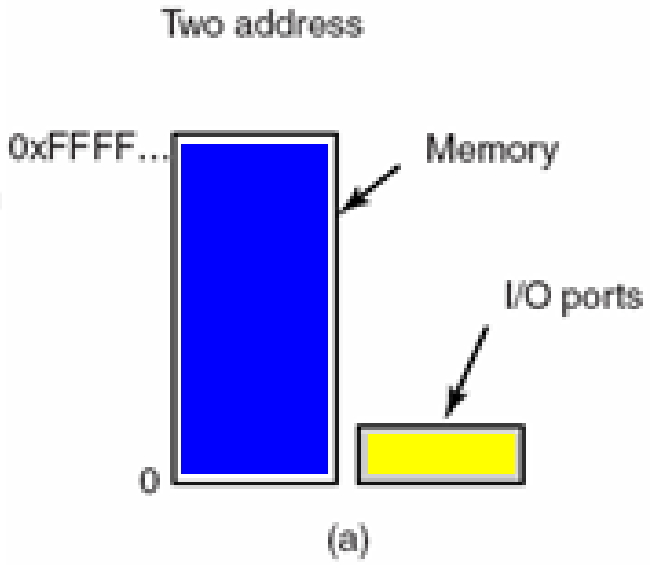
- Memory-Mapped I/O
- Polling
- Interrupts
- Direct Memory Access (DMA)
- I/O Hardware Summary

Memory-Mapped I/O

- Bộ xử lý cung cấp lệnh và dữ liệu cho bộ điều khiển như thế nào để thực hiện quá trình truyền I/O?
  - Bộ điều khiển có 1 hoặc nhiều thanh ghi dữ liệu và tín hiệu điều khiển
  - Bộ xử lý giao tiếp với bộ điều khiển bằng cách đọc và ghi các mẫu bit trong các thanh ghi này
  - Nhiều thiết bị có data buffer cho phép đọc và ghi
- Bộ xử lý liên lạc với thanh ghi điều khiển và data buffer của thiết bị như thế nào?
  - 3 cách

Memory-Mapped I/O

- Cách tiếp cận 1
  - Mỗi thanh ghi điều khiển được gán cho 1 I/O port number, là số nguyên 8 hoặc 16 bit
  - Tập tất cả các I/O port tạo thành 1 không gian I/O port và được bảo vệ
  - CPU có thể đọc nội dung từ PORT tới REG (thanh ghi trong CPU) và có thể ghi nội dung từ REG tới PORT
  - Không gian địa chỉ của memory và I/O khác nhau



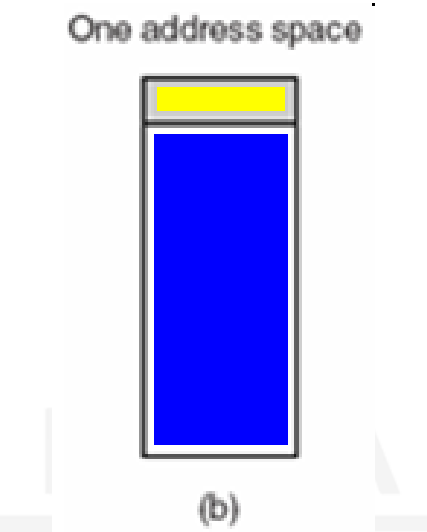
Memory-Mapped I/O

- Cách tiếp cận 1
  - Device I/O port locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

Memory-Mapped I/O

- Cách tiếp cận 2
  - Ánh xạ tất cả thanh ghi điều khiển vào không gian bộ nhớ
  - Mỗi thanh ghi điều khiển được gán cho một không gian bộ nhớ cụ thể và duy nhất (thường sẽ gán vào đầu không gian địa chỉ)
- Cách tiếp cận kết hợp
  - Bộ đệm dữ liệu (data buffer) I/O được ánh xạ theo bộ nhớ và các cổng I/O riêng biệt cho các thanh ghi điều khiển



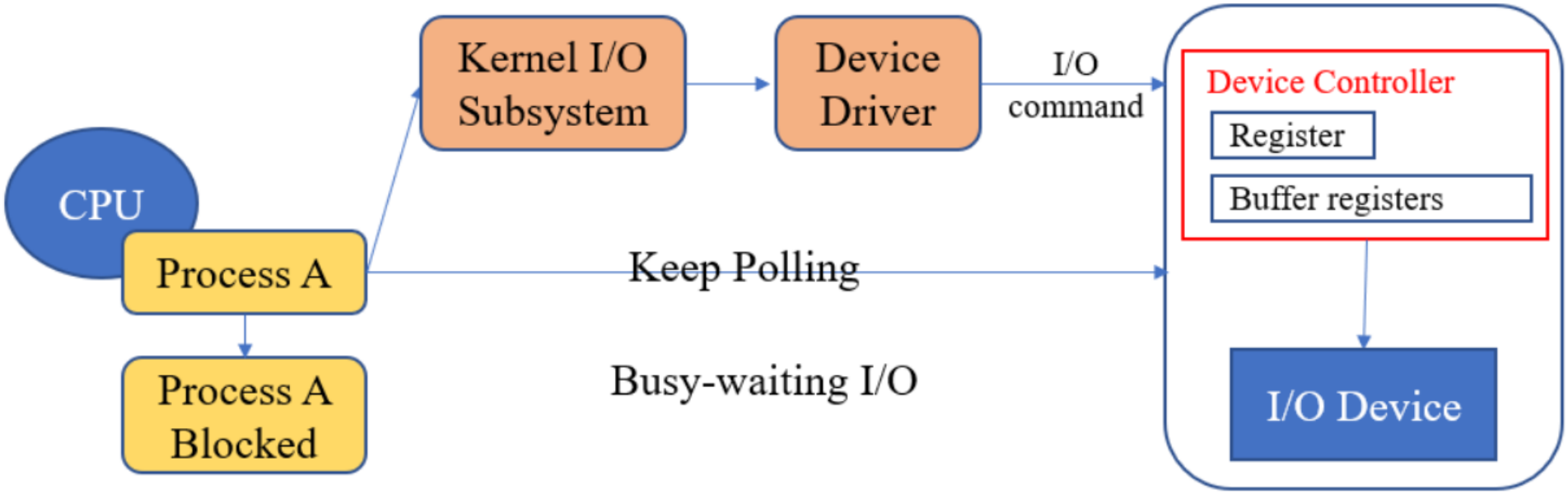
Memory-Mapped I/O

- Ưu điểm:
  - Trình điều khiển thiết bị I/O (I/O driver) có thể viết toàn bộ bằng C thay vì bằng hợp ngữ
  - Không cần có cơ chế bảo vệ đặc biệt nào để ngăn quá trình người dùng thực hiện I/O
  - Mọi lệnh có thể tham chiếu bộ nhớ cũng có thể tham chiếu các thanh ghi điều khiển (sử dụng 1 lệnh thay vì 2)

Memory-Mapped I/O

- Nhược điểm:
  - Việc lưu vào bộ đệm của 1 thanh ghi điều khiển thiết bị sẽ rất tai hại – tham chiếu từ cache thay vì thiết bị → không rõ ràng do có nhiều thiết bị I/O → có thể vô hiệu quá cache → nhưng chậm
  - Các thiết bị I/O không có cách nào nhìn thấy các địa chỉ bộ nhớ khi chúng di chuyển trên bus bộ nhớ, vì vậy chúng ko có cách nào phản hồi tự động
  - Khi khởi động hệ thống, phải thêm bước kiểm tra xem địa chỉ bộ nhớ nào không thực sự là địa chỉ bộ nhớ

Polling


Kernel I/O Subsystem
Device Driver
I/O command
Device Controller
Register
Buffer registers
CPU
Process A
Keep Polling
Process A Blocked
Busy-waiting I/O
I/O Device

Polling

- Host ghi output thông qua port, phối hợp với controller bằng cách bắt tay như sau:
  1. Host liên tục đọc bit busy cho đến khi nó được clear
  2. Host set bit write vào thanh ghi lệnh và ghi 1 byte vào thanh ghi dữ liệu ra
  3. Host set bit command-ready
  4. Khi controller thông báo rằng bit command-ready đã được set thì set bit busy
  5. Controller đọc thanh thi lệnh và nhìn thấy lệnh ghi. Nó đọc thanh ghi xuất dữ liệu để lấy byte và thực hiện I/O cho thiết bị
  6. Controller clear bit command-ready, clear bit eror trong thanh ghi trạng thái để cho biết rằng I/O của thiết bị thành công và clear bit busy để cho biết rằng nó đã hoàn thành

