Anatomy of a disk

- **Stack of magnetic platters**
  - Rotate together on a central spindle @3,600-15,000 RPM
  - Drives speed drifts slowly over time
  - Can’t predict rotational position after 100-200 revolutions

- **Disk arm assembly**
  - Arms rotate around pivot, all move together
  - Pivot offers some resistance to linear shocks
  - Arms contain disk heads–one for each recording surface
  - Heads read and write data to platters
Storage on a magnetic platter

- Platters divided into concentric *tracks*
- A stack of tracks of fixed radius is a *cylinder*
- Heads record and sense data along cylinders
  - Significant fractions of encoded stream for error correction
- Generally only one head active at a time
  - Disks usually have one set of read-write circuitry
  - Must worry about cross-talk between channels
  - Hard to keep multiple heads exactly aligned
Disk positioning system

- Move head to specific track and keep it there
  - Resist physical socks, imperfect tracks, etc.

- A *seek* consists of up to four phases:
  - *speedup*—accelerate arm to max speed or half way point
  - *coast*—at max speed (for long seeks)
  - *slowdown*—stops arm near destination
  - *settle*—adjusts head to actual desired track

- Very short seeks dominated by settle time (∼1 ms)

- Short (200-400 cyl.) seeks dominated by speedup
  - Accelerations of 40g
Seek details

- **Head switches comparable to short seeks**
  - May also require head adjustment
  - Settles take longer for writes than reads

- **Disk keeps table of pivot motor power**
  - Maps seek distance to power and time
  - Disk interpolates over entries in table
  - Table set by periodic “thermal recalibration”
  - 500 ms recalibration every 25 min, bad for AV

- “**Average seek time**” quoted can be many things
  - Time to seek 1/3 disk, 1/3 time to seek whole disk,
Sectors

- Disk interface presents linear array of sectors
  - Generally 512 bytes, written atomically

- Disk maps logical sector #s to physical sectors
  - Zoning—puts more sectors on longer tracks
  - Track skewing—sector 0 pos. varies for sequential I/O
  - Sparing—flawed sectors remapped elsewhere

- OS doesn’t know logical to physical sector mapping
  - Larger logical sector # difference means larger seek
  - Highly non-linear relationship (and depends on zone)
  - OS has no info on rotational positions
  - Can empirically build table to estimate times
Disk interface

- Controls hardware, mediates access

- Computer, disk often connected by bus
  - Example: IDE (you saw in bootloader lab)
  - Most high-performance systems use SCSI (will discuss)

- Command queuing: Give disk multiple requests
  - Disk can schedule them using rotational information

- Disk cache used for read-ahead
  - Otherwise, sequential reads would incur whole revolution
  - Cross track boundaries? Can’t stop a head-switch

- Some disks support write caching
  - But data not stable—not suitable for all requests
Scheduling: First come first served (FCFS)

• Process disk requests in the order they are received

• Advantages
  - 
  - 

• Disadvantages
  - 
  - 
Scheduling: First come first served (FCFS)

- Process disk requests in the order they are received

- **Advantages**
  - Easy to implement
  - Good fairness

- **Disadvantages**
  - Cannot exploit request locality
  - Increases average latency, decreasing throughput
Shortest positioning time first (SPTF)

- Always pick request with shortest seek time

- Advantages
  - 
  - 

- Disadvantages
  - 
  -

- Improvement
  - 
  -
Shortest positioning time first (SPTF)

- Always pick request with shortest seek time

- **Advantages**
  - Exploits locality of disk requests
  - Higher throughput

- **Disadvantages**
  - Starvation
  - Don’t always know what request will be fastest

- **Improvement: Aged SPTF**
  - Give older requests higher priority
  - Adjust “effective” seek time with weighting factor:
    \[ T_{\text{eff}} = T_{\text{pos}} - W \cdot T_{\text{wait}} \]
“Elevator” scheduling (SCAN)

