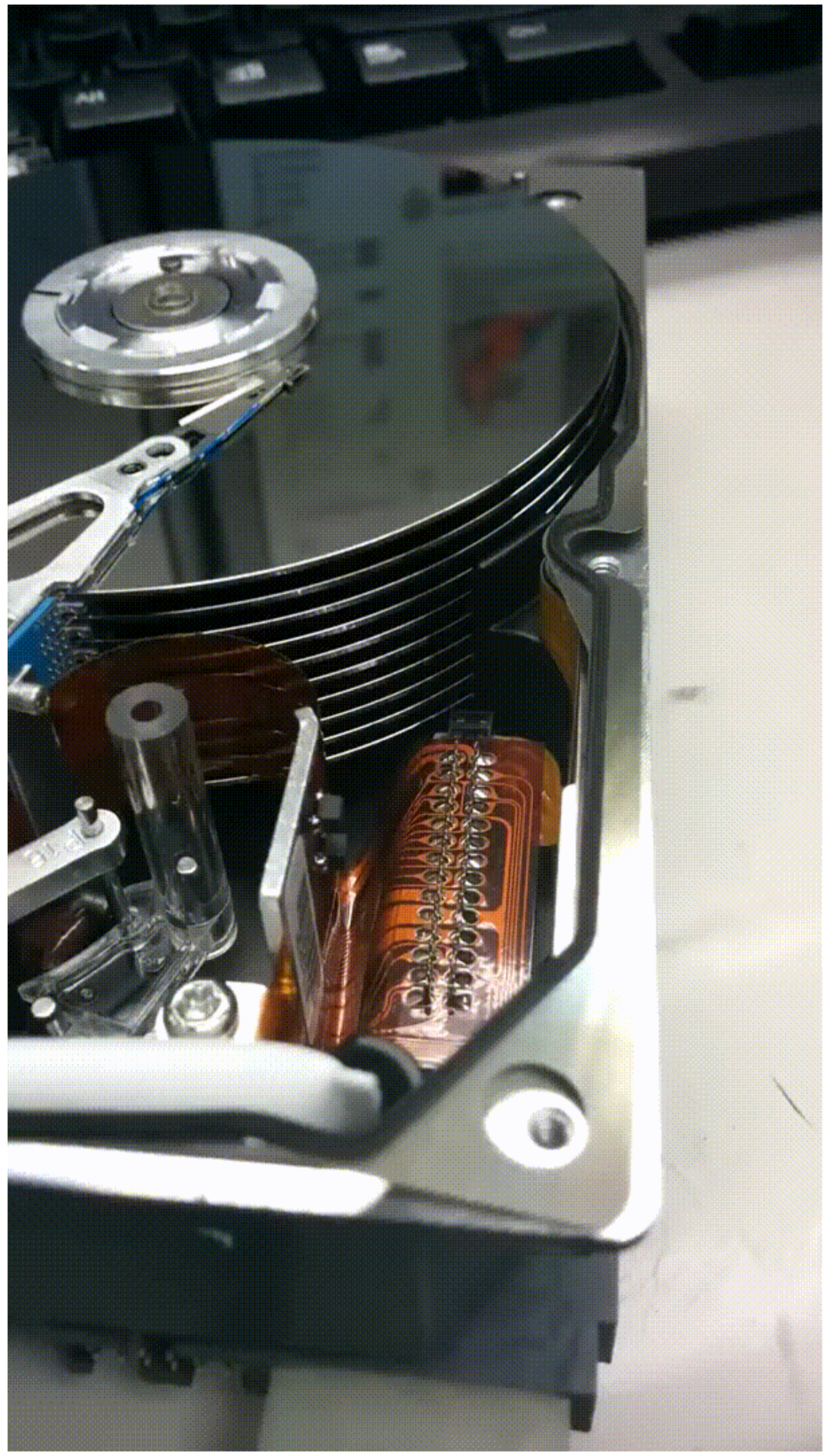
CS202 (003): Operating Systems
Disks

Instructor: Jocelyn Chen

Most of the materials covered in this slide come from the lecture notes of Mike Walfish's CS202