Interrupts

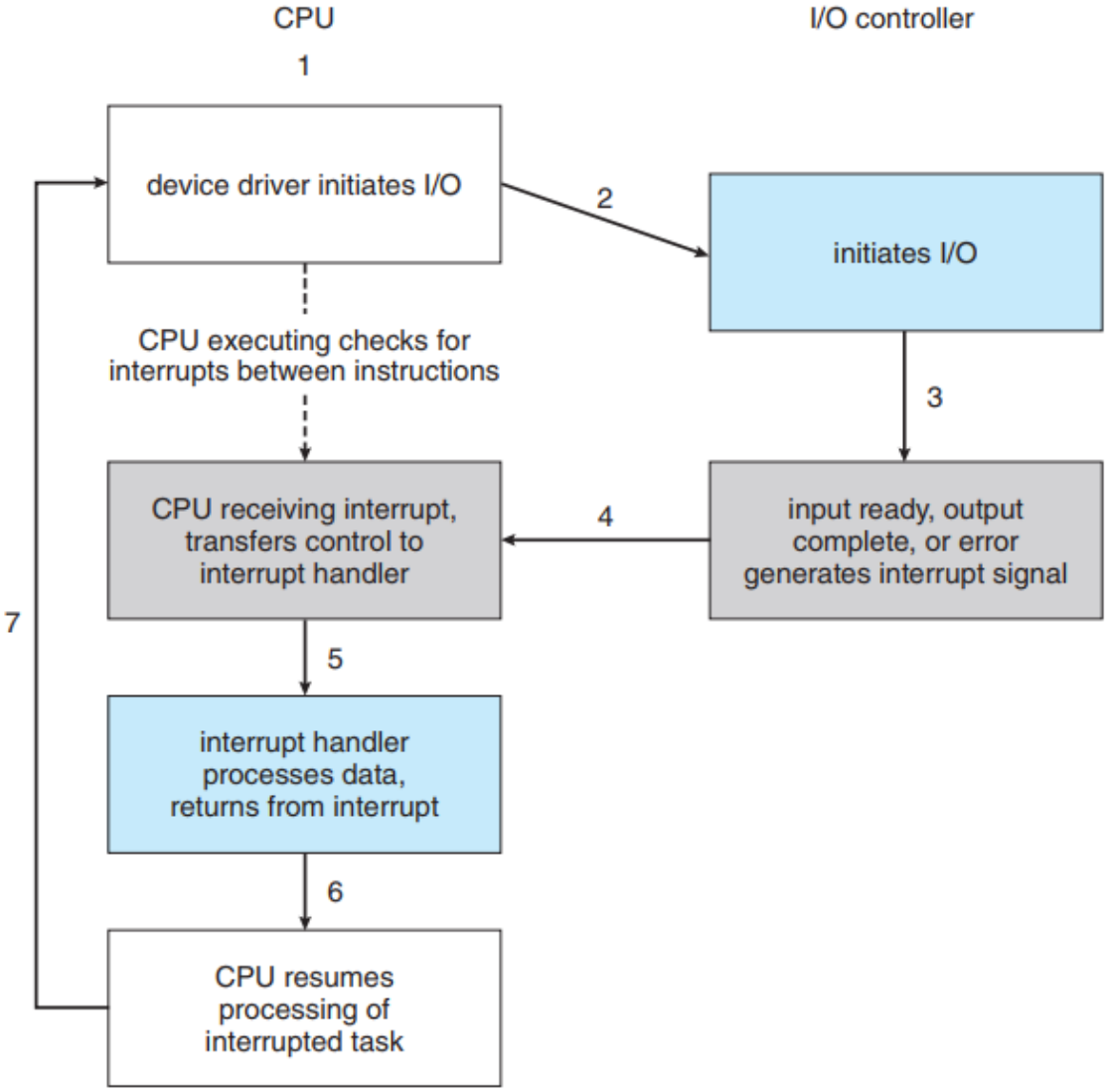


Figure 12.3 Interrupt-driven I/O cycle.

Interrupts

- Phần cứng CPU có một dây gọi là đường yêu cầu ngắt mà CPU cảm nhận được sau khi thực hiện mọi lệnh.
- Khi CPU phát hiện bộ điều khiển đã xác nhận tín hiệu trên dòng yêu cầu ngắt, CPU sẽ thực hiện lưu trạng thái và chuyển sang quy trình xử lý ngắt tại một địa chỉ cố định trong bộ nhớ.
- Trình xử lý ngắt xác định nguyên nhân gây ra ngắt, thực hiện xử lý cần thiết, thực hiện khôi phục trạng thái và thực hiện lệnh quay về từ lệnh ngắt để đưa CPU về trạng thái thực thi trước khi ngắt.

Interrupts

```
Fri Nov 25 13:55:59                                    0:00:10
                        SCHEDULER      INTERRUPTS
---------------------------------------------------------------
total_samples                13           22998

delays <  10 usecs           12           16243
delays <  20 usecs            1            5312
delays <  30 usecs            0             473
delays <  40 usecs            0             590
delays <  50 usecs            0              61
delays <  60 usecs            0             317
delays <  70 usecs            0               2
delays <  80 usecs            0               0
delays <  90 usecs            0               0
delays < 100 usecs            0               0
total  < 100 usecs           13           22998
```

Interrupts

- Cần khả năng trì hoãn việc xử lý ngắt trong quá trình xử lý quan trọng
- Cần 1 cách hiệu quả để gửi đến bộ xử lý ngắt thích hợp cho 1 thiết bị mà không cần polling của tất cả các thiết bị trước để xem thiết bị nào gây ra ngắt
- Cần multilevel interrups để OS có thể phân biệt giữa các ngắt có mức ưu tiên cao và thấp và có thể phản hồi với mức độ khẩn cấp thích hợp khi có nhiều ngắt đồng thời
- Cần 1 cách để lệnh thu hút sự chú ý trực tiếp của OS (tách biệt với các yêu cầu I/O), đối với các hoạt động như page fault, lỗi chia cho 0, ... Nhưng task này được thực hiện bằng trap

Trong phần cứng máy tính hiện đại, những tính năng này được cung cấp bởi CPU và phần cứng bộ điều khiển ngắt (interrupt-controller hardware).

Interrupts

- Hầu hết CPUs đều có 2 dòng yêu cầu ngắt.
  - Không thể che dấu, dành riêng cho các sự kiện như lỗi bộ nhớ không thể phục hồi
  - Có thể che dấu, CPU có thể tắt nó trước khi thực hiện các chuỗi lệnh quan trọng không được ngắt – được sử dụng bởi bộ điều khiển thiết bị để yêu cầu dịch vụ

DMA (Direct Memory Access)

- Đối với thiết bị truyền tải lớn như ổ đĩa, sử dụng bộ vi xử lý đa năng đắt tiền để xem trạng thái bit, nạp dữ liệu vào bộ điều khiển register từng byte một tại mỗi thời điểm (gọi là PIO process termed programmed I/O) là lãng phí
- Máy tính tránh tạo gánh nặng PIO cho CPU bằng cách chuyển số công việc này sang cho bộ xử lý có mục đích đặc biệt gọi là DMA (Direct memory access) – bộ điều khiển truy cập bộ nhớ trực tiếp

DMA (Direct Memory Access)

- Để bắt đầu truyền DMA, host ghi khối lệnh DMA vào bộ nhớ. Khối này chứa 1 con trỏ tới nguồn truyền, 1 con trỏ tới đích và số byte được truyền
- CPU ghi địa chỉ của khối lệnh này vào bộ điều khiển DMA, sau đó tiếp tục công việc khác
- Bộ điều khiển DMA tiến hành vận hành trực tiếp bus bộ nhớ, đặt địa chỉ trên bus để thực hiện truyền mà không cần sự trợ giúp của CPU
- Bộ điều khiển DMA là thành phần tiêu chuẩn trong tất cả các máy tính hiện đại, điện thoại thông minh, ...

DMA

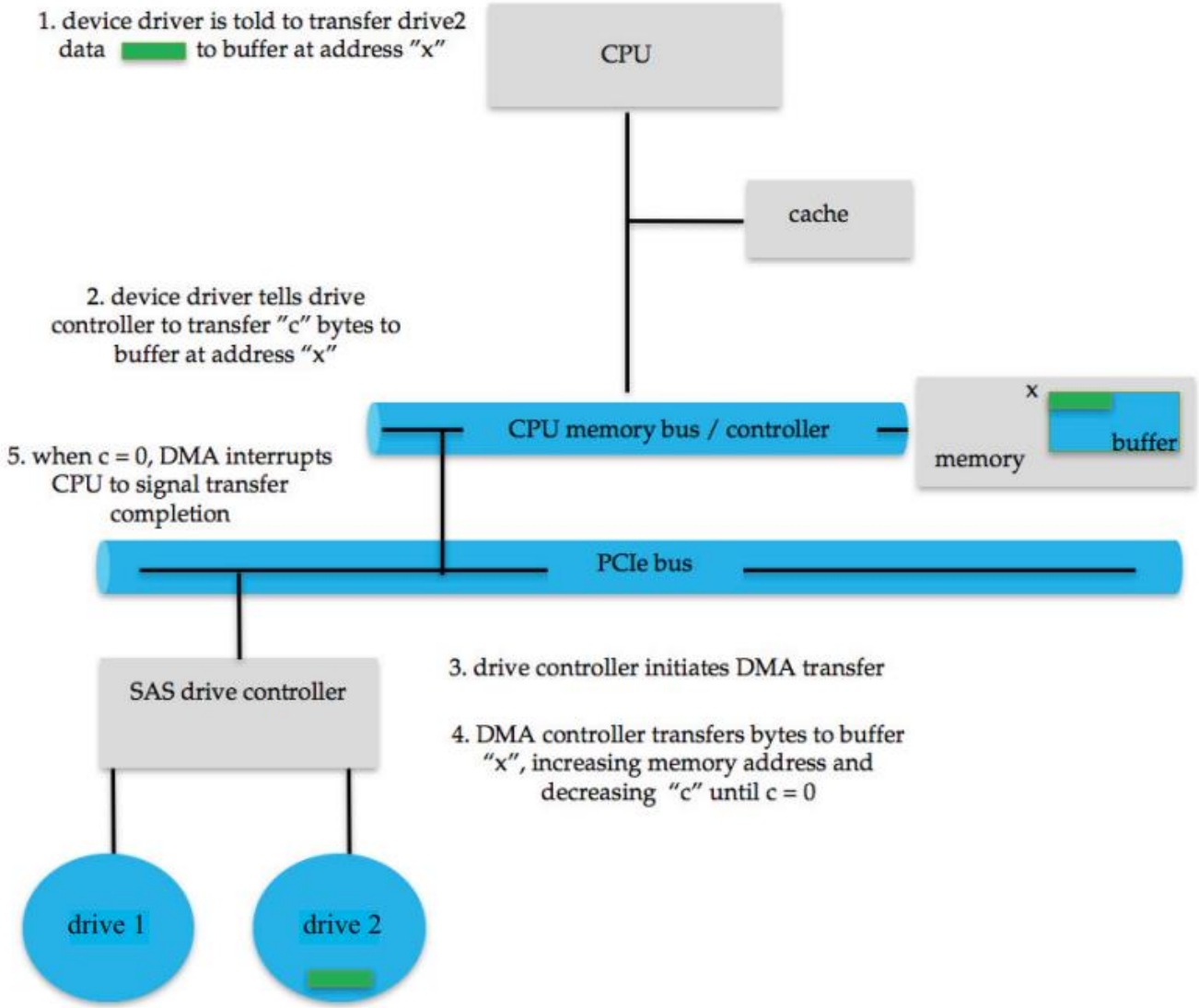


Figure 12.6 Steps in a DMA transfer.

