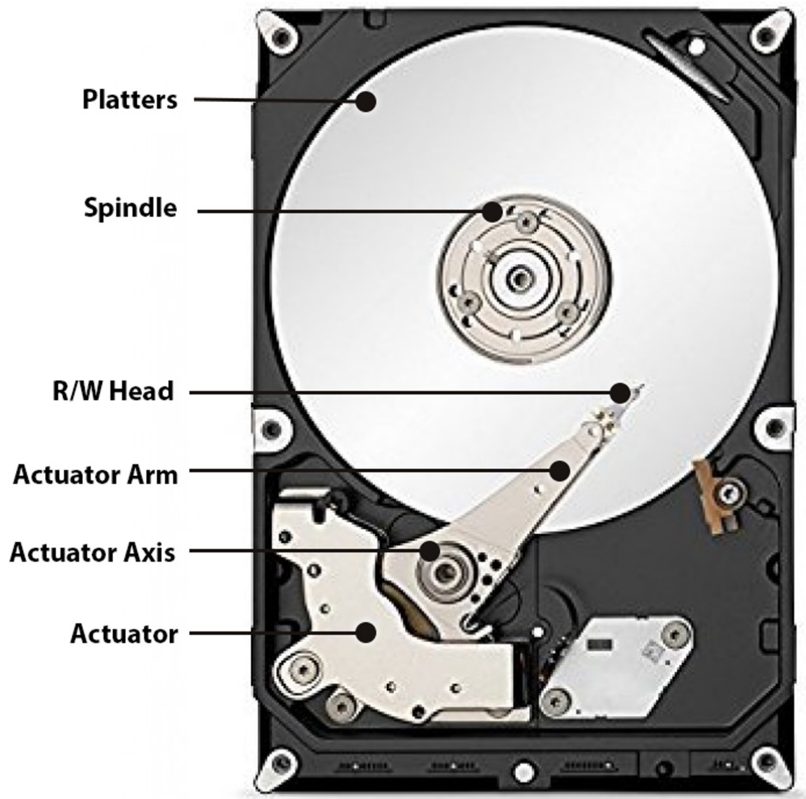
377 Operating Systems

File System Implementation

Disks

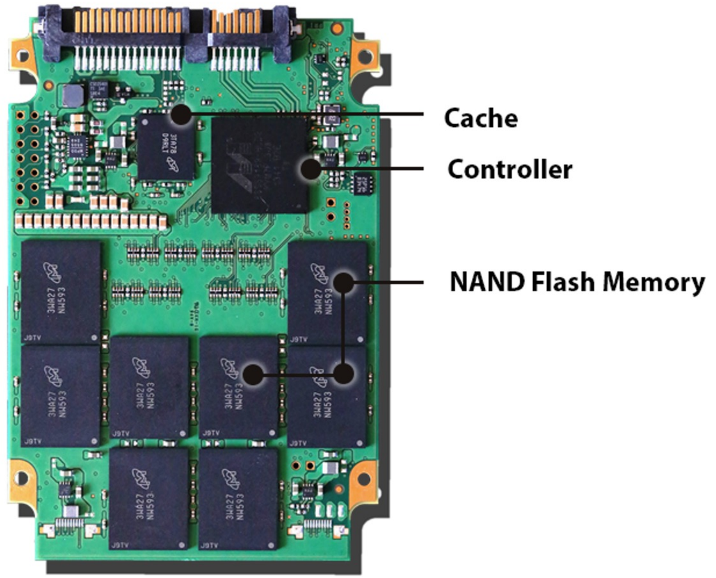
- Disks are a crucial part of computing systems
- This is where we save persistent data and swap memory pages to
- We find two primary forms: Hard Disk Drives (aka spinning rust) and Solid State Drives (SSD)
- SSDs are plainly better on performance
- HDDs 2x-3x cheaper

HDD
3.5”



Shock resistant up to 55g (operating)
Shock resistant up to 350g (non-operating)

SSD
2.5”



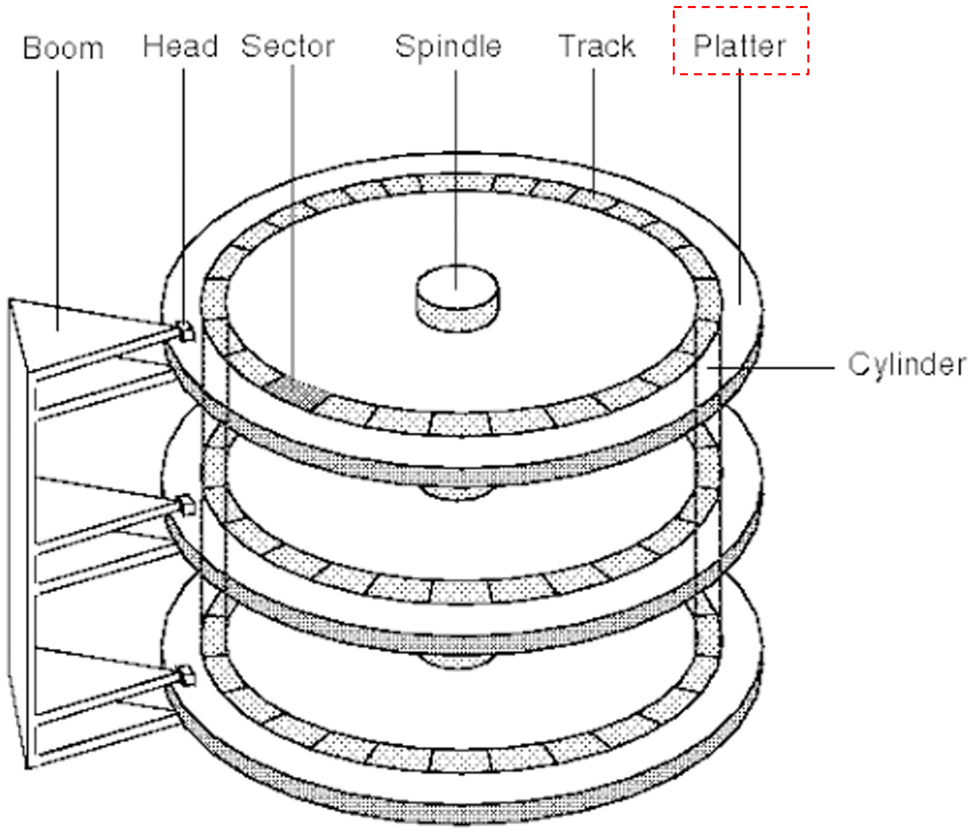
Shock resistant up to 1500g
(operating and non-operating)

