UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537
Introduction to Operating Systems

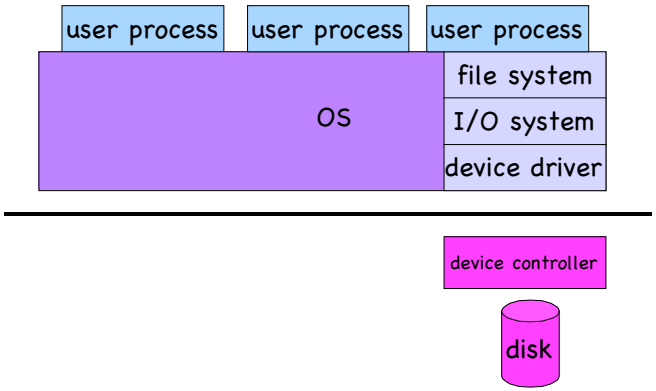
Andrea C. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau

# I/O System: Disks

Questions answered in this lecture:

- What are the layers of the I/O systems?
- How does a device driver interact with device controllers?
- What are the characteristics of modern disk drives?
- How do disks schedule requests?

# I/O System




# Device Drivers

Mechanism: Encapsulate details of device

- File system not aware of device details
- Much of OS code is in device drivers
  - Responsible for many of the errors as well!

Device driver interacts with device controller

- Read status registers, read data
- Write control registers, provide data for write operations

How does device driver access controller?

- Special instructions
  - Valid only in kernel mode, No longer popular
- Memory-mapped
  - Read and write to special memory addresses
  - Protect by placing in kernel address space only
    - May map part of device in user address space for fast access

# Device Drivers: Starting I/O

Programmed I/O (PIO)

- Must initiate and watch every byte
- Disadvantage: Large overhead for large transfers

Direct Memory Access (DMA)

- Offload work from CPU to to special-purpose processor responsible for large transfers
- CPU: Write DMA command block into main memory
  - Pointer to source and destination address
  - Size of transfer
- CPU: Inform DMA controller of address of command block
- DMA controller: Handles transfer with I/O device controller
- Can use physical or virtual addresses (DVMA)
  - Disadvantages of each approach??

## Device Drivers: When is I/O complete?

Polling

- Handshake by setting and clearing flags
  - Controller sets flag when done
  - CPU repeatedly checks flag
- Disadvantage: Busy-waiting
  - CPU wastes cycles when I/O device is slow
  - Must be attentive to device, or could lose data

Interrupts: Handle asynchronous events

- Controller asserts interrupt request line when done
- CPU jumps to appropriate interrupt service routine (ISR)
  - Interrupt vector: Table of ISR addresses
  - Index by interrupt number
- Low priority interrupts postponed until higher priority finished
- Combine with DMA: Do not interrupt CPU for every byte

## Disk Controller

Responsible for interface between OS and disk drive

- Common interfaces: ATA/IDE vs. SCSI
  - ATA/IDE used for personal storage
  - SCSI for enterprise-class storage

Basic operations

- Read block
- Write block

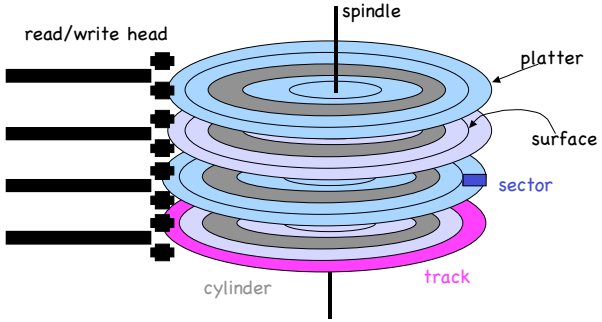
OS does not know of internal complexity of disk

- Disk exports array of Logical Block Numbers (LBNs)
- Disks map internal sectors to LBNs

Implicit contract:

- Large sequential accesses to contiguous LBNs achieve much better performance than small transfers or random accesses

## Disk Terminology




ZBR (Zoned bit recording): More sectors on outer tracks

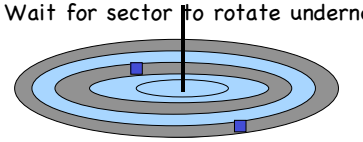
## Disk Performance

How long to read or write n sectors?