DMA (Direct Memory Access)

- Quá trình bắt tay giữa bộ điều khiển DMA và bộ điều khiển thiết bị được thực hiện thông qua một cặp dây gọi là DMA-request và DMA-acknowledge.
- Bộ điều khiển thiết bị đặt tín hiệu trên dây DMA-request khi có một word dữ liệu để truyền. Tin hiệu này làm cho bộ điều khiển DMA chiếm giữ bus bộ nhớ, đặt địa chỉ mong muốn trên dây memory-address, và đặt tín hiệu trên dây DMA-acknowledge
- Khi bộ điều khiển thiết bị nhận được tín hiệu DMA-acknowledge, nó sẽ chuyển từ dữ liệu vào bộ nhớ và loại bỏ tín hiệu DMA-request
- Khi toàn bộ quá trình truyền kết thúc, bộ điều khiển DMA sẽ ngắt CPU.

I/O Hardware Summary

- Xem lại các khái niệm:
  - Bus
  - Bộ điều khiển (controller)
  - I/O port và register của chúng
  - Mỗi quan hệ bắt tay giữa host và bộ điều khiển thiết bị
  - Việc thực hiện bắt tay trong polling và interrupt
  - Việc chuyển việc sang cho DMA để truyền số lượng lớn dữ liệu

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- Hệ thống Vào/ Ra (I/O system)
  - Tổng quan
  - I/O Hardware
  - **Application I/O Interface**
  - Kernel I/O Subsystem
  - Transforming I/O requests to hardware operations
  - STREAMS
  - Hiệu suất
  - Summary

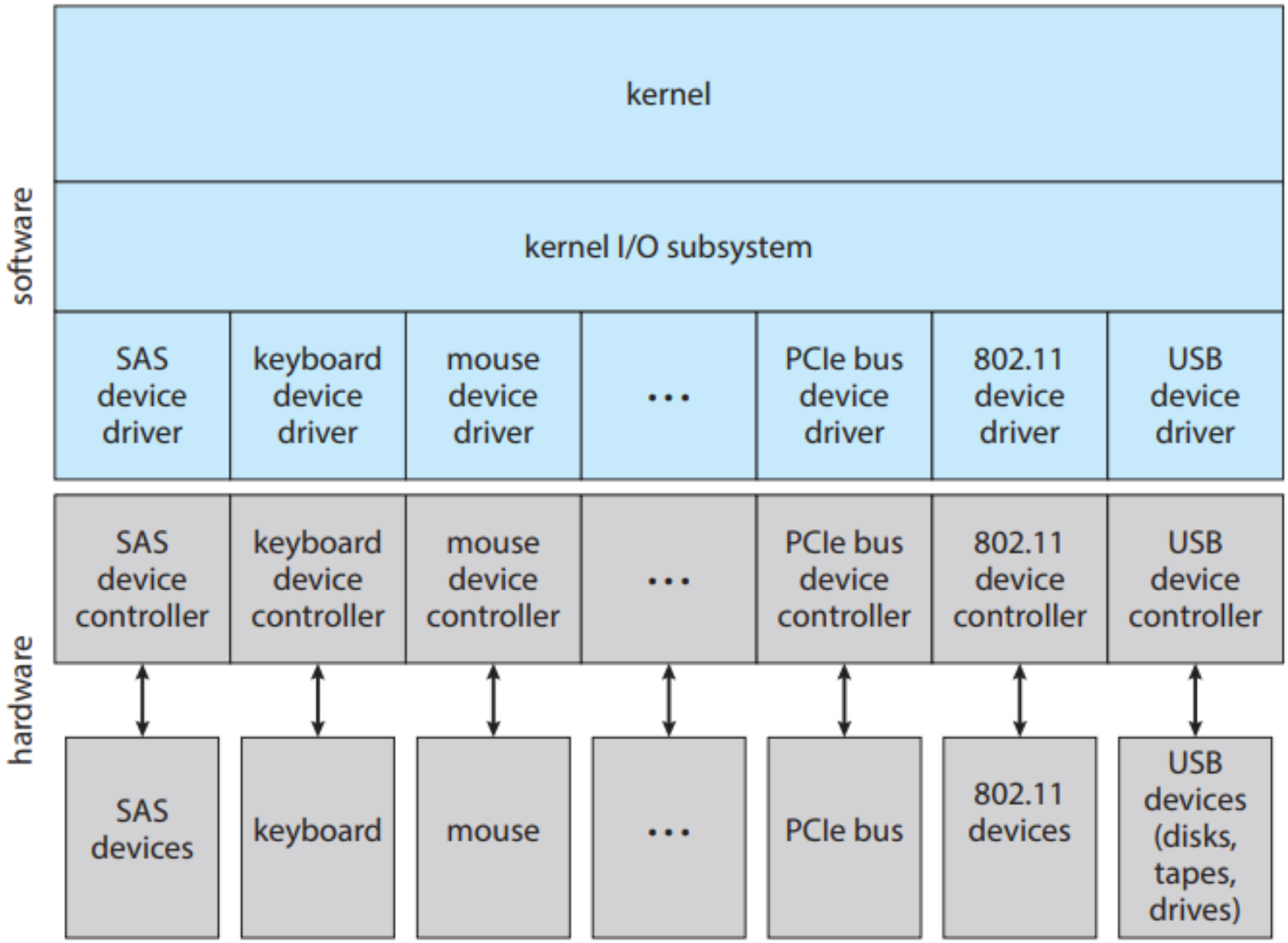


Figure 12.7 A kernel I/O structure.

Application I/O Interface

- Mục đích chính của lớp device-driver là che giấu sự khác biệt giữa các bộ điều khiển thiết bị khỏi hệ thống con I/O của kernel
- Ưu điểm:
  - các hệ thống con I/O độc lập với phần cứng giúp đơn giản hóa công việc của nhà phát triển OS
  - Nhà sản xuất phần cứng thiết kế các thiết bị mới tương thích với controller interface mà host hiện có hoặc viết device driver để giao tiếp với phần cứng mới cho các OS phổ biến
  - Có thể gắn các thiết bị ngoại vi mới vào máy tính mà không cần đợi nhà cung cấp OS phát triển mã hỗ trợ

Application I/O Interface

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Figure 12.8 Characteristics of I/O devices.

Application I/O Interface

- Character-stream hoặc block: Thiết bị luồng ký tự truyền từng byte một, trong khi thiết bị khối truyền một khối byte dưới dạng đơn vị.
- Sequential hoặc random access: Thiết bị tuần tự truyền dữ liệu theo thứ tự cố định do thiết bị xác định, trong khi người dùng thiết bị truy cập ngẫu nhiên có thể hướng dẫn thiết bị tìm kiếm bất kỳ vị trí lưu trữ dữ liệu có sẵn nào.
- Synchronous hoặc asynchronous: Một thiết bị đồng bộ thực hiện truyền dữ liệu với thời gian phản hồi có thể dự đoán được, phối hợp với các khía cạnh khác của hệ thống. Thiết bị không đồng bộ thể hiện thời gian phản hồi không đều hoặc không thể đoán trước, không phối hợp với các sự kiện máy tính khác.

Application I/O Interface

- Sharable hoặc dedicated: Một thiết bị có thể chia sẻ có thể được sử dụng đồng thời bởi một số tiến trình hoặc luồng; một thiết bị chuyên dụng không thể.
- Speed of operation: Tốc độ thiết bị dao động từ vài byte mỗi giây đến gigabyte mỗi giây.
- Read-write, read only, write once: Một số thiết bị thực hiện cả đầu vào và đầu ra, nhưng một số khác chỉ hỗ trợ một hướng truyền dữ liệu. Một số cho phép sửa đổi dữ liệu sau khi ghi, nhưng một số khác chỉ có thể được ghi một lần và ở chế độ chỉ đọc sau đó.

Application I/O Interface

- Thiết bị block và character
- Thiết bị mạng
- Clocks và Timers
- Nonblocking và Asynchronous I/O
- Vectored I/O

Thiết bị block và character

- Thiết bị block
  - Phải hiểu được các lệnh như read(), write()
  - Nếu nó là 1 thiết bị truy cập ngẫu nhiên, nó phải có lệnh seek().
- Thiết bị character
  - Phải hiểu được các lệnh get(), put() một ký tự.
  - Các thư viện có thể được xây dựng để cung cấp khả năng truy cập theo từng dòng, với các dịch vụ đệm và chỉnh sửa (VD người dùng nhập phím backspace, ký tự trước đó sẽ bị xóa khỏi luồng input)

Thiết bị mạng

- Đối với I/O mạng, OS cung cấp network socket interface (có trên Windows và Linux)
- System call trong socket interface cho phép 1 ứng dụng tạo socket, kết nối local socket tới 1 remote address
- Để hỗ trợ việc triển khai các network server, socket interface cung cấp hàm select() để quản lý một tập hợp các socket – hàm select() trả về thông tin socket nào có gói tin đang chờ nhận và socket nào có không gian nhận gói.
- Nhiều cách tiếp cận khác để giao tiếp giữa các tiến trình và giao tiếp mạng đã được triển khai.
  - VD Windows cung cấp một interface cho network interface card và interface thứ 2 cho các giao thức mạng.

Clocks và Timers

- Nhiều máy tính có phần cứng clock và timer cung cấp 3 chức năng cơ bản:
  - Cho biết thời gian hiện tại
  - Cho biết thời gian đã trôi qua
  - Đặt hẹn giờ để kích hoạt thao tác X tại thời điểm T
- Các lệnh system call triển khai các chức năng này không được chuẩn hóa trên các OS.
- Phần cứng để đo thời gian đã trôi qua và kích hoạt các hoạt động được gọi là bộ định thời khoảng thời gian có thể lập trình.

