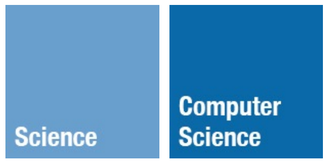
Hard disk drives



CS 450: Operating Systems
Michael Lee <lee@iit.edu>

Agenda

- Disk API
- HDD geometry and access
- Disk scheduling

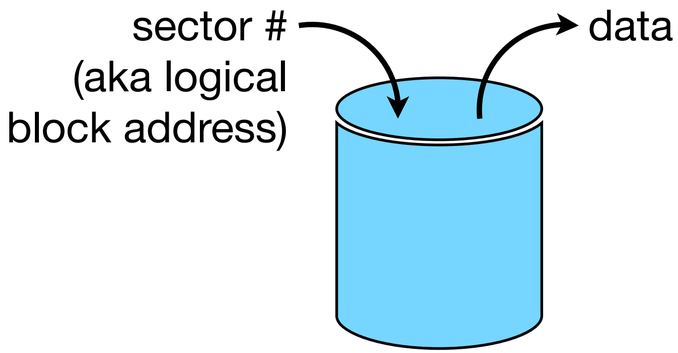
§ Disk API

Blocks and Sectors

- The OS manages data persisted to disks (and some other devices) in fixed or variable sized **blocks**
  - Typically 512 bytes - 64 KB in size
- Minimum addressable unit in a HDD is a 512-byte **sector**

HDD usage

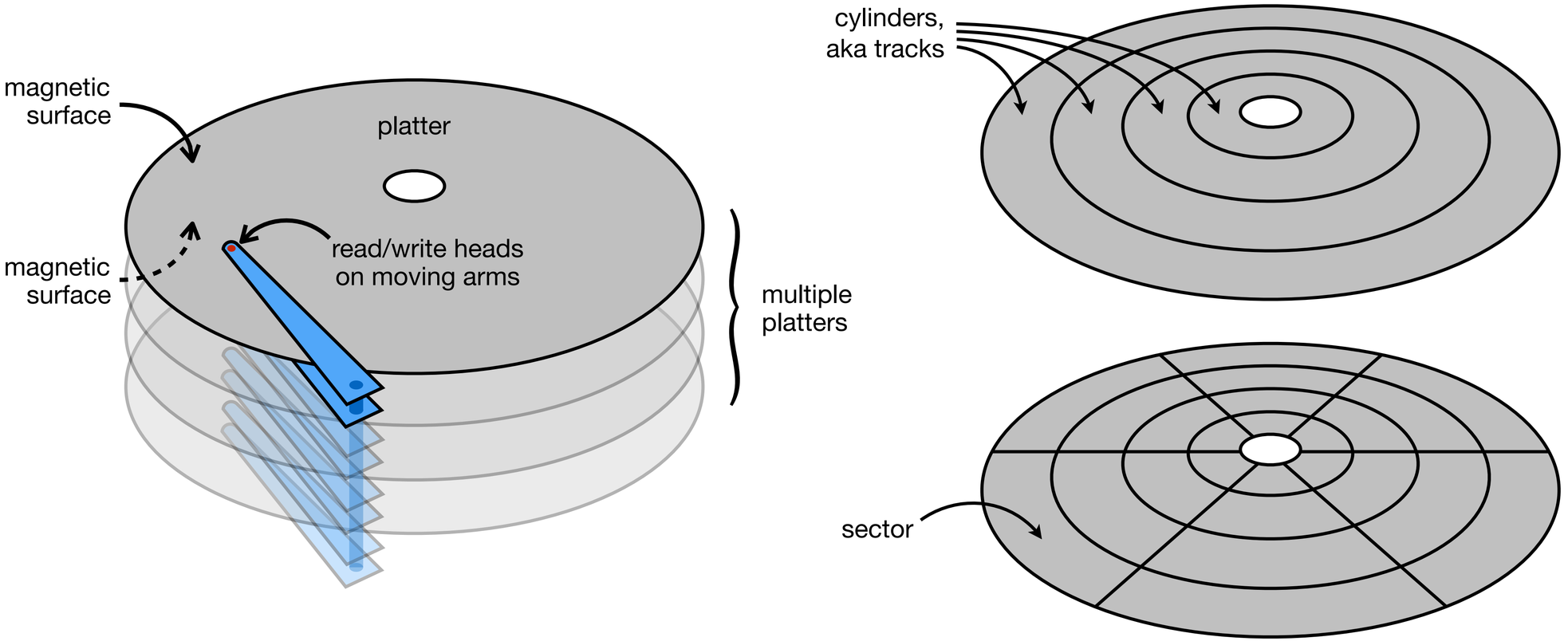
- Naive approach: treat a disk as a random-access array of sectors