Disk Abstraction

- The disk gives us an address space abstraction, much like physical memory
- The drive consists of N sectors (called blocks) of some size, typically 512B
- Unlike RAM, you have to read/write the whole block
- A single block write is atomic (this is really important later!)

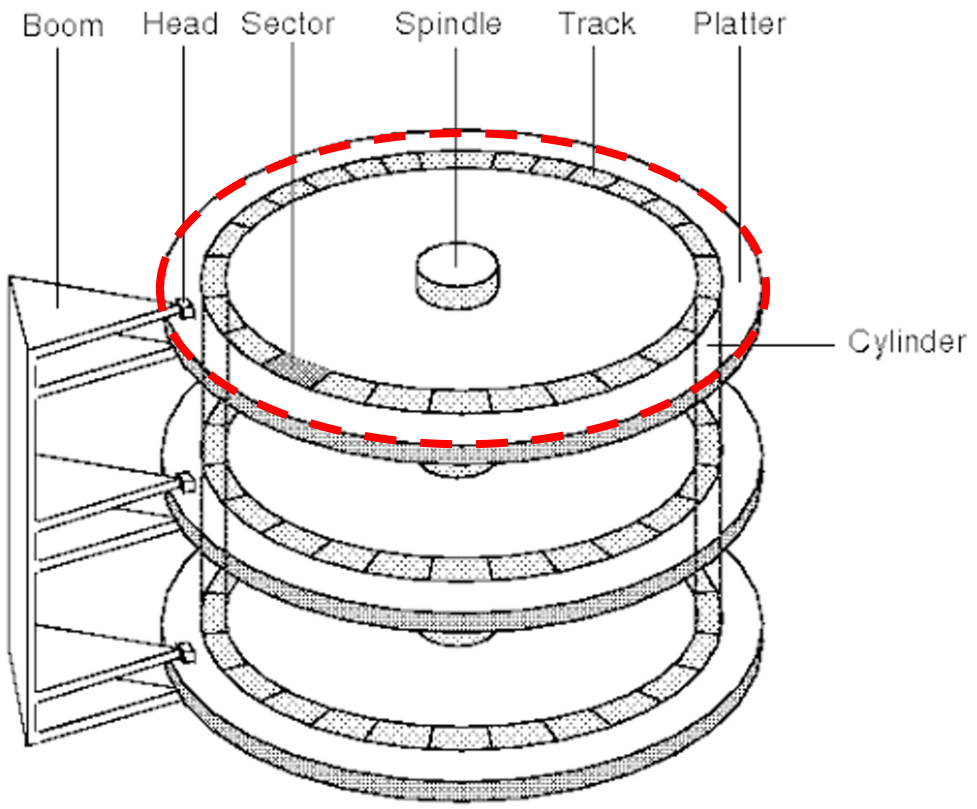
Hard Disk Geometry

- Platter (the physical disk)