Nonblocking và Asynchronous I/O

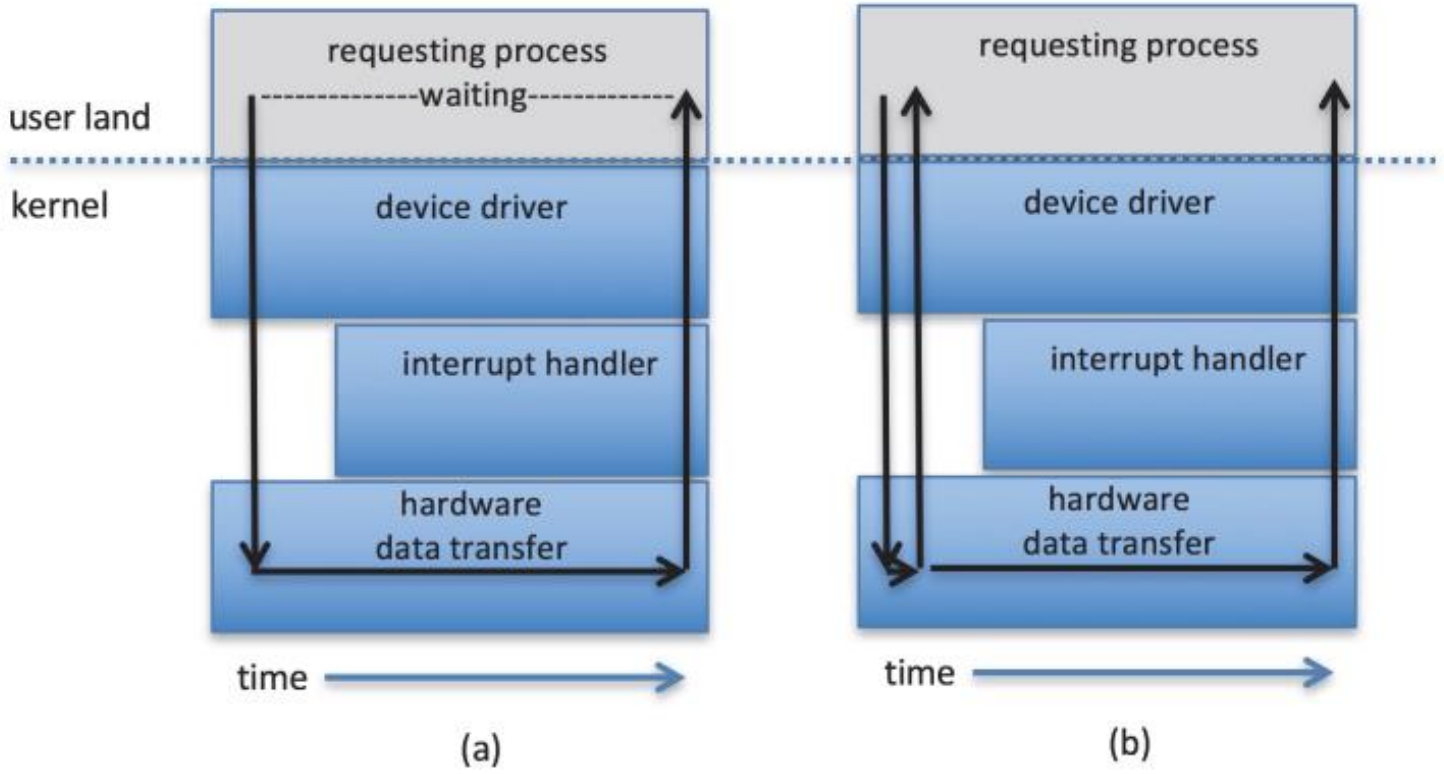


Figure 12.9 Two I/O methods: (a) synchronous and (b) asynchronous.

Vectored I/O

- Vectored I/O cho phéo 1 system call thực hiện nhiều thao tác I/O liên quan đến nhiều vị trí
- VD: lệnh system call readv trong UNIX chấp nhận 1 vector gồm nhiều bộ đệm và đọc từ nguồn tới vector đó hoặc ghi từ vector đến đích.
- Lý do:
  - Nhiều bộ đệm riêng biệt có thể chuyển nội dung của chúng qua 1 system call, tránh việc switch context và chi phí system call.
  - Nếu không có vectored I/O, dữ liệu cần được chuyển sang bộ đệm lớn hơn theo đúng thứ tự rồi truyền đi → không hiệu quả

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- Hệ thống Vào/ Ra (I/O system)
  - Tổng quan
  - I/O Hardware
  - Application I/O Interface
  - **Kernel I/O Subsystem**
  - Transforming I/O requests to hardware operations
  - STREAMS
  - Hiệu suất
  - Summary

Kernel I/O Subsystem

- I/O Scheduling
- Buffering
- Caching
- Spooling and Device Reservation
- Error Handling
- I/O Protection
- Kernel Data Structures
- Power Management
- Kernel I/O Subsystem Summary

I/O Scheduling

- OS thực hiện lập lịch I/O bằng cách duy trì hàng đợi yêu cầu cho mỗi thiết bị
- Khi 1 ứng dụng đưa ra system call blocking I/O, yêu cầu được đưa vào hàng đợi cho thiết bị đó. Bộ lập lịch sắp xếp lại thứ tự của hàng đợi để cải thiện hiệu quả tổng thể của hệ thống và thời gian phản hồi trung bình mà các ứng dụng gặp phải
- OS cố gắng công bằng để không ứng dụng nào nhận dịch vụ đặc biệt kém hoặc có thể cung cấp dịch vụ ưu tiên cho các yêu cầu nhạy cảm với deadline
  - VD các yêu cầu từ hệ thống con bộ nhớ ảo có thể được ưu tiên hơi các yêu cầu ứng dụng
- Khi 1 kernel hỗ trợ I/O asynchronous, nó phải có khả năng theo dõi nhiều yêu cầu I/O cùng 1 lúc.

I/O Scheduling

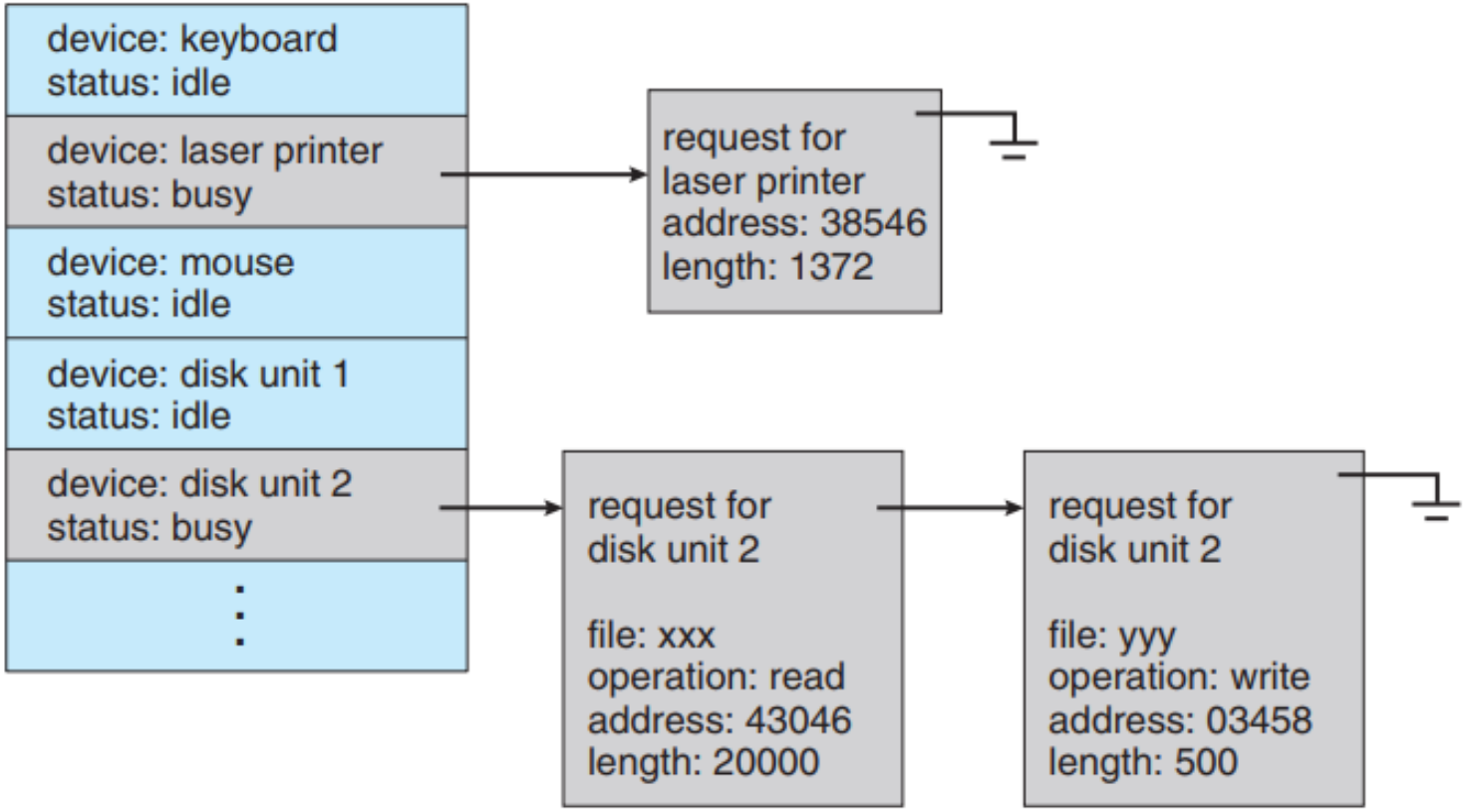


Figure 12.10 Device-status table.