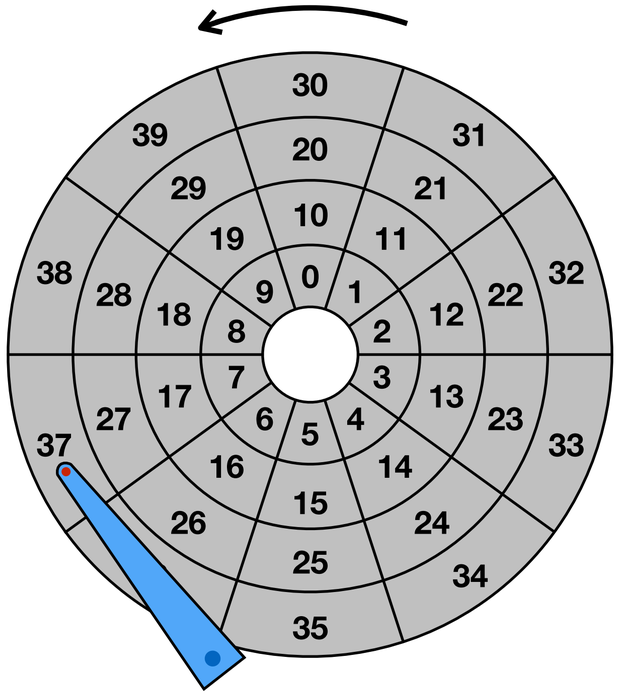
- But HDDs have characteristics that would make this very suboptimal!
  - Need a better understanding of them to use them efficiently