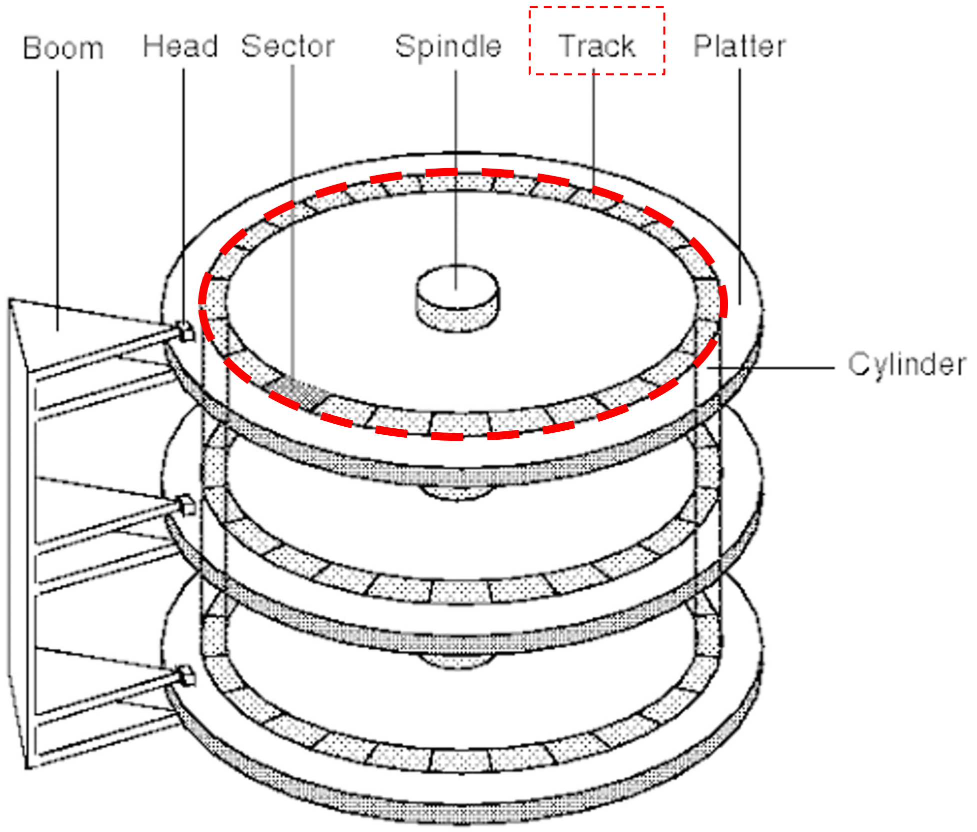
Hard Disk Geometry

- Platter (the physical disk)
- Surface (one side of the disk)



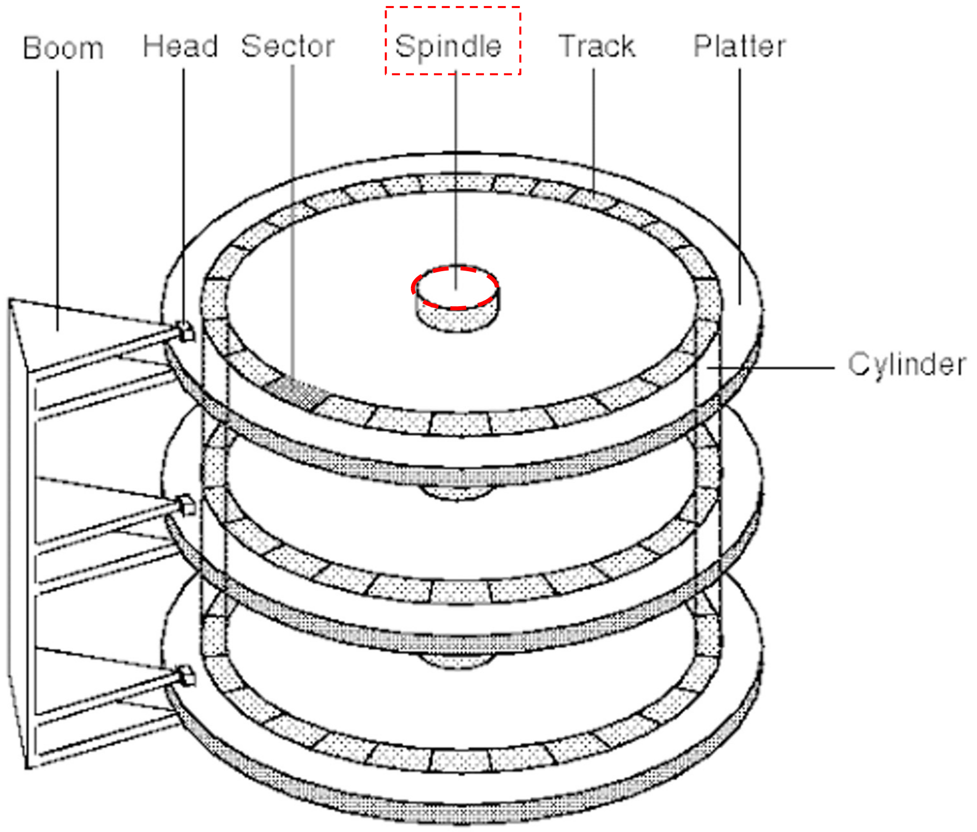
Hard Disk Geometry

- Platter (the physical disk)
- Surface (one side of the disk)
- Track: Concentric circles on surface