- Positioning time + Transfer time (n)
- Positioning time: Seek time + Rotational Delay
- Transfer time: n / (RPM * bytes/track)

Seek: Time to position head over destination cylinder

Rotation: Wait for sector to rotate underneath head



## Disk Calculations

Example disk:

- #surfaces: 4
- #tracks/surface: 64K
- #sectors/track: 1K (assumption??)
- #bytes/sector: 512
- RPM: 7200 = 120 tracks/sec
- Seek cost: 1.3ms – 16ms

Questions

- How many disk heads? How many cylinders?
- How many sectors/cylinder? Capacity?
- What is the maximum transfer rate (bandwidth)?
- Average positioning time for random request?
- Time and bandwidth for random request of size:
  - 4KB?
  - 128 KB?
  - 1 MB?

## Disk Abstraction

How should disk map internal sectors to LBNs?

Goal: Sequential accesses (or contiguous LBNs) should achieve best performance

Approaches:

- Traditional ordering
- Serpentine ordering

## Positioning

Drive servo system keeps head on track

- How does the disk head know where it is?
- Platters not perfectly aligned, tracks not perfectly concentric (runout) -- difficult to stay on track
- More difficult as density of disk increase
  - More bits per inch (BPI), more tracks per inch (TPI)

Use servo burst:

- Record placement information every few (3-5) sectors
- When head cross servo burst, figure out location and adjust as needed

## Reliability

Disks fail more often....

- When continuously powered-on
- With heavy workloads
- Under high temperatures

How do disks fail?

- Whole disk can stop working (e.g., motor dies)
- Transient problem (cable disconnected)
- Individual sectors can fail (e.g., head crash or scratch)
  - Data can be corrupted or block not readable/writable

Disks can internally fix some sector problems

- ECC (error correction code): Detect/correct bit flips
- Retry sector reads and writes: Try 20-30 different offset and timing combinations for heads
- Remap sectors: Do not use bad sectors in future
  - How does this impact performance contract??

## Buffering

Disks contain internal memory (2MB-16MB) used as cache

Read-ahead: "Track buffer"

- Read contents of entire track into memory during rotational delay

Write caching with volatile memory

- Immediate reporting: Claim written to disk when not
- Data could be lost on power failure
  - Use only for user data, not file system meta-data

Command queueing

- Have multiple outstanding requests to the disk
- Disk can reorder (schedule) requests for better performance

## Disk Scheduling

Goal: Minimize positioning time

- Performed by both OS and disk itself; Why?

FCFS: Schedule requests in order received

- Advantage: Fair
- Disadvantage: High seek cost and rotation

Shortest seek time first (SSTF):

- Handle nearest cylinder next
- Advantage: Reduces arm movement (seek time)
- Disadvantage: Unfair, can starve some requests

## Disk Scheduling

SCAN (elevator): Move from outer cylinder in, then back out again

- Advantage: More fair to requests, similar performance as SSTF
- Variation: Circular-Scan (C-Scan)
  - Move head only from outer cylinder inward (then start over)
  - Why??? (Two reasons)

LOOK: SCAN, except stop at last request

Calculate seek distance for workload with cylinder #s: 10, 2, 0, 85, 50, 40, 1, 37, 41; Start at #43, moving up

## Disk Scheduling

Real goal: Minimize positioning time

- Trend: Rotation time dominating positioning time
  - Very difficult for OS to predict
    - ZBR, track and cylinder skew, serpertine layout, bad block remapping, caching, ...
  - Disk controller can calculate positioning time
- Shortest positioning time first (SPTF)

Technique to prevent starvation

- Two queues
- Handle requests in current queue
- Add newly arriving requests added to other queue