- Sweep across disk, servicing all requests passed
  - Like SPTF, but next seek must be in same direction
  - Switch directions only if no further requests

- Advantages
  - 
  - 
  - 

- Disadvantages
  - 
  - 
  - 

- Variant
“Elevator” scheduling (SCAN)

- Sweep across disk, servicing all requests passed
  - Like SPTF, but next seek must be in same direction
  - Switch directions only if no further requests

- Advantages
  - Takes advantage of locality
  - Bounded waiting

- Disadvantages
  - Cylinders in the middle get better service
  - Might miss locality SPTF could exploit

- CSCAN: Only sweep in one direction

  *Very commonly used algorithm in Unix*
VSCAN(r)

- **Continuum between SPTF and SCAN**
  - Like SPTF, but slightly uses “effective” positioning time
    If request in same direction as previous seek: $T_{\text{eff}} = T_{\text{pos}}$
    Otherwise: $T_{\text{eff}} = T_{\text{pos}} + r \cdot T_{\text{max}}$
  - when $r = 0$, get SPTF, when $r = 1$, get SCAN
  - E.g., $r = 0.2$ works well

- **Advantages and disadvantages**
  - Those of SPTF and SCAN, depending on how $r$ is set
SCSI overview

- **SCSI domain** consists of devices and an SDS
  - Devices: host adapters & SCSI controllers
  - *Service Delivery Subsystem* connects devices—e.g., SCSI bus

- **SCSI-2 bus (SDS)** connects up to 8 devices
  - Controllers can have > 1 “logical units” (LUNs)
  - Typically, controller built into disk and 1 LUN/target, but “bridge controllers” can manage multiple physical devices

- **Each device can assume role of *initiator* or *target***
  - Traditionally, host adapter was initiator, controller target
  - Now controllers act as initiators (e.g., COPY command)
  - Typical domain has 1 initiator, ≥ 1 targets
SCSI requests

- A *request* is a command from initiator to target
  - Once transmitted, target has control of bus
  - Target may disconnect from bus and later reconnect
    (very important for multiple targets or even multitasking)

- **Commands contain the following:**
  - *Task identifier*—initiator ID, target ID, LUN, tag
  - *Command descriptor block*—e.g., read 10 blocks at pos. $N$
  - Optional *task attribute*—SIMPLE, ORDERD, HEAD OF QUEUE
  - Optional: output/input buffer, sense data
  - *Status byte*—GOOD, CHECK CONDITION, INTERMEDIATE, …
Executing SCSI commands

- Each LUN maintains a queue of *tasks*
  - Each task is DORMANT, BLOCKED, ENABLED, or ENDED
  - SIMPLE tasks are dormant until no ordered/head of queue
  - ORDERED tasks dormant until no HoQ/more recent ordered
  - HoQ tasks begin in enabled state

- Task management commands available to initiator
  - Abort/terminate task, Reset target, etc.

- Linked commands
  - Initiator can link commands, so no intervening tasks
  - E.g., could use to implement atomic read-modify-write
  - Intermediate commands return status byte INTERMEDIATE
SCSI exceptions and errors

- After error stop executing most SCSI commands
  - Target returns with CHECK CONDITION status
  - Initiator will eventually notice error
  - Must read specifics w. REQUEST SENSE

- Prevents unwanted commands from executing
  - E.g., initiator may not want to execute 2nd write if 1st fails

- Simplifies device implementation
  - Don’t need to remember more than one error condition

- Same mechanism used to notify of media changes
  - I.e., ejected tape, changed CD-ROM
But back in the 80s…

- Disks spin at 3,600 RPM
  - 17 ms/Rotation (vs. 4 ms on fastest disks today)

- Fixed # sectors/track (no zoning)

- Head switching free (?)

- Requests issued one at a time
  - No caching in disks
  - Head must pass over sector after getting a read
  - By the time OS issues next request, too late for next sector

- Slower CPUs, memory
  - Noticeable cost for block allocation algorithms