Buffering

- Buffer là vùng nhớ lưu trữ dữ liệu truyền tải giữa 2 thiết bị hoặc giữa 1 thiết bị và 1 ứng dụng.
- Lý do sử dụng buffer:
  - Đáp ứng sự không khớp tốc độ giữa producer và consumer luồng dữ liệu
  - Cung cấp khả năng thích ứng cho các thiết bị có kích thước truyền dữ liệu khác nhau
  - Hỗ trợ ngữ nghĩa sao chép cho I/O ứng dụng

Buffering

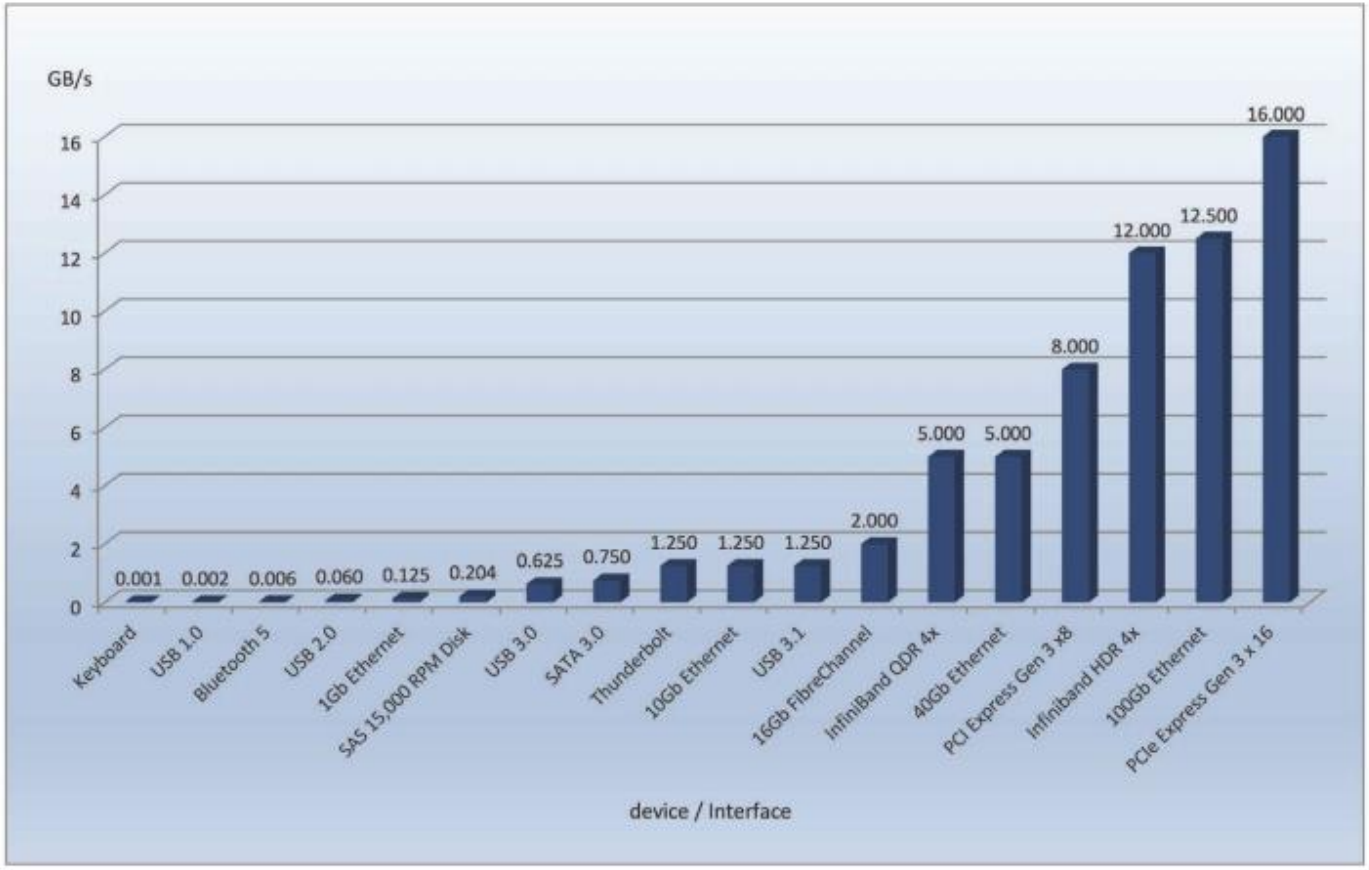


Figure 12.11 Common PC and data-center I/O device and interface speeds.

Caching

- Caching là vùng bộ nhớ nhanh chứa các bản sao dữ liệu
- Truy cập vào bản sao được lưu trữ trong cache hiệu quả hơn truy cập vào bản gốc
- Sự khác biệt giữa buffer và cache là buffer có thể giữ bản sao duy nhất hiện có của dữ liệu, trong khi đó, cache giữ 1 bản sao trên bộ lưu trữ nhanh hơn của dữ liệu nằm ở nơi khác
- Cache và buffer có các chức năng riêng biệt, nhưng đôi khi 1 vùng bộ nhớ có thể được sử dụng cho cả 2 mục đích.

Spooling và Device Reservation

- Spool là buffer chứa output của 1 thiết bị, như máy in, không thể chấp nhận các luồng dữ liệu xen kẽ
- Đầu ra của mỗi ứng dụng được lưu vào 1 file lưu trữ phụ riêng biệt. Khi một ứng dụng in xong, hệ thống spooling sẽ xếp hàng file spool tương ứng để xuất ra máy in

Error Handling

- OS sử dụng bộ nhớ được bảo vệ (protected memory) có thể bảo vệ khỏi nhiều loại lỗi phần cứng và ứng dụng
- Thiết bị và quá trình truyền I/O có thể lỗi/ thất bại theo nhiều cách
  - VD mạng quá tải hoặc bộ điều khiển đĩa bị lỗi
- OS thường có thể bù đắp hiệu quả cho những lỗi nhất thời
  - VD lỗi read() trên đĩa thì thử lại read(), lỗi send() trên mạng thì thử lại resend()
- Theo nguyên tắc chung, system call I/O sẽ trả về 1 bit thông tin liên quan đến trạng thái nó là thành công hay thất bại. Trong UNIX, 1 biến số nguyên bổ sung errno được sử dụng để trả về mã lỗi (khoảng 100 giá trị, cho biết lỗi gì)

I/O protection

- Lỗi có liên quan chặt chẽ đến vấn đề bảo vệ
- 1 tiến trình người dùng có thể vô tình hoặc cố ý làm gián đoạn hoạt động bình thường của hệ thống bằng cách cố gắng đưa ra các lệnh I/O bất hợp pháp
- Có nhiều cơ chế khác nhau để đảm bảo rằng những gián đoạn không thể xảy ra trong hệ thống:
  - Xác định tất cả các lệnh I/O đều là lệnh đặc quyền → user không thể trực tiếp đưa ra lệnh I/O; phải làm thông qua OS
  - Mọi vị trí bộ nhớ được ánh xạ (memory-mapped) và vị trí bộ nhớ port I/O phải được bảo vệ khỏi sự truy cập của user bằng hệ thống memory-protection

I/O protection

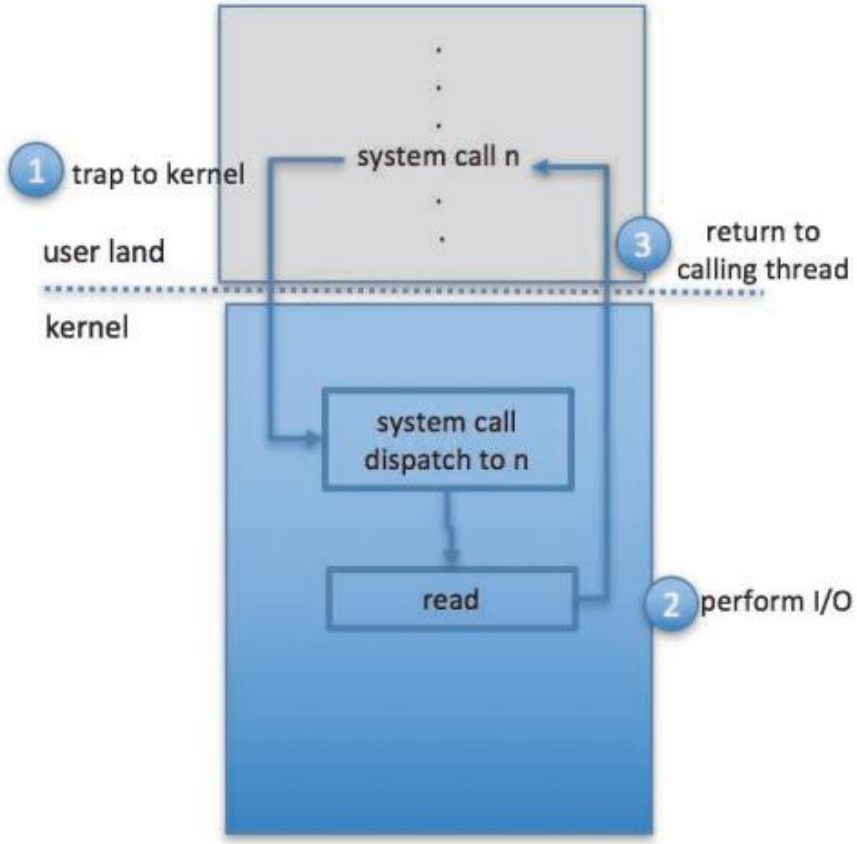


Figure 12.12 Use of a system call to perform I/O.

Cấu trúc dữ liệu Kernel

- Kernel cần lưu trữ thông tin trạng thái về việc sử dụng các thành phần I/O
  - Thông qua nhiều CTDL trong kernel
- Kernel sử dụng nhiều cấu trúc tương tự để theo dõi các kết nối mạng, liên lạc giữa ký tự và thiết bị và các hoạt động I/O khác
- Một số OS sử dụng các phương pháp hướng đối tượng
  - VD Windows sử dụng message-passing cho I/O