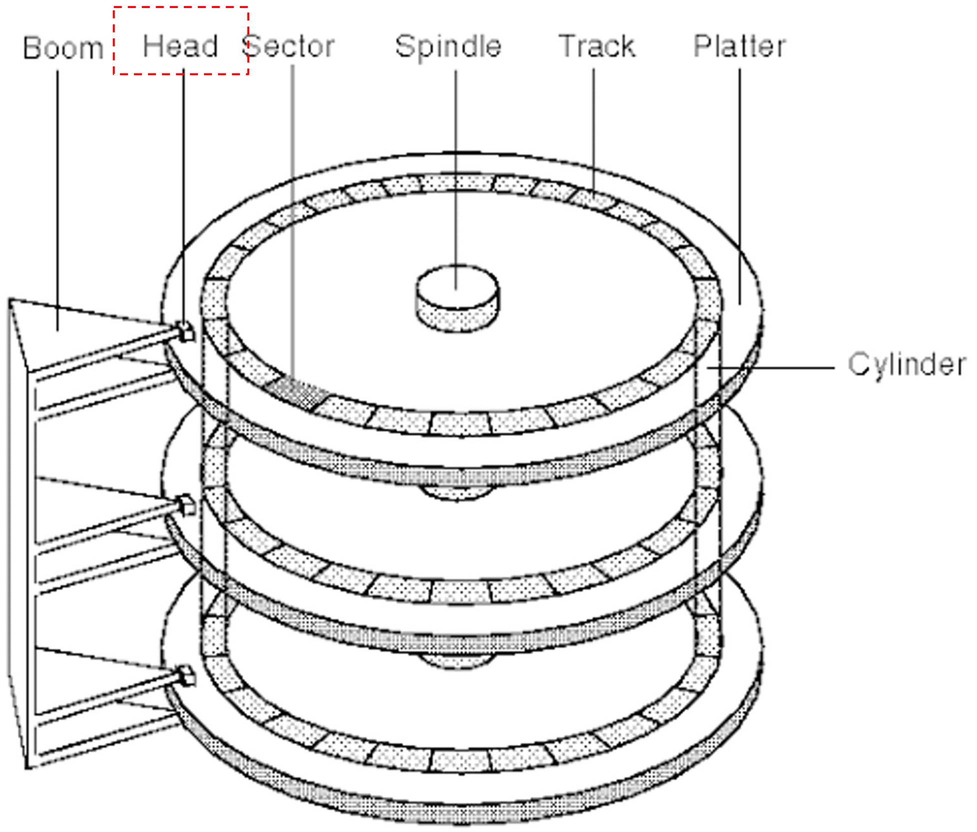
Hard Disk Geometry

- Platter (the physical disk)
- Surface (one side of the disk)
- Track: Concentric circles on surface
- Spindle (the spinning motor shaft, typical 5k-10k RPM)



Hard Disk Geometry

- Platter (the physical disk)
- Surface (one side of the disk)
- Track: Concentric circles on surface
- Spindle (the spinning motor shaft, typical 5k-10k RPM)
- Head: attached to an Arm/Boom



A Single Track With a Head

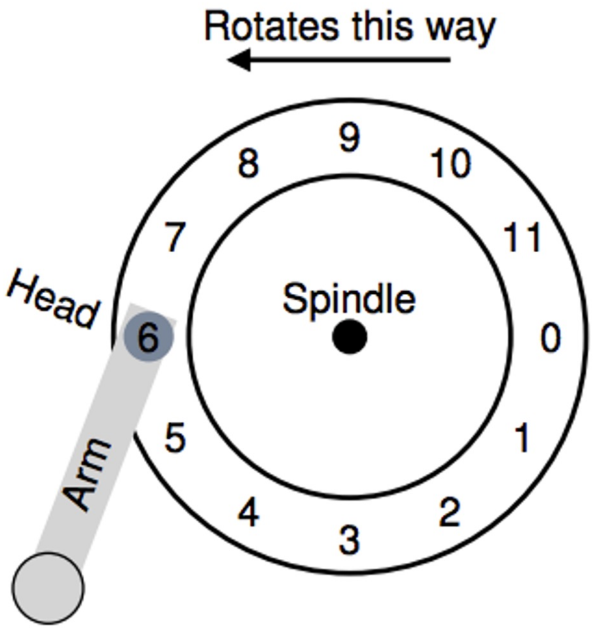
A simple disk with a single track.

This track has 12 sectors, each of which is 512 bytes and addressed therefore by the numbers 0 through 11.

The single platter we have here rotates around the spindle, to which a motor is attached.

A disk head attached to a disk arm performs the read/write operations to a sector/block.

In this figure, the disk head is positioned over sector 6, and the surface is rotating counter-clockwise.



Single-track Latency: The Rotational Delay

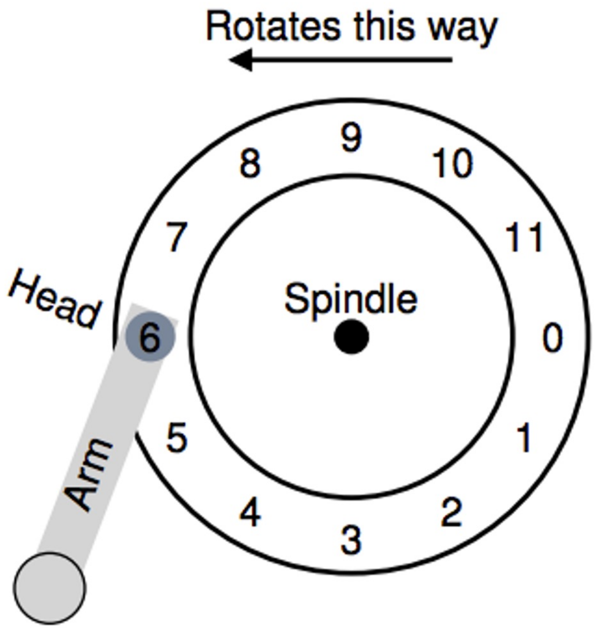
Imagine we receive a request to read block 0.

For this simple disk it doesn't need to do much except *wait* for sector 0 to appear under the head.

This waiting time is called **rotational delay**.

In this example, if the full rotational delay is *R* then the disk incurs a rotational delay of *R/2* to wait for sector 0 to be under the head.

