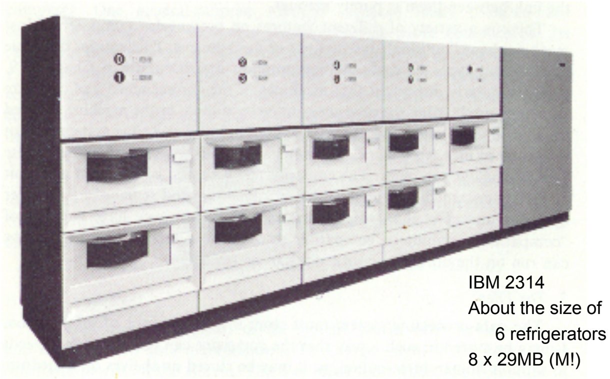
CS 537
Lecture 9
Disks

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: milliseconds to access
    - 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!

Disks and the OS

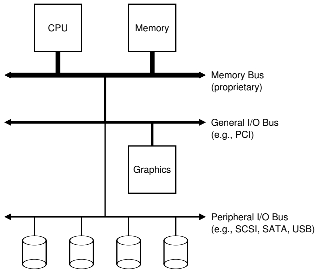
- Disks are messy, messy devices
  - errors, bad blocks, missed seeks, etc.
- Job of OS is to hide this mess from higher-level software
  - low-level device drivers (initiate a disk read, etc.)
  - higher-level abstractions (files, databases, etc.)
- OS may provide different levels of disk access to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

Types of IO devices

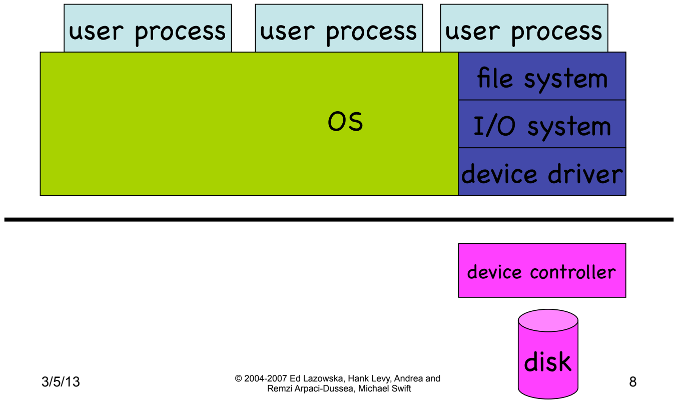
- Two categories
  - A ***block device*** stores information in fixed-size blocks, each one with its own address
    - e.g., disks
  - A ***character device*** delivers or accepts a stream of characters, and individual characters are not addressable
    - e.g., keyboards, printers, network cards
- Device driver provides interface for these two types of devices
  - Other OS components see block devices and character devices, but not the details of the devices.
  - How to effectively utilize the device is the responsibility of the device driver

System Architecture

- Devices attach to buses
  - Discover device and notify OS
  - Provide commands for communication
- Examples:
  - USB: message-based
  - PCI: memory-mapped I/O
- Buses attach to other buses
  - USB attaches through PCI



I/O System



Device Drivers

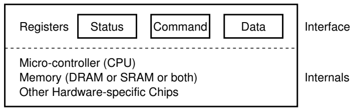
- Mechanism: Encapsulate details of device
  - File system not aware of device details
  - Much of OS code is in device drivers
    - Responsible for many of the errors as well!
- Device driver interacts with device controller
  - Read status registers, read data
  - Write control registers, provide 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

Calling Drivers

- Every driver for a type of device (a **class**) has the same interface
  - If you can call one, you can call them all
- Example
  - Network:
    - Initialize, send packet, configure, close
  - Sound
    - Play sound, record sound, control
  - Block
    - Read block, write block

Programmed I/O

- Devices have registers available through memory-mapped I/O
  - Status, command, data



- Programmed I/O loops reading/writing registers = POLLING
  1. Wait for ready
  2. Write data
  3. Write command
  4. Wait for completion

```
While (STATUS == BUSY)
    ; // wait until device is not busy
Write data to DATA register
Write command to COMMAND register
    (Doing so starts the device and executes the command)
While (STATUS == BUSY)
    ; // wait until device is done with your request
```

Faster I/O through Interrupts

- Polling for available/completion is inefficient
  - CPU is busy waiting for slow I/O device

```
CPU   1111111111ppppppppppp111111111111
Disk  ----------11111111111------------
```

- Better approach: do something else, get **notified** when I/O complete: an interrupt
  - Interrupt service routine (ISR) in driver gets called to do the work (e.g. loop writing data

```
CPU   111111111122222222222111111111111
Disk  ----------11111111111------------
```

  - Speeds steps 1,4
- Interrupts are not always faster. When not?