§ HDD geometry and access

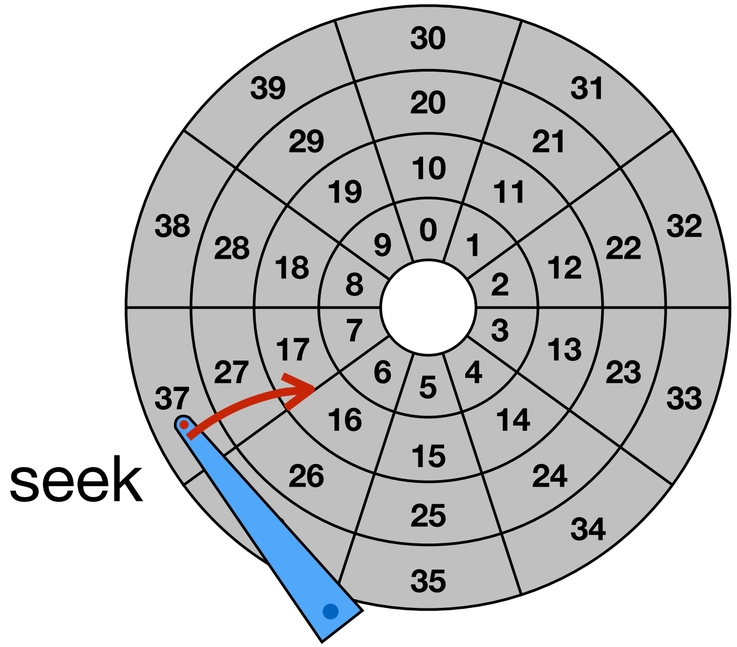
HDD geometry



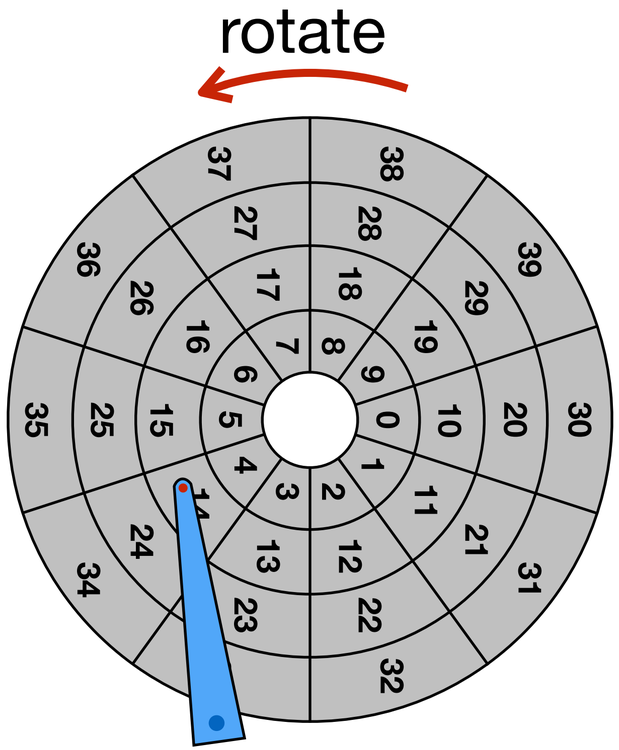
Block requests ⇒ Disk sectors



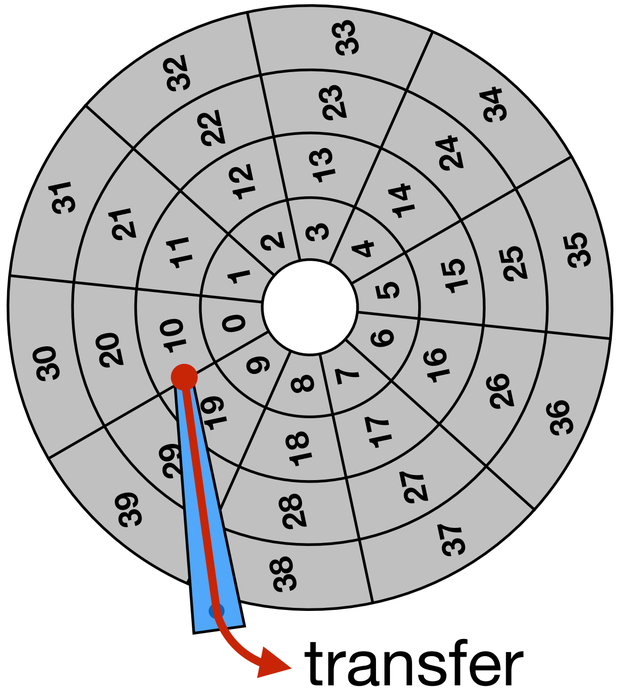
E.g., read sector 10



E.g., read sector 10



E.g., read sector 10



Seek, Rotate, Transfer

- Mechanical movement to place head over appropriate track
  - Phases: accelerate, coast, decelerate, settle
  - Typical time ≈ 5-10ms

Seek, **Rotate**, Transfer

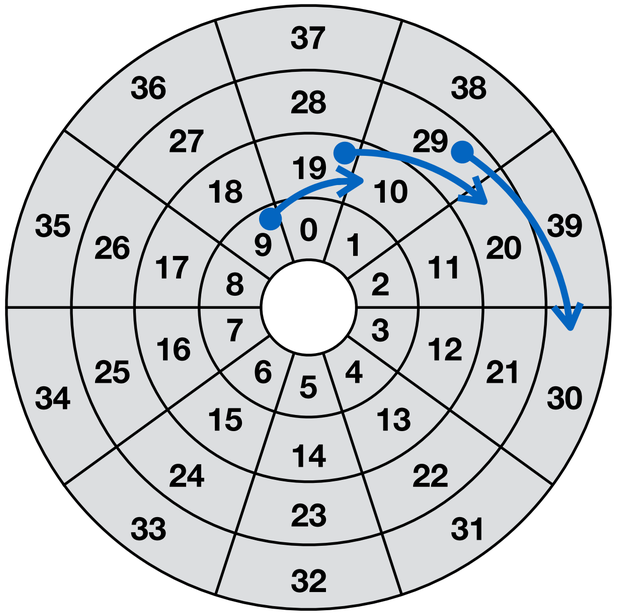
- HHDs have a fixed RPM (rotations per minute)
  - Typical values: 5400 (laptop), 7200, 10000 (workstation)
- E.g., for 7200 RPM drive:
  - 7200 / minute = 120 / second ≈ 1 / 8.3 ms
  - Average of 8.3 × ½ = 4.1 ms to rotate target sector under head