Last Time

Disks

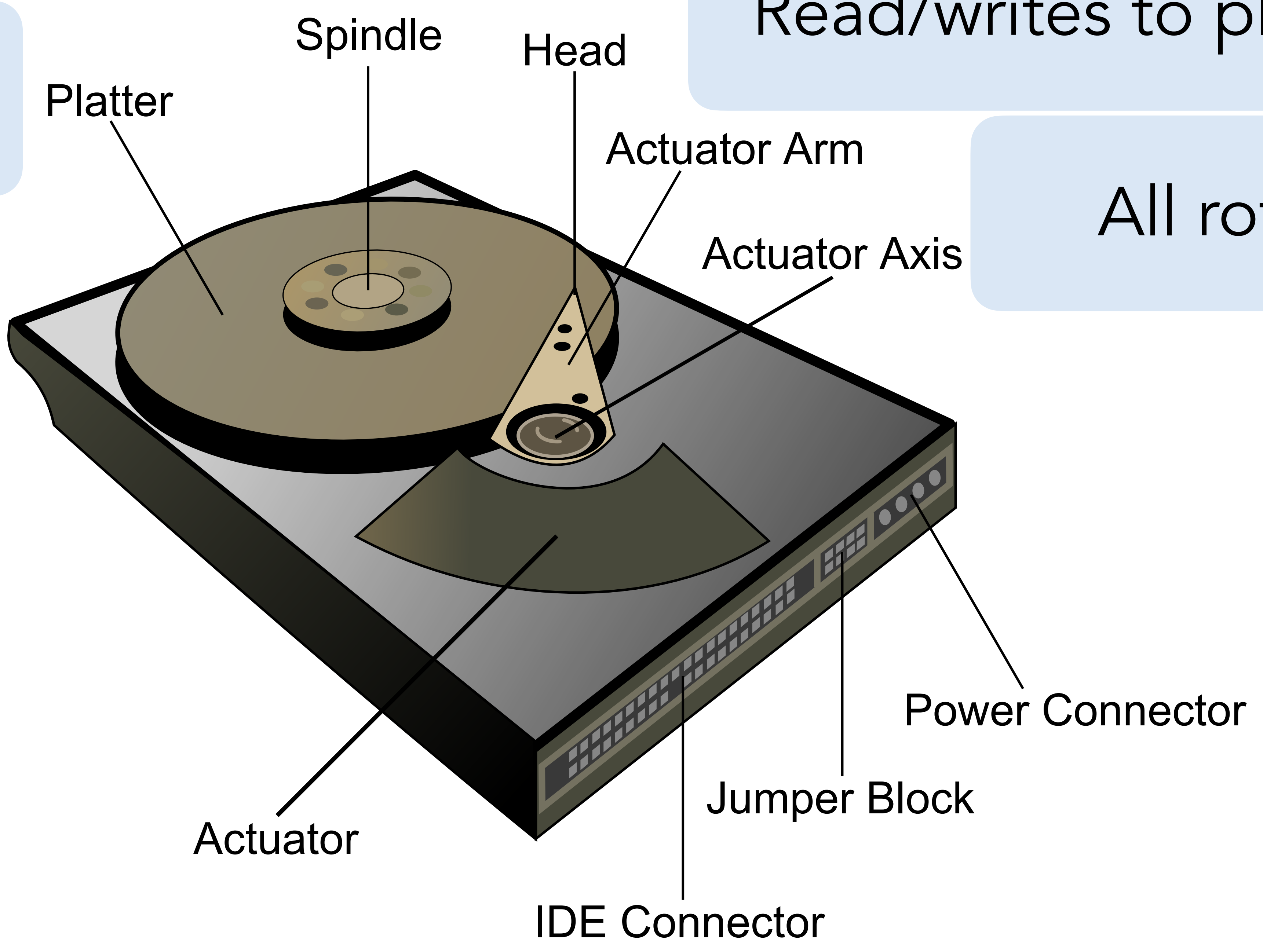


Rotate together on central spindle (3600-15000 RPM)