Device Drivers: Starting I/O

- Interrupts don’t help with step 2: copying data

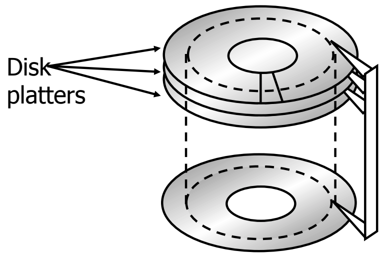
```
CPU    1111111111cccccc22222222222111111111111
Disk   ----------------11111111111------------
```

- 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

```
CPU    111111111122222222222222222111111111111
DMA    ----------cccccc-----------------------
Disk   ----------------11111111111------------
```

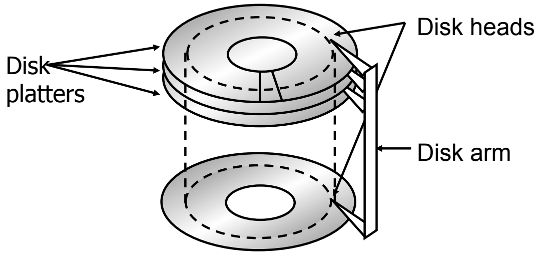
Disk Characteristics

- ***Disk platters:*** coated with magnetic materials for recording



Disk Characteristics

- ***Disk arm:*** moves a comb of ***disk heads***
  - Only one disk head is active for reading/writing

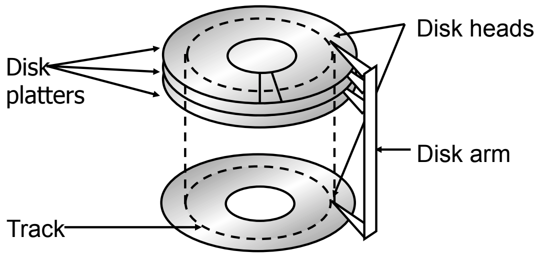


Hard Disk Trivia…

- Aerodynamically designed to fly
  - As close to the surface as possible
  - No room for air molecules
  - Like flying a 747 at Mach 41 inch off the ground (in 1982!)
- Therefore, hard drives are filled with special inert gas
- If head touches the surface
  - Head crash
  - Scrapes off magnetic information

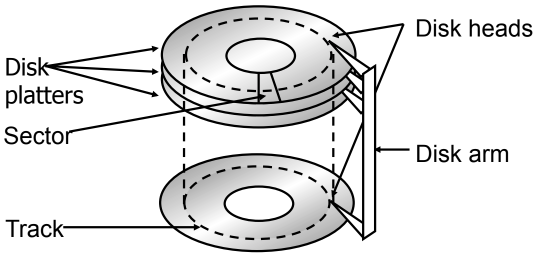
Disk Characteristics

- Each disk platter is divided into concentric ***tracks***



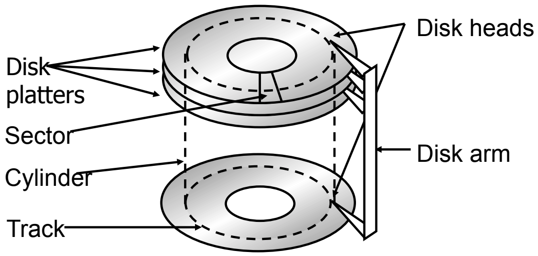
Disk Characteristics

- A track is further divided into ***sectors.*** A sector is the smallest unit of disk storage