Seek, Rotate, **Transfer**

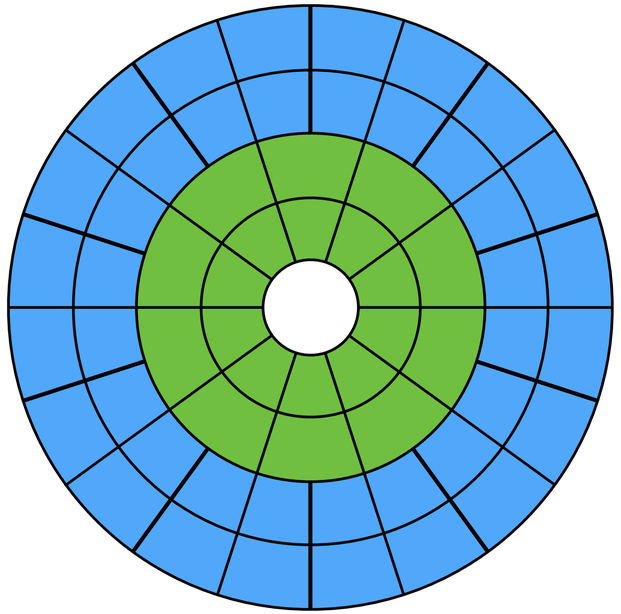
- Depends on RPM and sector density
  - Typical speeds of 100+ MB/s
    - But sustaining this speed is dependent on transferring data from sequential sectors!

Geometry tweaks

- Track skew: allow time for advancing to next sector in adjacent tracks



- Zones: optimize tracks for storage density (outer zones have more sectors/track)



Access characteristics

- Seek & Rotation are time consuming; Transfer is fast
- Sequential disk workloads yield significantly better throughput
  - I.e., contiguous sectors with minimal head movement
- Random workloads are seek & rotation dominated

E.g., random throughput

Given a 7200 RPM HDD with an average seek time of 6 ms and a maximum transfer rate of 200 MB/s, what is the average throughput for random 64 KB (contiguous) disk requests?

- Avg seek time = 6 ms
- Avg rotational latency = (1 / 7200 RPM) × ½ = 4.1 ms
- Avg transfer time = 64 KB / (200 MB/s) = 0.31 ms