Stack of magnetic platters

Read/writes to platters

All rotate together

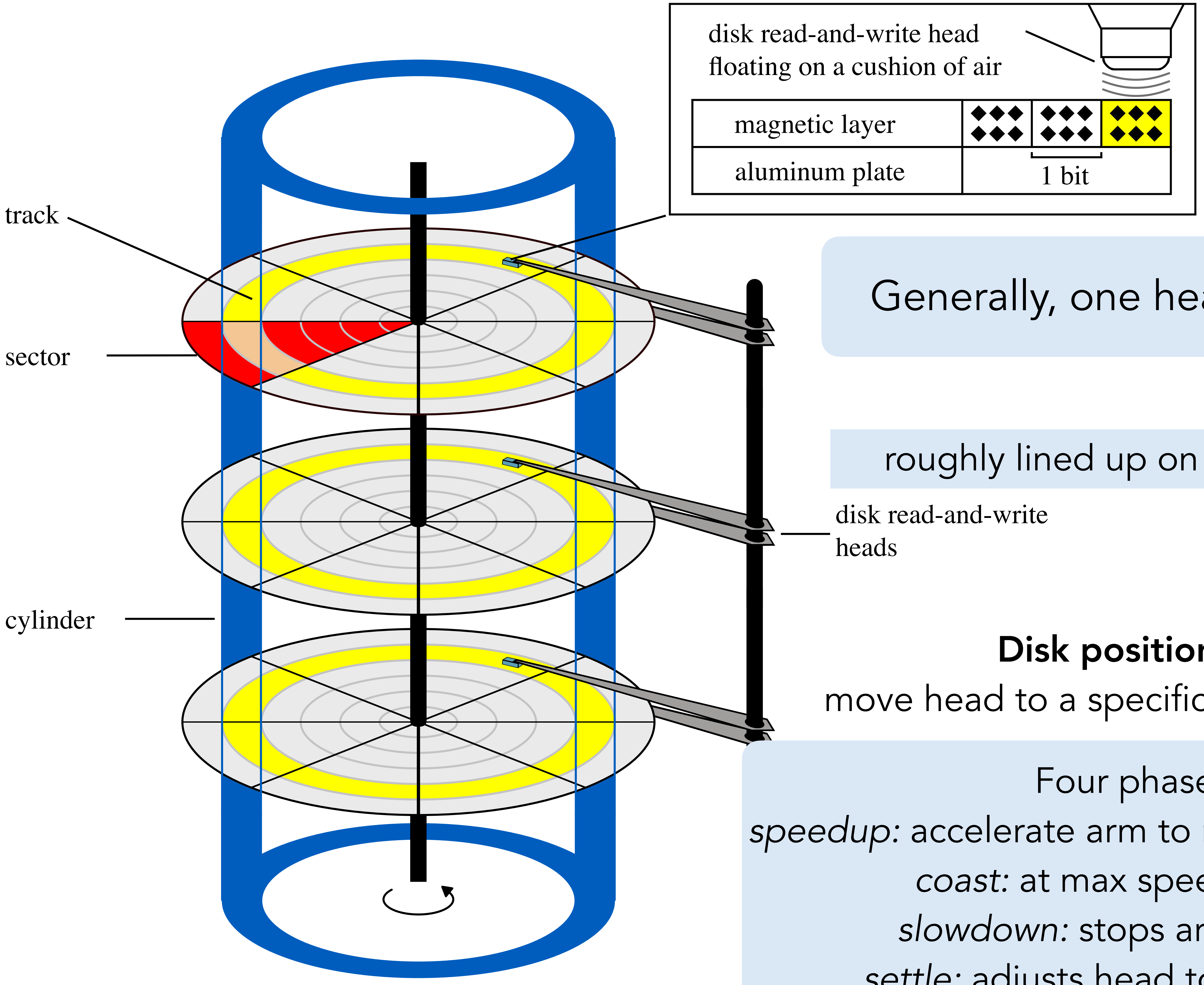


Geometry of a disk

circle on a platter. each platter is divided into concentric tracks

chunk of a track

locus of all tracks of fixed radius on all platters



Generally, one head is active at a time

roughly lined up on a cylinder

Disk positioning system:
move head to a specific track and keep it there

Four phases of **seek:**
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

Performance

Total transfer time

Rotational delay + seek delay + transfer time

Seek

Seeking track-to-track: comparatively fast (~1ms). mainly settle time
Short seeks (200-400 cyl.): dominated by speedup
Longer seeks: dominated by coast
Head switches: comparable to short seeks

Note: settle time takes longer for writes than reads.
(because if read strays, the error will be caught, and the disk can retry,
if the write strays, some other track just got clobbered, so write settles need to be done precisely)

"Average seek time"

For instance:
- Time to seek 1/3 of the disk
- 1/3 of the time to seek the whole disk

Question: are they the same? **No!**

Common #s

Capacity: in TBs

Platters: 8

# of cylinders: >= tens of thousands

RPM: 10,000

Transfer rate: 50-150 MB/s

Mean time between failures: ~1-2 million hours

How driver interfaces to disk?