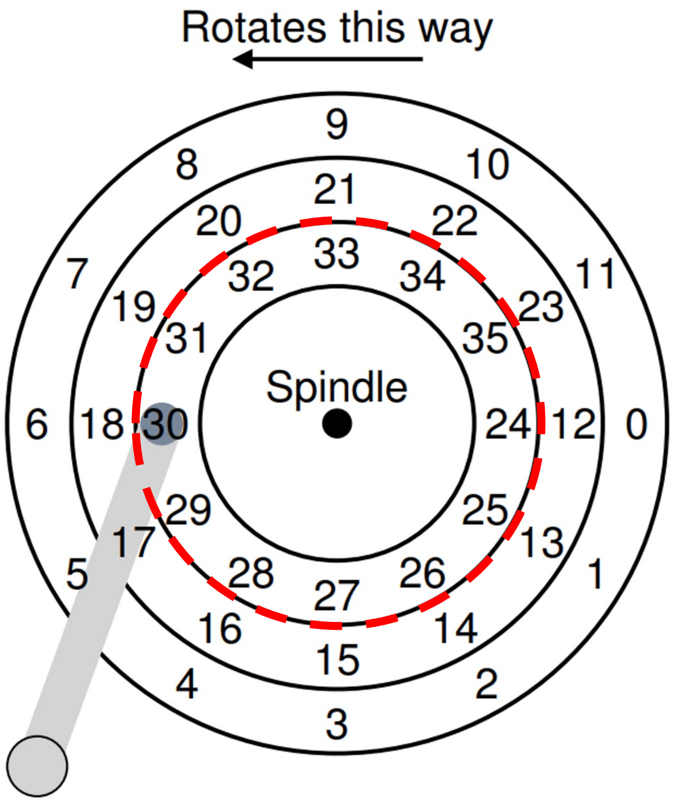
A worst-case request on this single track would be to sector 5.



Multi-track Latency: Seek Time

On the right is a more interesting example of a disk containing 3 tracks.

In the figure, the head is positioned over the innermost track, containing sectors 24-35.

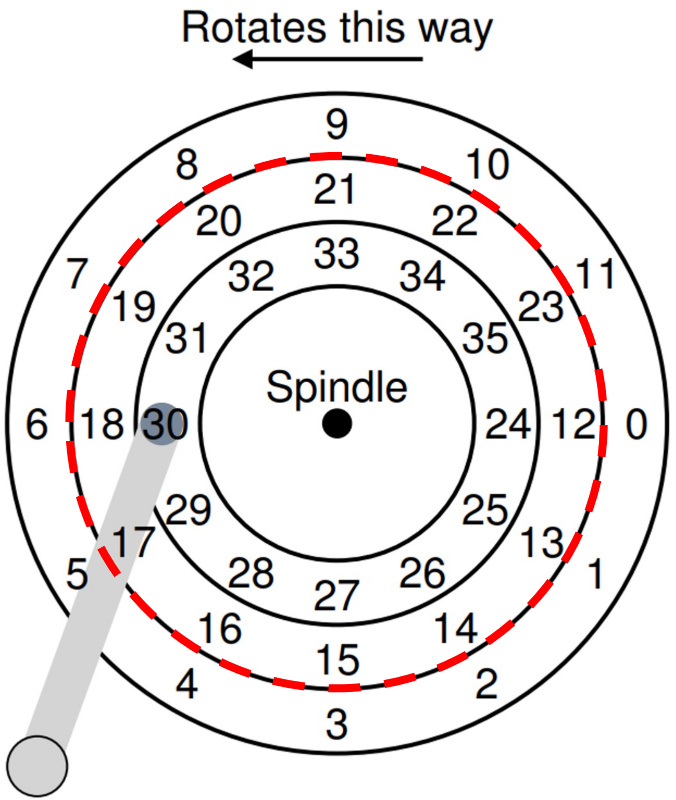


Multi-track Latency: Seek Time

On the right is a more interesting example of a disk containing 3 tracks.

In the figure, the head is positioned over the innermost track, containing sectors 24-35.

The next track over contains sectors 12-23.



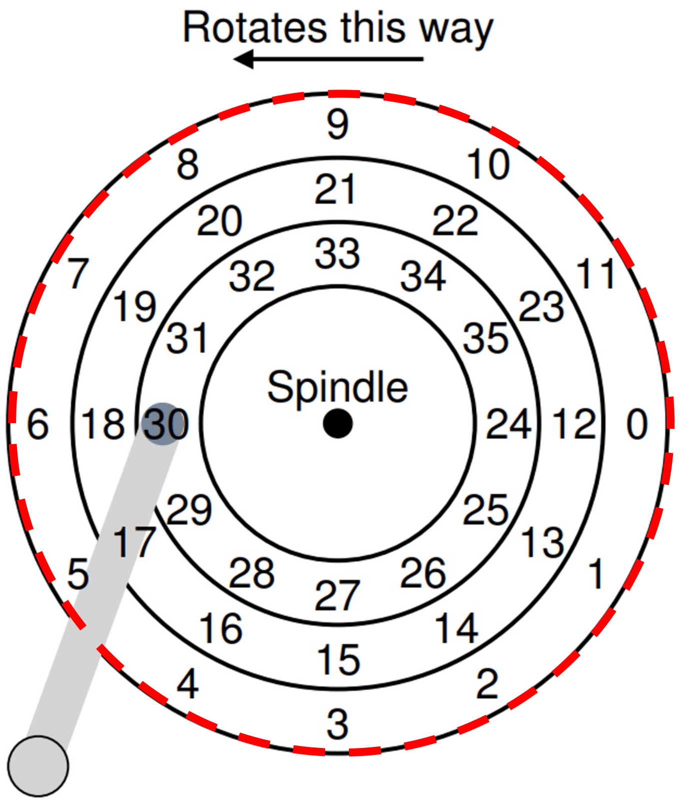
Multi-track Latency: Seek Time

On the right is a more interesting example of a disk containing 3 tracks.