- Throughput = 64 KB / (6 + 4.1 + 0.31) = 6.1 KB/ms = **6 MB/s**

§ Disk scheduling

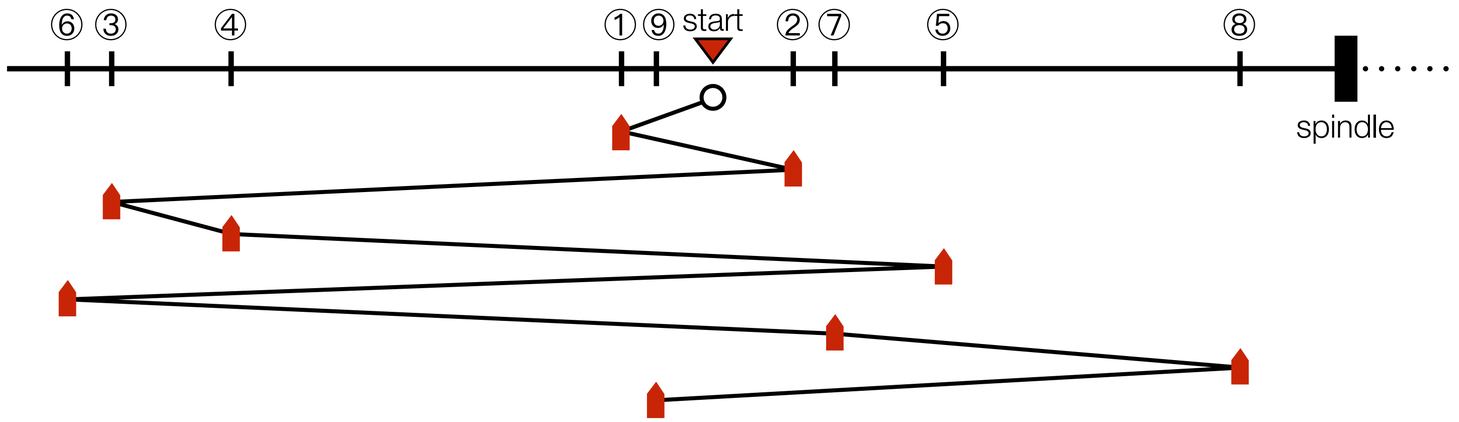
The problem

- Requests for blocks come from many sources (e.g., via the filesystem)
  - In what order should these requests be scheduled?
  - Can we optimize the requests before carrying them out?
  - Where should this scheduler be implemented?
    - The OS?
    - The disk (firmware)?

Goals

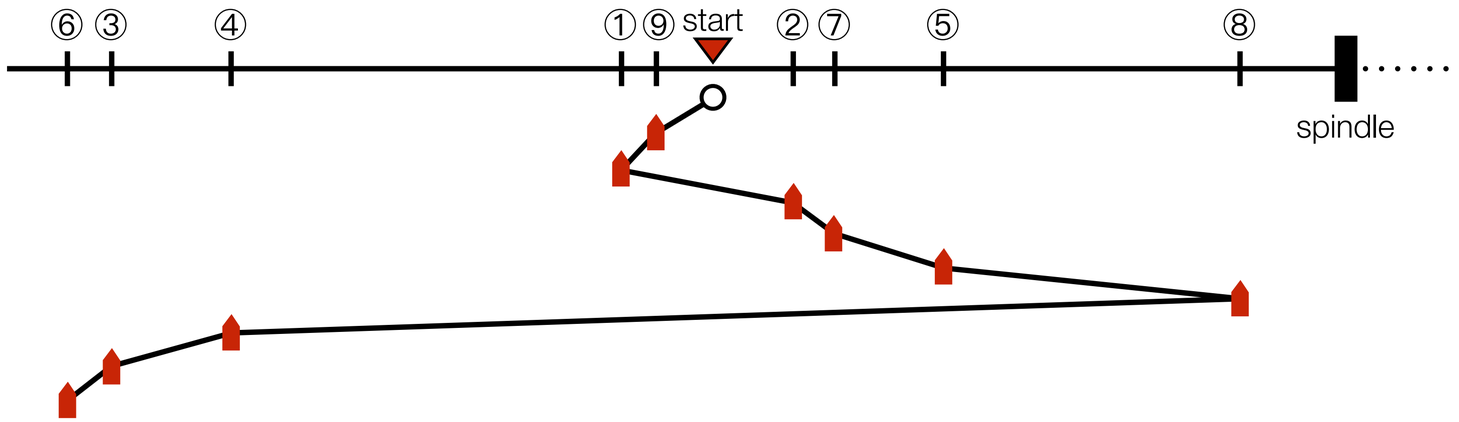
- Maximize throughput
  - This means minimizing seeks and maximizing sequential transfers
  - Maybe avoid going to disk at all, if possible (caching)
- Minimize access latency and avoid starvation
  - Keep in mind that requests are coming in from different processes!

