# CS 537
# Lecture 11
# Secondary Storage

Michael Swift





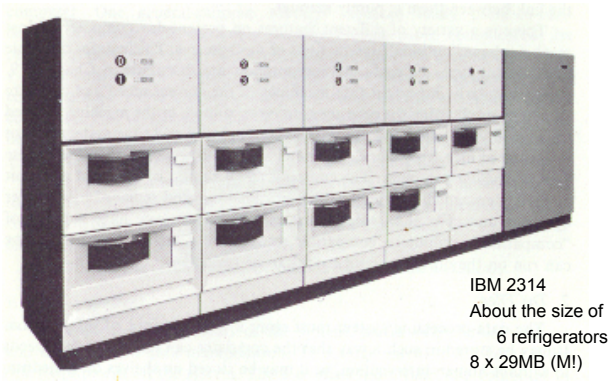
## Secondary storage

- Secondary storage typically:
  - is anything that is outside of "primary memory"
  - does not permit direct execution of instructions or data retrieval via machine load/store instructions
- Characteristics:
  - it's large: 80GB-1TB
  - it's cheap: 0.30¢/GB
  - it's persistent: data survives power loss
  - it's slow: 100us-10 ms to access (compared to 100ns for ram)
    - why is this slow??





## Another trip down memory lane …



IBM 2314
About the size of
6 refrigerators
8 x 29MB (M!)





## Disk trends

- Disk capacity, 1975-1989
  - doubled every 3+ years
  - 25% improvement each year
  - factor of 10 every decade
  - exponential, but far less rapid than processor performance
- Disk capacity since 1990
  - doubling every 12 months
  - 100% improvement each year
  - factor of 1000 every decade
  - 10x as fast as processor performance!

## Rotating Disks vs. Solid-State Disks (AKA Flash)

- Forget everything you knew about rotating disks. SSDs are different
- SSDs are complex software systems
- More later

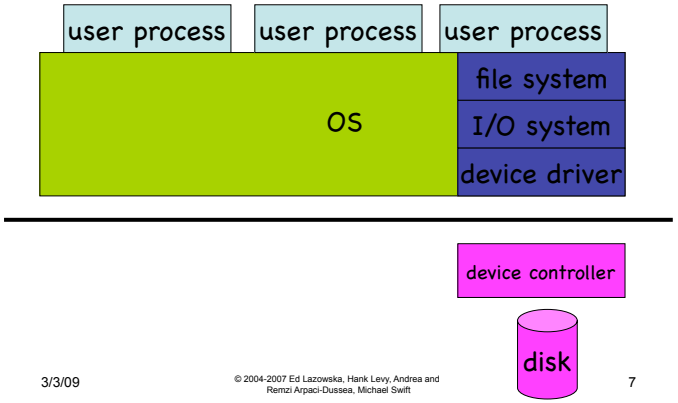
## Disks and the OS

- Disks are messy, messy devices
  - errors, bad blocks, missed seeks, etc.
- Job of OS is to hide this mess from higher-level software
  - low-level device drivers (initiate a disk read, etc.)
  - higher-level abstractions (files, databases, etc.)
- OS may provide different levels of disk access to different clients
  - physical disk block (surface, cylinder, sector)
  - disk logical block (disk block #)
  - file logical (filename, block or record or byte #)





## 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??





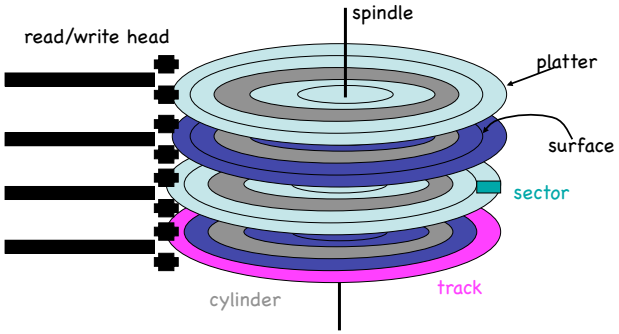
## Device Drivers: When is I/O complete?

- 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 Terminology




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



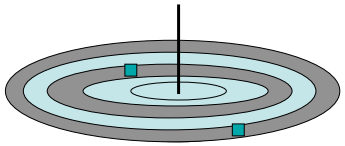


## Example disk characteristics

- IBM Ultrastar 36XP drive
  - form factor: 3.5”
  - capacity: 36.4 GB
  - rotation rate: 7,200 RPM (120 RPS)
  - platters: 10
  - surfaces: 20
  - sector size: 512-732 bytes
  - cylinders: 11,494
  - cache: 4MB
  - transfer rate: 17.9 MB/s (inner) – 28.9 MB/s (outer)
  - full seek: 14.5 ms
  - head switch: 0.3 ms