In the figure, the head is positioned over the innermost track, containing sectors 24-35.

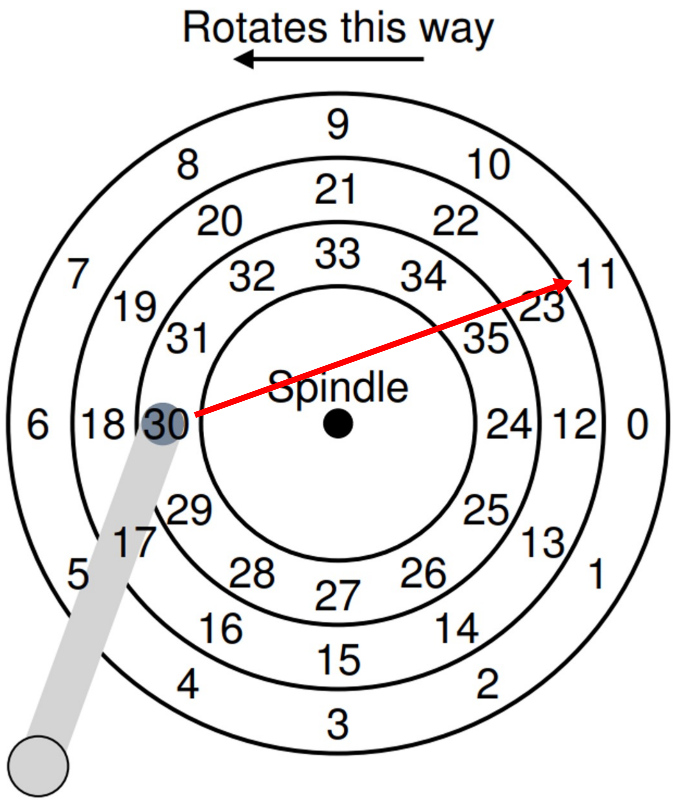
The next track over contains sectors 12-23.

The outermost track contains sectors 0-11.



Multi-track Latency: Seek Time

What if the disk received a read request for a distant sector such as 11?

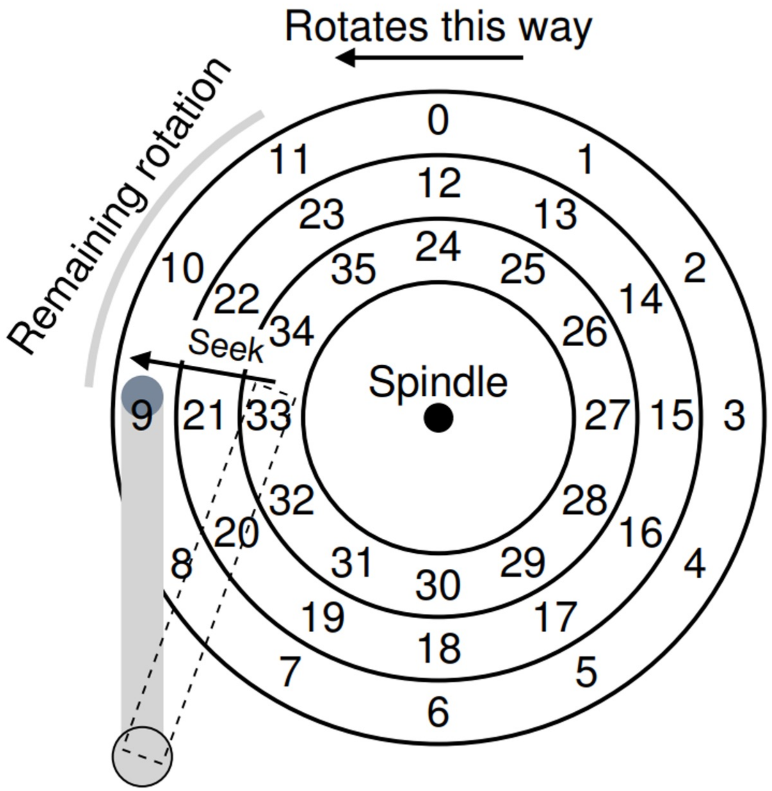


Multi-track Latency: Seek Time

What if the disk received a read request for a distant sector such as 11?

To service this read, the drive first has to move the disk arm to the correct track.

This process is known as a **seek**.