Cấu trúc dữ liệu Kernel

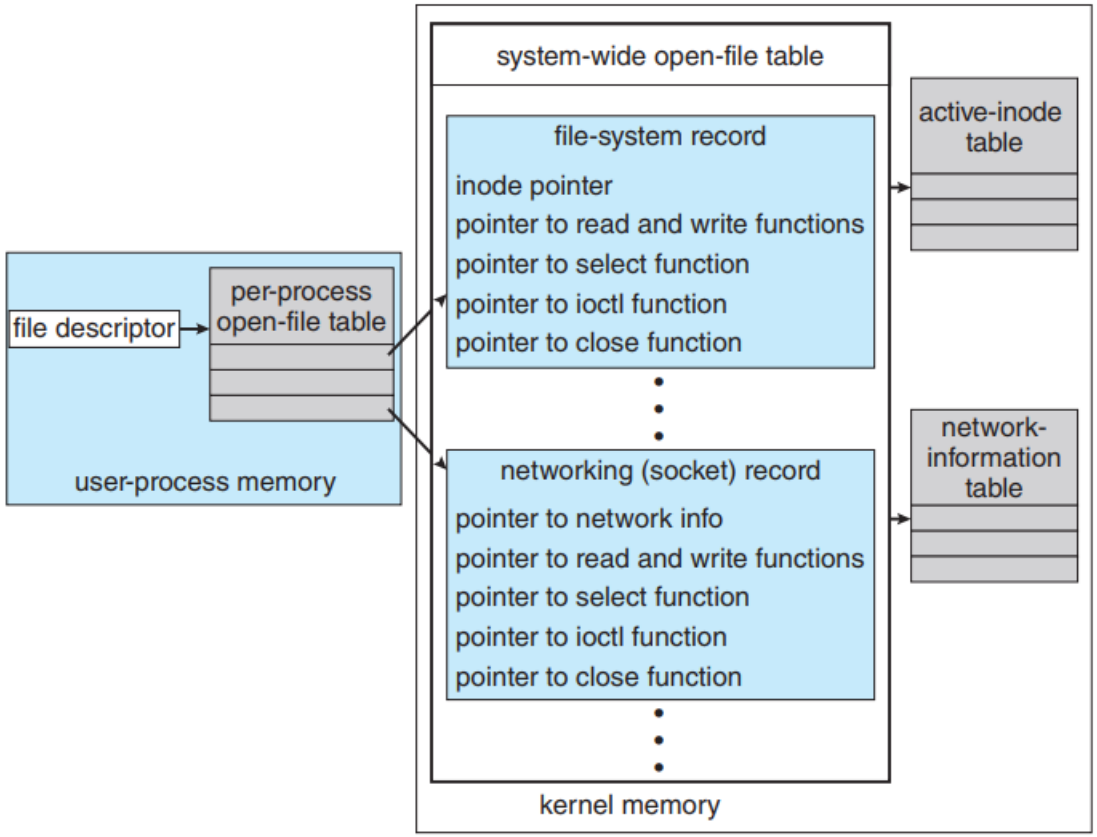


Figure 12.13 UNIX I/O kernel structure.

Power management

- OS đóng vai trò trong việc sử dụng năng lượng
  - Tạo ra nhiệt
  - Làm át
- OS có thể phân tích tải của hệ thống và tắt nguồn các thành phần như CPU và thiết bị I/O bên ngoài
- Các máy tính hiện đại đa năng sử dụng ACPI (advanced configuratio and power interface) để quản lý trạng thái và nguồn điện của thiết bị
  - ACPI là một tiêu chuẩn công nghiệp với nhiều tính năng
  - VD khi kernel cần tắt 1 thiết bị, kernel sẽ gọi driver thiết bị, driver này sẽ gọi ACPI và giao tiếp với thiết bị để tắt

Kernel I/O Subsystem Summary

- I/O subsystem điều phối 1 tập hợp các dịch vụ có sẵn cho ứng dụng và các phần khác của kernel
  - Quản lý không gian tên cho file và thiết bị
  - Kiểm soát quyền truy cập vào các file và thiết bị
  - Kiểm soát hoạt động (VD modem không thể seek())
  - Phân bổ không gian hệ thống tập tin
  - Phân bổ thiết bị
  - Buffering, caching và spooling
  - Lập lịch I/O
  - Giám sát trạng thái thiết bị, xử lý lỗi và khôi phục lỗi
  - Cấu hình và khởi tạo trình điều khiển thiết bị
  - Quản lý nguồn điện của thiết bị I/O

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- Hệ thống Vào/ Ra (I/O system)
  - Tổng quan
  - I/O Hardware
  - Application I/O Interface
  - Kernel I/O Subsystem
  - **Transforming I/O requests to hardware operations**
  - STREAMS
  - Hiệu suất
  - Summary

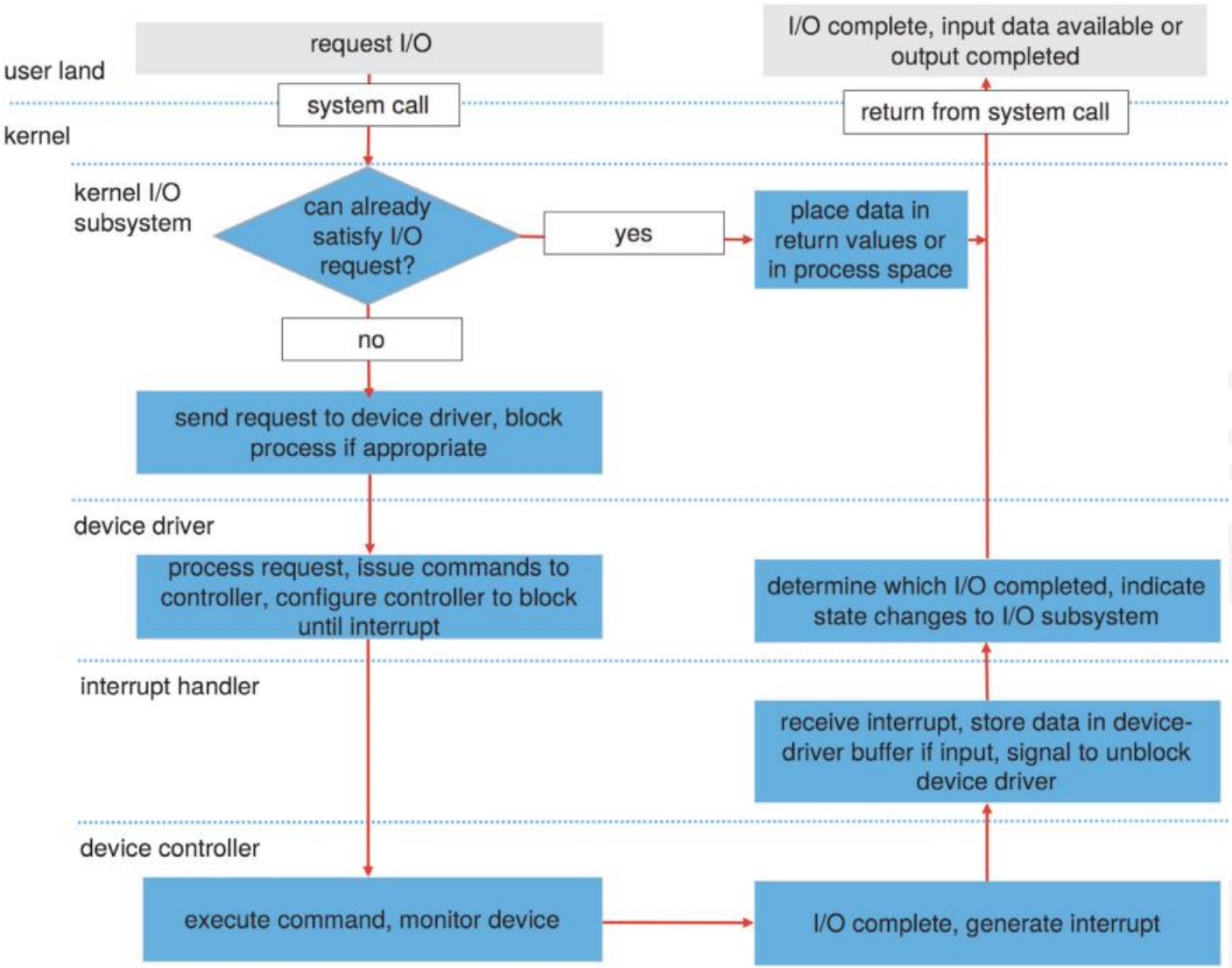


Figure 12.14 The life cycle of an I/O request.

Transforming I/O requests to hardware operations

1. 1 tiến trình đưa ra lệnh chặn system call read() tới 1 file descriptor của 1 file đã được mở trước đó
2. Code system-call trong kernel kiểm tra tính chính xác của các tham số. Trong trường hợp dữ liệu đã có sẵn trong buffer cache, dữ liệu sẽ được trả về tiến trình và yêu cầu I/O được hoàn thành
3. Nếu không, I/O vật lý phải thực hiện. Tiến trình sẽ chuyển sang trạng thái waiting, yêu cầu I/O được lập lịch. Cuối cùng, subsystem I/O gửi yêu cầu tới driver thiết bị. Tùy thuộc vào OS, yêu cầu được gửi thông qua subroutine call hoặc in-kernel message
4. Driver phân bổ không gian kernel buffer để nhận dữ liệu và lên lịch I/O. Sau đó, driver gửi lệnh đến controller bằng cách ghi vào các thanh ghi điều khiển thiết bị

Transforming I/O requests to hardware operations

5. Controller vận hành phần cứng của thiết bị để thực hiện truyền dữ liệu
6. Driver có thể poll trạng thái hoặc thiết lập DMA transfer trong kernel memory. Giả định rằng sử dụng DMA controller, nó sẽ tạo ra ngắt khi quá trình hoàn tất
7. Trình xử lý ngắt sẽ nhận ngắt thông qua bảng vector ngắt, lưu trữ mọi dữ liệu cần thiết, báo hiệu cho driver và trả về từ ngắt
8. Driver nhận tín hiệu, xác định yêu cầu I/O nào đã hoàn thành, xác định trạng thái của yêu cầu và báo hiệu cho kernel I/O subsystem rằng yêu cầu hoàn tất
9. Kernel truyền dữ liệu hoặc trả lại code vào không gian địa chỉ của tiến trình yêu cầu và chuyển tiến trình sang trạng thái ready
10. Khi tiến trình chạy, tiến trình sẽ tiếp tục thực thi hoàn thành system call

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- Hệ thống Vào/ Ra (I/O system)
  - Tổng quan
  - I/O Hardware
  - Application I/O Interface
  - Kernel I/O Subsystem
  - Transforming I/O requests to hardware operations
  - **STREAMS**
  - Hiệu suất
  - Summary

STREAMS

- Cơ chế STREAMS có trên UNIX System V (và nhiều bản phát hành UNIX tiếp theo) cho phép ứng dụng lắp ráp các đường dẫn mã trình điều khiển (pipelines of driver code) một cách linh hoạt
- Stream là 1 kết nối full-duplex giữa driver và tiến trình user-level. Bao gồm:
  - Stream head giao tiếp với tiến trình người dùng
  - Driver end điều khiển thiết bị
  - Không có hoặc có nhiều stream module giữa stream head và driver end
  - Mỗi thành phần chứa 1 cặp hàng đợi: hàng đợi read và hàng đợi write
- Message passing được sử dụng để truyền dữ liệu giữa các hàng đợi