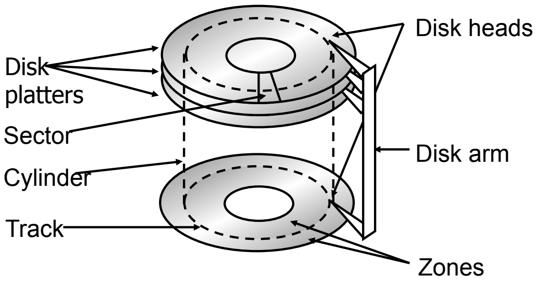
Disk Characteristics

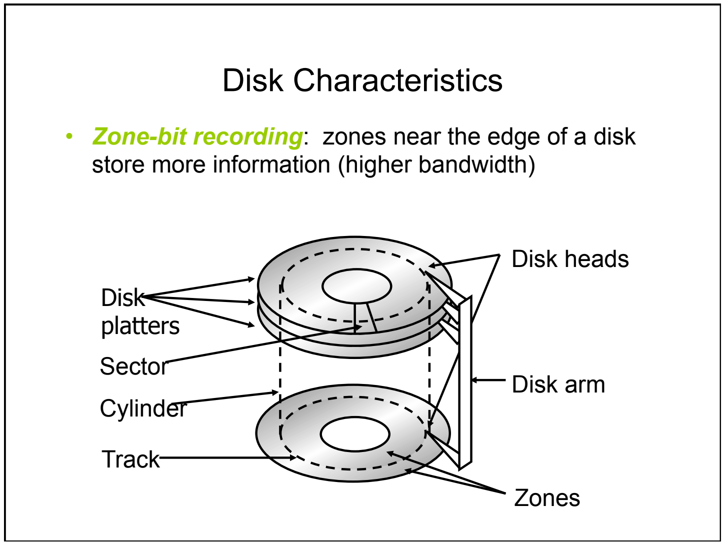
- A ***cylinder*** consists of all tracks with a given disk arm position



Disk Characteristics

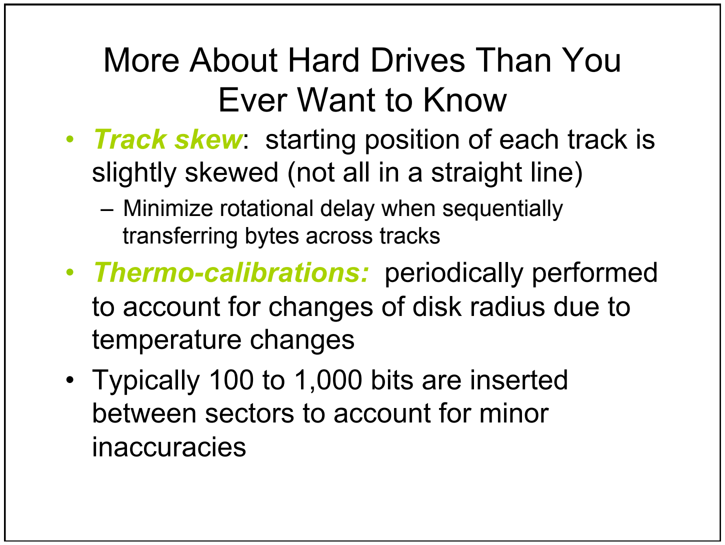
- Cylinders are further divided into ***zones***





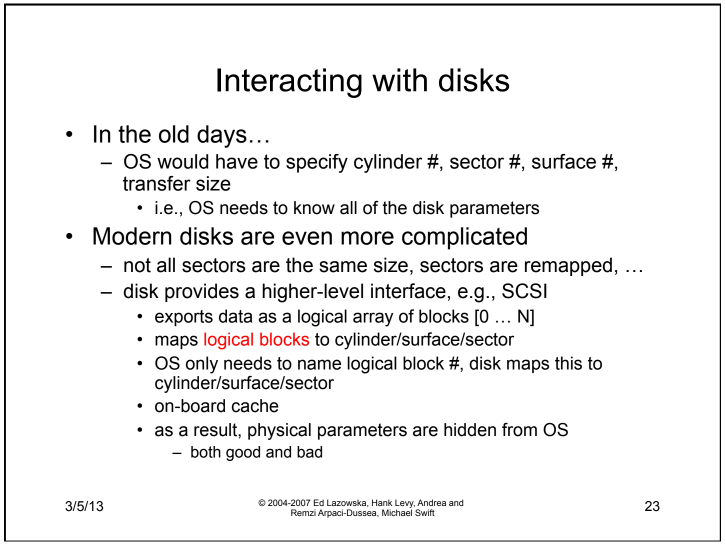
Disk Characteristics

- ***Zone-bit recording***: zones near the edge of a disk store more information (higher bandwidth)