Multi-track Latency: Seek Time

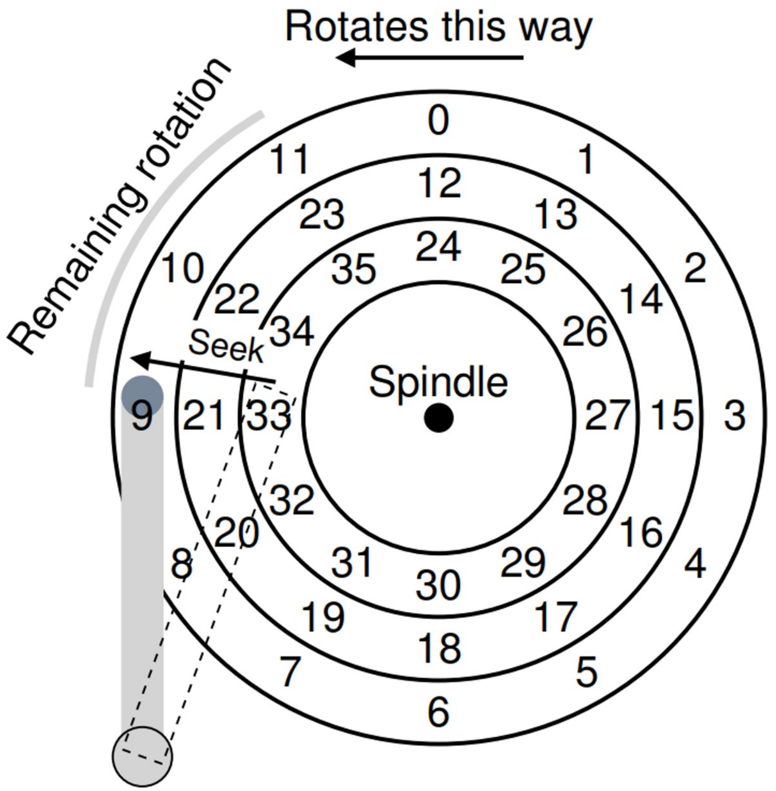
What if the disk received a read request for a distant sector such as 11?

To service this read, the drive first has to move the disk arm to the correct track.

This process is known as a **seek**.

After the seek, the head is over the right track. The disk is rotating at the same time as the seek.

As you can see in this example, the head is over sector 9 and must rotate a little more until it is over sector 11.

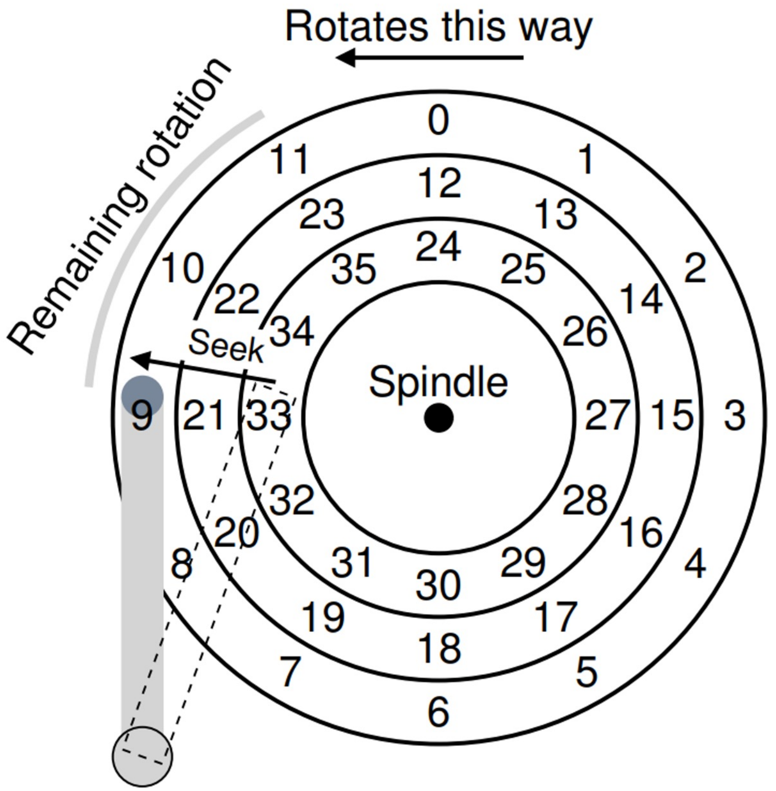


Multi-track Latency: Transfer Time

When sector 11 finally arrives under the head the final phase of I/O will take place.

This is known as the **transfer**, where data is either read/written to the surface.

So, the amount of time it takes to perform I/O on a disk is the *seek time*, *rotational latency*, and the *transfer time*.



I/O Time

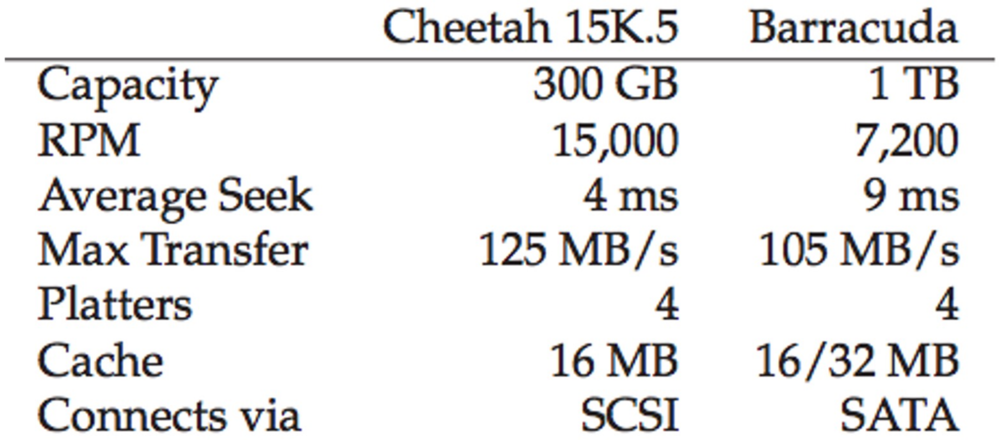
$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

The time to perform I/O is the seek time (T_{seek}) plus the rotation time ($T_{rotation}$) plus the time it takes to transfer the data ($T_{transfer}$).