Sectors

Disk interface presents linear array of sectors
Traditionally 512 bytes (moving to 4KB nowadays)
Written atomically (even if power failure; disk saves enough momentum to complete)
Larger atomic units have to be synthesized by OS (will discuss later)

Disk maps logical sector # to physical sectors

Some optimizations

Zoning: puts more sectors on longer tracks
Track skewing: sector 0 position varies by track, but let the disk worry about it.
Why? (for speed when doing sequential access)
Sparing: flawed sectors remapped elsewhere

OS has no idea what is going on with all these!

Disk performance example

Spindle Speed: 7200 RPM
Avg Seek Time, read/write: 10.5ms / 12 ms
Transfer rate (surface to buffer): 54 MB/s

How long would it take to do 500 sector reads, spread out randomly over the disk (and serviced in FIFO order)?

```
(rotation delay + seek time + transfer_time)*500
rotation delay: 60s/1min * 1 min/7200 rotations = 8.33 ms
  on average, you have to wait for half a rotation: 4.15 ms
seek time: 10.5 ms (given)
transfer time: 512 bytes * 1 s/54 MB * 1MB/10^6 bytes = .0095 ms
**per read**: 4.15 ms + 10.5 ms + .0095 ms = 14.66 ms
500 reads: 14.66 ms/request * 500 requests = 7.3 seconds.
total throughput: data/time = 35KB/s
```

Disk performance example

Spindle Speed: 7200 RPM
Avg Seek Time, read/write: 10.5ms / 12 ms
Transfer rate (surface to buffer): 54 MB/s

How long would it take to do 500 sector reads, **sequentially on the disk** (and serviced in FIFO order)?

```
rotation delay + seek time + 500*transfer_time
rotation delay: 4.15 ms (same as above)
seek time: 10.5 ms (same as above)
transfer time: 500 * .0095 ms = 4.75 ms
total: 4.15 ms + 10.5 ms + 4.75 ms = 19.5 ms
total throughput: 13.1 MB/s
```

Sequential vs. Random Reads

Sequential reads are **MUCH** faster than random reads!

So what do we do?

"The secret to making disks fast is to treat them like tape"

Dish Cache for Read-ahead

Disk keeps reading at last host request otherwise sequential read would incur whole revolution

Should read-ahead cross track boundaries?
a head-switch cannot be stopped, so there is a cost to aggressive read ahead.

Write Caching

(if battery backed): data in buffer can be written over many times before actually being put back to disk. also, many writes can be stored so they can be scheduled more optimally
(if not battery backed): then policy decision between disk and host about whether to report data in cache as on disk or not

Minimize seek times

The system (OS or disk controller) can reorder pending I/O requests to minimize head movement

Requirements for effective request ordering:

- Multiple pending I/O requests must be available
- System must support I/O concurrency
- More requests = better optimization opportunities

- **Strategy 1: Maximize I/O Concurrency:**
  - Issue multiple I/O requests simultaneously
  - Allows the system to optimize request ordering
- **Strategy 2: Memory-Centric Design:**
  - Keep primary data structures in memory
  - Use write-logging for persistence
  - Write backups sequentially to disk
  - Avoid random-access reads entirely

Technology and System Trends

Disk Performance	Mechanics of disks (seeks and rotational delays) have not kept up with huge growth in other computer components (CPU, RAM, …) However, data transfer bandwidth has shown steady improvement at roughly 10x per decade
Storage Density	**Density is growing fast!** (because it is less about mechanical limits) **Key:** minimizing the distance between the read/write head and disk surface (well, what happen if the head touch the surface?)
Addressing Disk Access Bottleneck	Leverage increased bandwidth to fetch larger chunks of data per access Trade higher latency for better bandwidth utilization Optimize data placement by clustering related data physically close together on disk (This clustering allows efficient retrieval of related data once the initial seek cost is paid)
Memory Size Impact	System memory (RAM) size is growing faster than typical workload sizes, leading to: • More data fitting in file cache • Changed disk access patterns: predominantly writes and new data access • Makes logging and journaling more practical as performance strategies
Cloud Computing Impact	Allow decoupling computer from storage: Small CPUs with a lot of storage attached

Remarks about Disks

HDD have historically been the bottleneck in many systems

Although this becomes less and less true every year
(because of SSDs, PM, …)

Hmmm, so why are we studying them?
Disks are still widely used (cheap, better durability than SSD, great for backup), especially in large cloud infrastructure

Many filesystems were designed with disk in mind
(sequential access throughput much higher than random access)

Pattern: large setup costs, followed by efficient batch transfer, shows up in a lot of hardware and systems

Quiz Time!