STREAMS

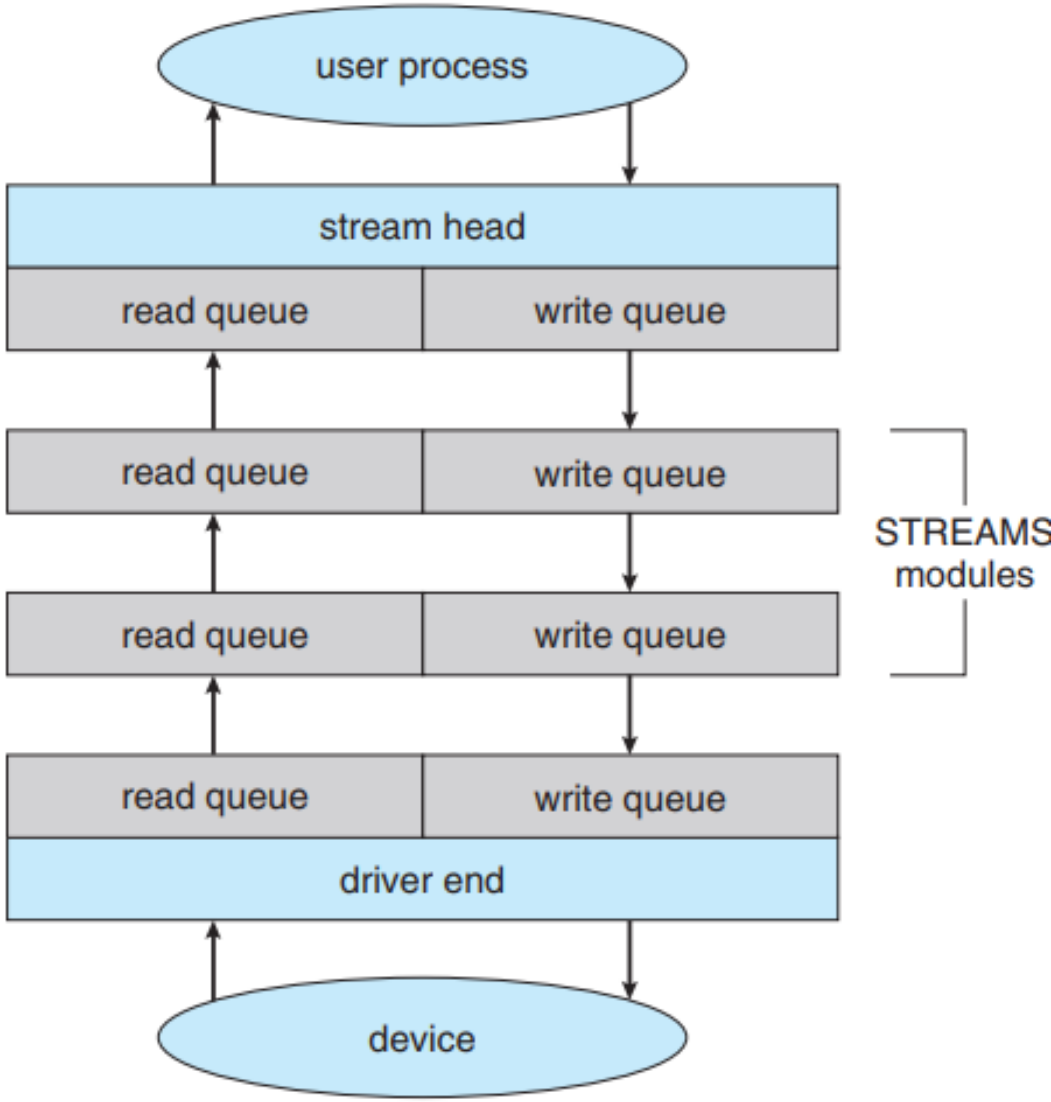


Figure 12.15 The STREAMS structure.

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- Hệ thống Vào/ Ra (I/O system)
  - Tổng quan
  - I/O Hardware
  - Application I/O Interface
  - Kernel I/O Subsystem
  - Transforming I/O requests to hardware operations
  - STREAMS
  - **Hiệu suất**
  - Summary

Hiệu suất

- I/O là yếu tố chính ảnh hưởng đến hiệu năng của hệ thống
- I/O đặt ra yêu cầu nặng nề cho CPU trong việc thực thi device-driver code và lập lịch tiến trình công bằng và hiệu quả khi chúng chặn hoặc bỏ chặn
  - Context switch gây stress cho CPU và hardware cache
- I/O kém hiệu quả trong cơ chế xử lý ngắt trong kernel
- I/O tải memory bus trong quá trình sao chép dữ liệu giữa controller và physical memory cũng như trong quá trình sao chép giữa kernel buffer và không gian dữ liệu ứng dụng

Hiệu suất

- Lưu lượng mạng cũng gây ra tỷ lệ context-switch cao

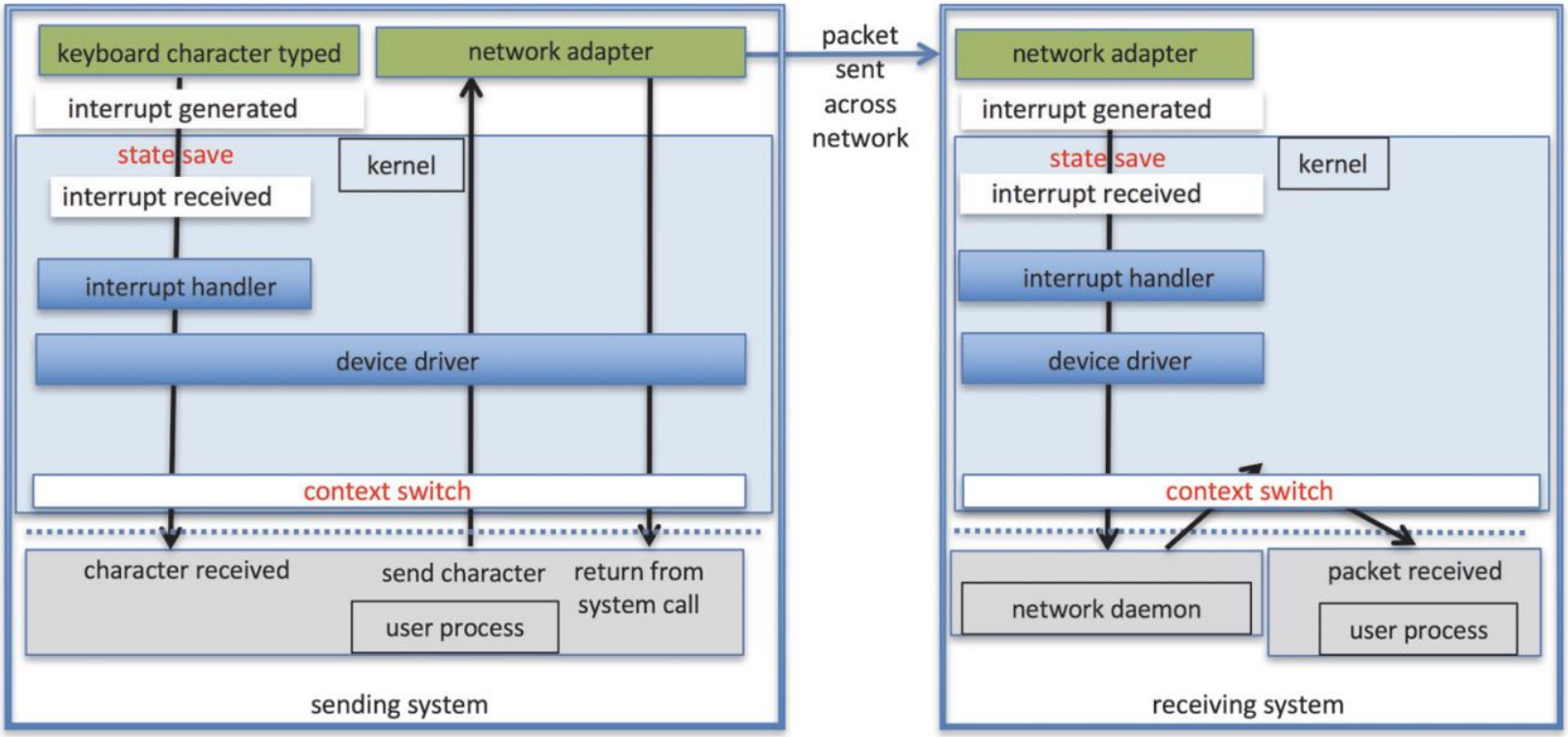


Figure 12.16 Intercomputer communications.