## 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)







## Disk performance

- Performance depends on a number of steps
  - seek: moving the disk arm to the correct cylinder
    - depends on how fast disk arm can move
      - seek times aren't diminishing very quickly (why?)
  - rotation (latency): waiting for the sector to rotate under head
    - depends on rotation rate of disk
      - rates are increasing, but slowly (why?)
  - transfer: transferring data from surface into disk controller, and from there sending it back to host
    - depends on density of bytes on disk
      - increasing, and very quickly
- When the OS uses the disk, it tries to minimize the cost of all of these steps
  - particularly seeks and rotation





## 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?





## Interacting with disks

- In the old days…
  - OS would have to specify cylinder #, sector #, surface #, transfer size
    - i.e., OS needs to know all of the disk parameters
- Modern disks are even more complicated
  - not all sectors are the same size, sectors are remapped, …
  - disk provides a higher-level interface, e.g., SCSI
    - exports data as a logical array of blocks [0 … N]
    - maps logical blocks to cylinder/surface/sector
    - OS only needs to name logical block #, disk maps this to cylinder/surface/sector
    - on-board cache
    - as a result, physical parameters are hidden from OS
      - both good and bad

## Disk Controller

- Responsible for interface between OS and disk drive
  - Common interfaces: ATA/IDE vs. SCSI
    - ATA/IDE used for personal storage: slow rotation, seek, high capacity
    - SCSI for enterprise-class storage: faster rotation and seek
    - QUESTION: which will be larger diameter? Which will have more platters?
- 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 Abstraction

- How should disk map internal sectors to LBNs?
- Goal: Sequential accesses (or contiguous LBNs) should achieve best performance
- Approaches:
  - Traditional ordering
  - Serpentine ordering





## 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

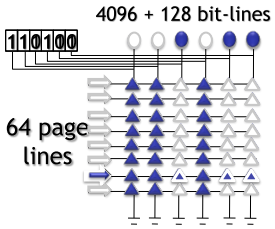
- SCAN (elevator): Move from outer cylinder in, then back out again
  - Advantage: More fair to requests, similar performance as SSTF
  - Variation: Circular-Scan (C-Scan)
    - Move head only from outer cylinder inward (then start over)
    - Why??? (Two reasons)
- LOOK: SCAN, except stop at last request
- Calculate seek distance for workload with cylinder #s: 10, 2, 0, 85, 50, 40, 1, 37, 41; Start at #43, moving up





## Flash disks: solid state storage
### NAND flash blocks

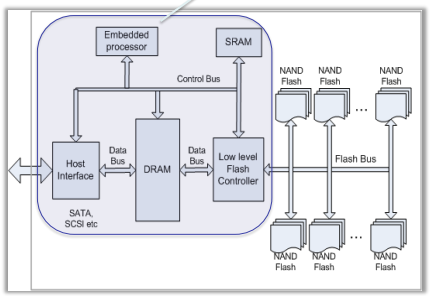
- A flash block is a grid of cells
  - Single Level Cell (SLC) = 1 bit per cell (faster, more reliable)
  - Multi Level Cell (MLC) = 2 bits per cell (slower, less reliable)




- Erase: set all bits to 1
- Program: clear some bits
- Read: NAND operation with a page selected

## Background
### SSD Structure

Flash Translation Layer (Proprietary firmware)



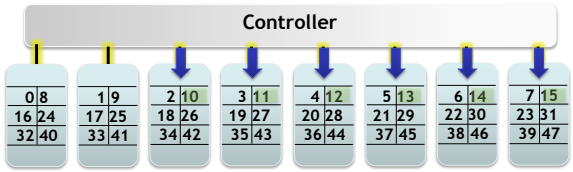

Simplified block diagram of an SSD

## Write-in-Place vs. Logging

- Rotating disks
  - Constant map from LBA to on-disk location
- SSDs
  - Writes always to new locations
  - Superseded blocks cleaned later

## Striping

- LBAs striped across flash packages
  - Single request can span multiple chips
  - Natural load balancing
- What's the right stripe size?




## Failure Modes
### Rotating disks

- Media imperfections, loose particles, vibration
- Latent sector errors [Bairavasundaram 07]
  - E.g., with uncorrectable ECC
  - Frequency of affected disks increases linearly with time
  - Most affected disks (80%) have < 50 errors
  - Temporal and spatial locality
  - Correlation with recovered errors
- Disk scrubbing helps

## Failure Modes
### SSDs

- Types of NAND flash errors (mostly when erases > wear limit)
  - Write errors: Probability varies with # of erasures
  - Read disturb: Increases with # of reads
  - Data retention errors: Charge leaks over time
  - Little spatial or temporal locality (within equally worn blocks)
- Better ECC can help
- Errors increase with wear: Need wear-leveling