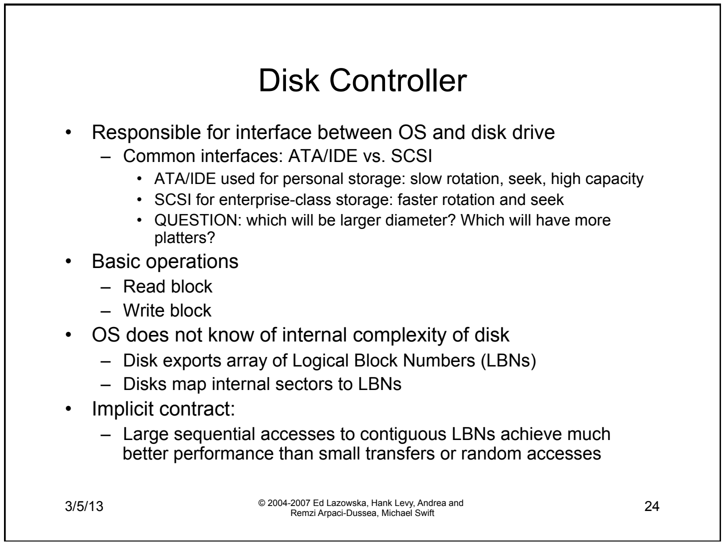
More About Hard Drives Than You Ever Want to Know

- ***Track skew***: starting position of each track is slightly skewed (not all in a straight line)
  - Minimize rotational delay when sequentially transferring bytes across tracks
- ***Thermo-calibrations:*** periodically performed to account for changes of disk radius due to temperature changes
- Typically 100 to 1,000 bits are inserted between sectors to account for minor inaccuracies



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

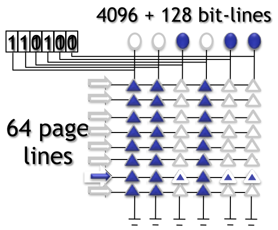
- Responsible for interface between OS and disk drive
  - Common interfaces: ATA/IDE vs. SCSI
    - ATA/IDE used for personal storage: slow rotation, seek, high capacity
    - SCSI for enterprise-class storage: faster rotation and seek
    - QUESTION: which will be larger diameter? Which will have more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

Flash disks: solid state storage
NAND flash blocks

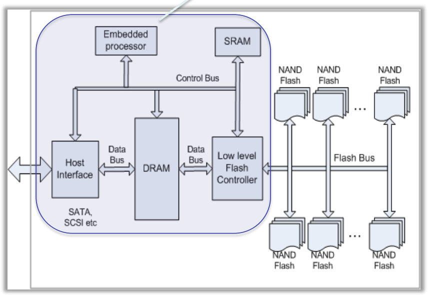
- A flash block is a grid of cells
  - Single Level Cell (SLC) = 1 bit per cell (faster, more reliable)
  - Multi Level Cell (MLC) = 2 bits per cell (slower, less reliable)



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

Background
SSD Structure

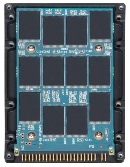
Flash Translation Layer (Proprietary firmware)



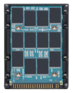
Simplified block diagram of an SSD

Write-in-Place vs. Logging

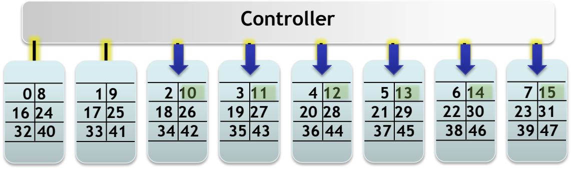
- Rotating disks
  - Constant map from LBA to on-disk location
- SSDs
  - Writes always to new locations
  - Superseded blocks cleaned later



Striping



- LBAs striped across flash packages
  - Single request can span multiple chips
  - Natural load balancing



SSD performance

Device	Random Read µs	Seq Read µs	Random Write µs	Seq. Write µs	$/GB
DRAM	0.05	0.05	0.05	0.05	$15
Flash	100	85	2,000	200-500	$3
Disk	5,000	500	5,000	500	$0.3