Hiệu suất

- Sử dụng 1 số nguyên tắc để nâng cao hiệu quả của I/O:
  - Giảm số lần context switch
  - Giảm số lần dữ liệu phải được sao chép vào bộ nhớ trong khi truyền giữa thiết bị và ứng dụng
  - Giảm tần suất bị gián đoạn bằng cách sử dụng các lần truyền lớn, bộ điều khiển thông minh và polling (nếu có thể giảm thiểu busy waiting)
  - Tăng khả năng tương tranh bằng cách sử dụng các bộ điều khiển hoặc kênh sử dụng DMA để giảm tải việc sao chép dữ liệu đơn giản từ CPU
  - Chuyển các tiến trình xử lý cơ bản vào phần cứng, cho phép hoạt động của chúng trong controller đồng thời với hoạt động của CPU và bus.
  - Cân bằng hiệu suất CPU, memory subsystem, bus và I/O vì tình trạng quá tải ở bất kỳ khu vực nào cũng gây ra tình trạng nhàn rỗi ở khu vực khác

Hiệu suất

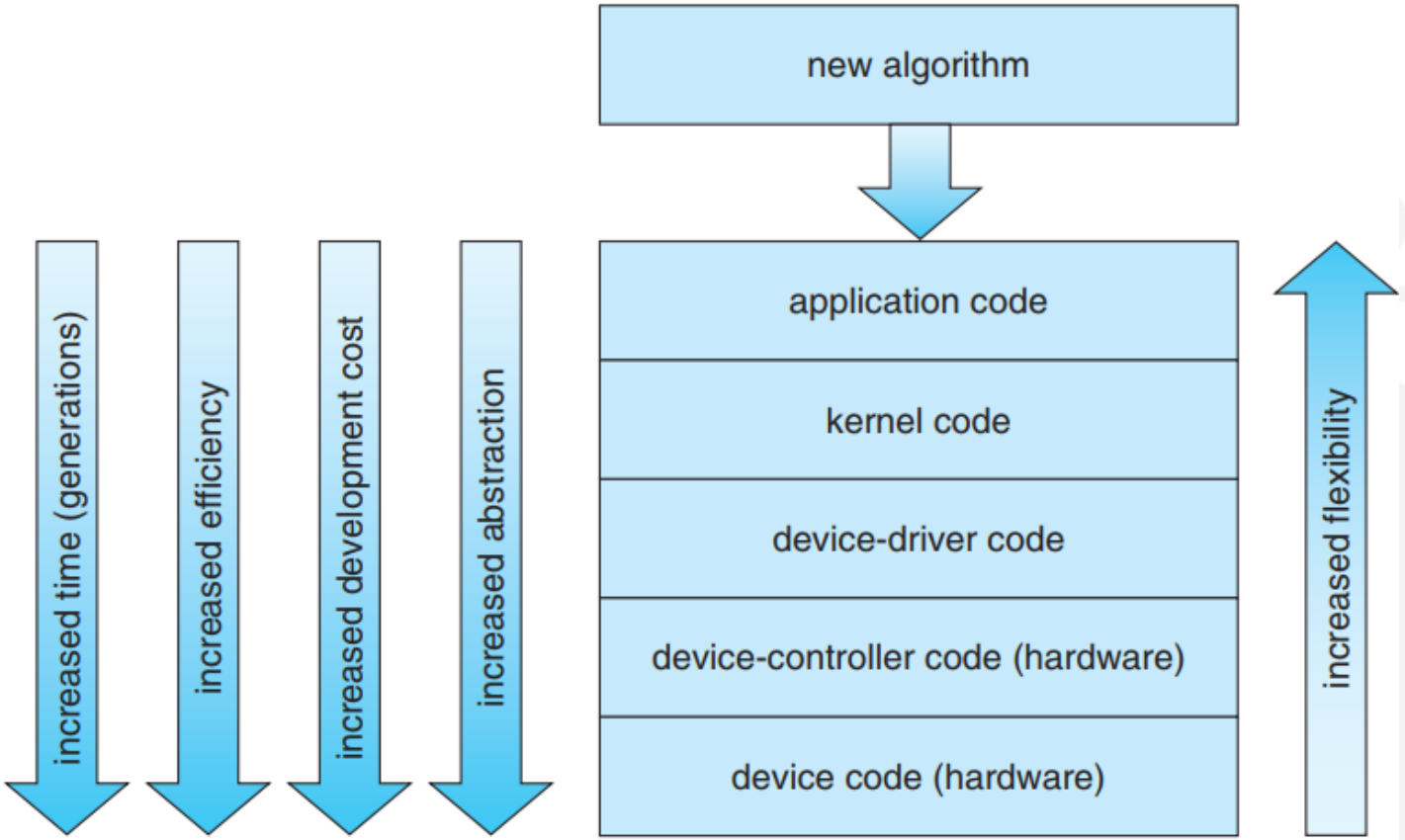


Figure 12.17 Device functionality progression.

Hiệu suất

- Ban đầu, triển khai các thuật toán I/O thử nghiệm ở cấp ứng dụng vì application code linh hoạt và các lỗi ứng dụng khó có thể gây ra sự cố hệ thống → tránh việc reboot hoặc reload driver thiết bị sau mỗi lần thay đổi code
  - Không hiệu quả do chi phí context switch và ứng dụng không thể tận dụng các CTDL và chức năng kernel (chẳng hạn như in-kernel messaging, threading và locking)
- Khi thuật toán cấp ứng dụng đã chứng minh được giá trị thì có thể triển khai tại kernel → cải thiện hiệu suất, nhưng việc phát triển gặp nhiều thách thức vì OS kernel lớn, phức tạp, cần phải debug kỹ lưỡng tránh hỏng dữ liệu và treo máy
- Hiệu suất cao nhất có thể đạt được là triển khai trên phần cứng, trong thiết bị hoặc trong bộ điều khiển. Những nhược điểm của việc triển khai phần cứng bao gồm khó khăn và chi phó trong việc cải tiến tiếp theo hoặc sửa lỗi, tăng thời gian phát triển và tình linh hoạt giảm.

Hiệu suất

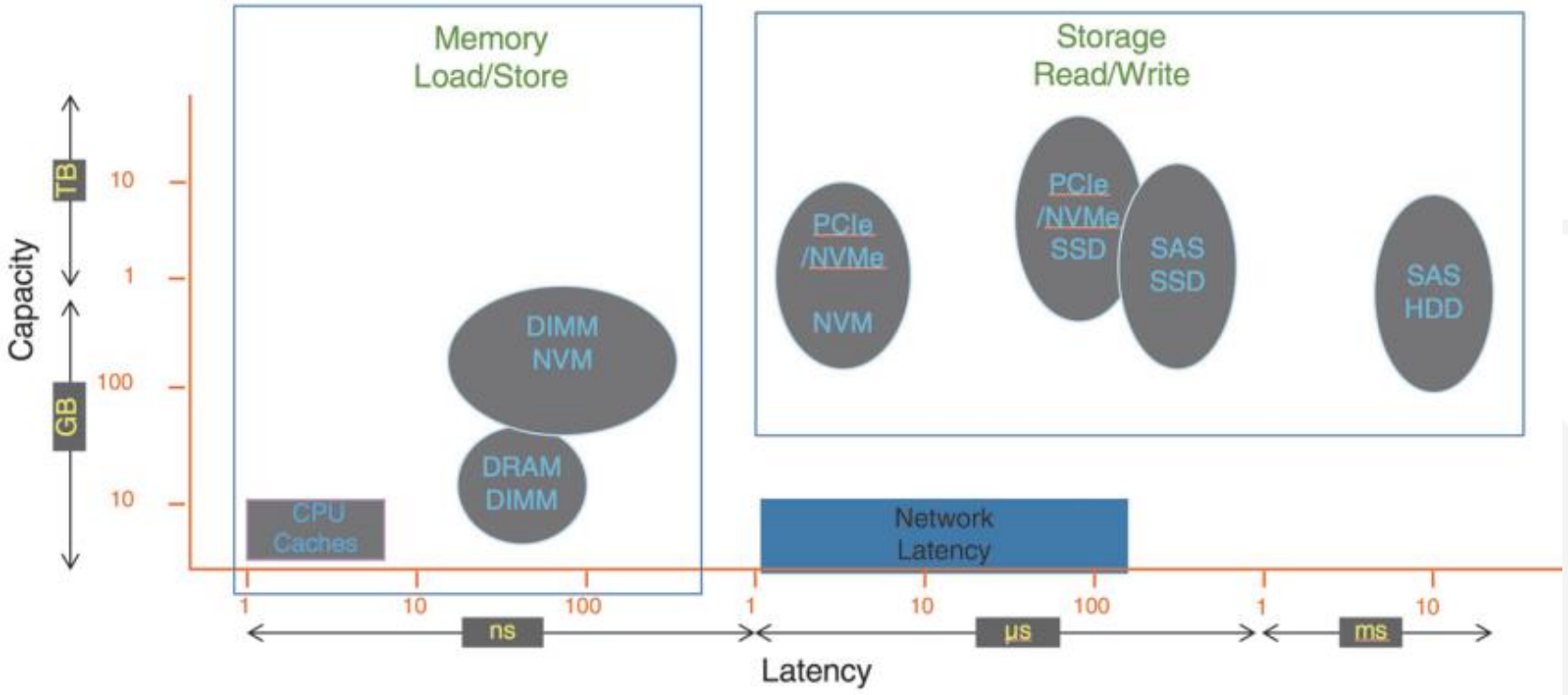


Figure 12.18 I/O performance of storage (and network latency).

Nội dung

- Cấu trúc lưu trữ lớn (Mass-storage Structure)
- Hệ thống Vào/ Ra (I/O system)
  - Tổng quan
  - I/O Hardware
  - Application I/O Interface
  - Kernel I/O Subsystem
  - Transforming I/O requests to hardware operations
  - STREAMS
  - Hiệu suất
  - **Summary**

Summary

- Các thành phần hardware cơ bản liên quan đến I/O là bus, device controller và chính thiết bị đó.
- Công việc di chuyển dữ liệu giữa các thiết bị và bộ nhớ được CPU thực hiện dưới dạng I/O được lập trình hoặc được chuyển sang bộ điều khiển DMA
- Module kernel điều khiển thiết bị là device driver. Interface của system call cung cấp cho các ứng dụng được thiết kế để xử lý 1 số loại hardware cơ bản, bao gồm block device, character stream device, network socket và bộ định thời khoảng thời gian được lập trình. System call thường chặn các tiến trình gọi chúng, nhưng các system call không chặn (nonblocking) và không đồng bộ được sử dụng bởi chính kernel và bơi ứng dụng không được sleep trong khi chờ thao tác I/O hoàn tất.

Summary

- Subsystem I/O của kernel cung cấp nhiều dịch vụ. Bao gồm lập lịch I/O, buffering, caching, spooling, device reservation, error handling, name translation.
- STREAMS là 1 triển khai và phương pháp cung cấp khuôn khổ cho cách tiếp cận module và tăng dần để viết device driver và các giao thức mạng. Thông qua STREAMS, các driver có thể được xếp chồng lên nhau, với dữ liệu truyền qua chúng một cách tuần tự và 2 chiều để xử lý.
- Các lệnh gọi hệ thống I/O tốn kém về mức tiêu thụ CPU do có nhiều lớp phần mềm giữa thiết bị vật lý và ứng dụng. Các lớp này bao hàm chi phí từ một số nguồn: context switch để vượt qua ranh giới bảo vệ của kernel, xử lý tín hiệu và ngắt để phục vụ các thiết bị I/O và tải lên CPU và hệ thống bộ nhớ để sao chép dữ liệu giữa kernel buffer và không gian ứng dụng.