FCFS



- Assuming seek & rotate = 10 ms on average
- Approximately how long to satisfy following sector requests?
  - 10000, 50000, 10001, 50001, 10002, 50002 ≈ 60 ms
  - 10000, 10001, 10002, 50000, 50001, 50002 ≈ 20 ms

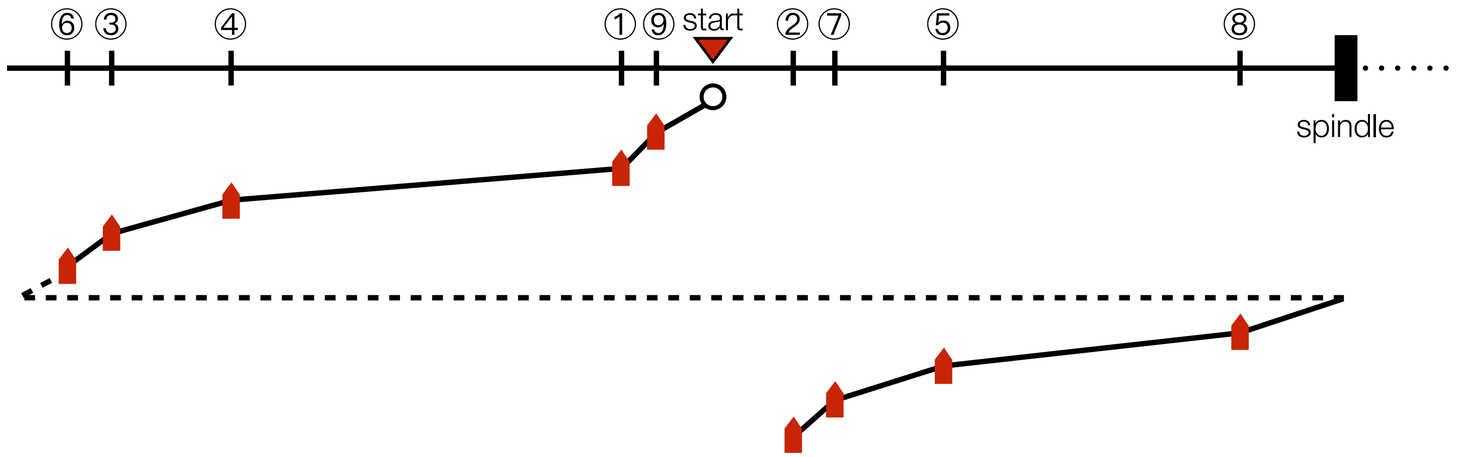
SSTF (shortest seek time first)



- Potential starvation if new requests closer to the head keep arriving
  - Unpredictable latency for individual requests (large standard deviation)
- OS may not have information to implement this precisely (but disk does)

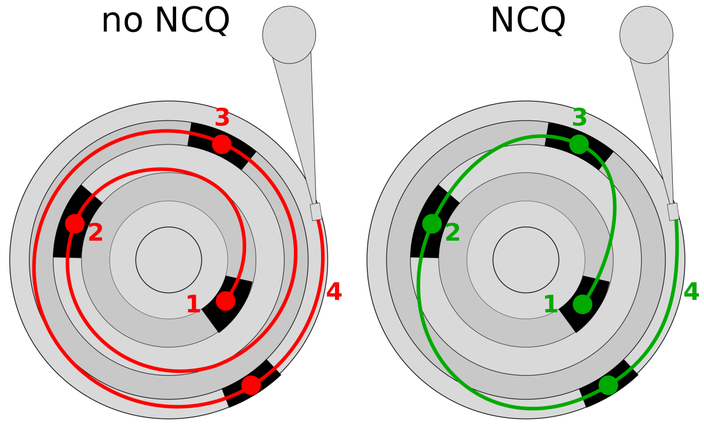
Elevator algorithms

- SCAN and variants
  - Sweep across platter picking up requests — variations include only servicing requests that came in by start of sweep, and “circular” sweep
- E.g., circular sweep:



OS/HDD collaboration

- Balanced disk scheduling leverages both OS and hardware
- OS aims to leverage strengths of HDDs
  - Maintains queue(s) to reorder block requests
  - Merge requests for adjacent blocks
  - Cache(s) to avoid accessing disk when possible
- HDDs are also heavily optimized
  - Native command queueing (NCQ) reorders requests internally
  - Cache for buffering reads/writes



What about SSDs?

- SSDs are much faster than HDDs, especially for random access
  - Render many long-standing filesystem and scheduling optimizations (based on HDDs) irrelevant
  - But so long as a price gap exists between the two, modern OSes will need to support HDDs!

The big picture