Cache

- Frequently the disk also contains a cache.
- This allows the disk to do things like prefetch data from the disk

Example Disks



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

A “performance” disk vs a capacity disk. The one on the left is more $$$.

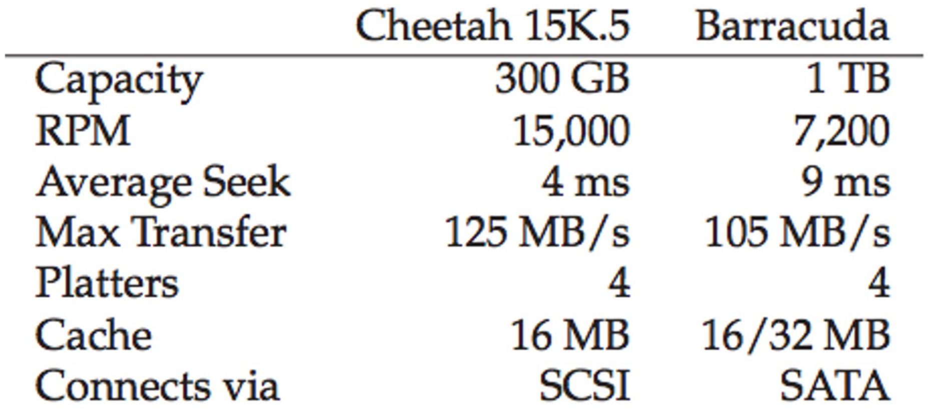
Two Sample Workloads

- **Random**: small, 4KB requests from across the disk
  - Might find this in a DB
- **Sequential**: reads a large number of adjacent sectors
  - copying large files, watching a movie, uploading etc.

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

First, let us compute the average seek time.



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

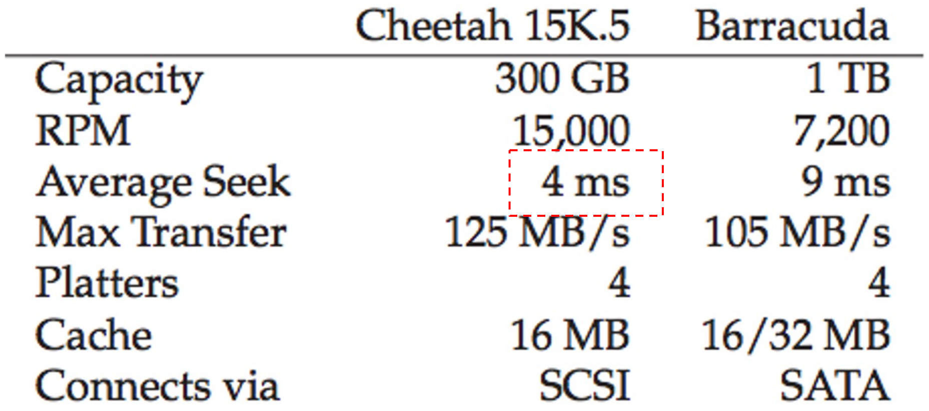
$$T_{I/O} = \underset{\boxed{\quad}}{T_{seek}} + \underset{\boxed{\quad}}{T_{rotation}} + \underset{\boxed{\quad}}{T_{transfer}}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

First, let us compute the average seek time.

This is easy, as it is given in the manufacturers description!



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = \underbrace{T_{seek}}_{\boxed{4ms}} + \underbrace{T_{rotation}}_{\boxed{}} + \underbrace{T_{transfer}}_{\boxed{}}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Next, let us figure out the average rotational delay. This requires some dimensional analysis!

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

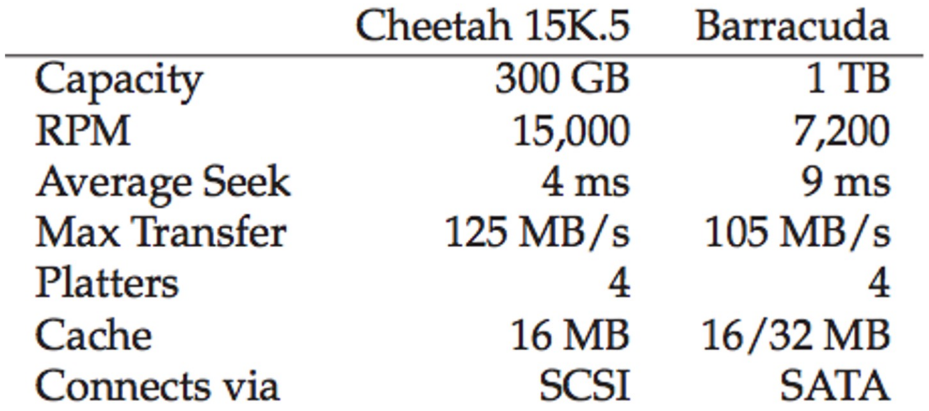
$$T_{I/O} = \underset{\boxed{4ms}}{T_{seek}} + \underset{\boxed{\quad}}{T_{rotation}} + \underset{\boxed{\quad}}{T_{transfer}}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Next, let us figure out the average rotational delay. This requires some dimensional analysis!

Time (ms) / 1 rotation



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

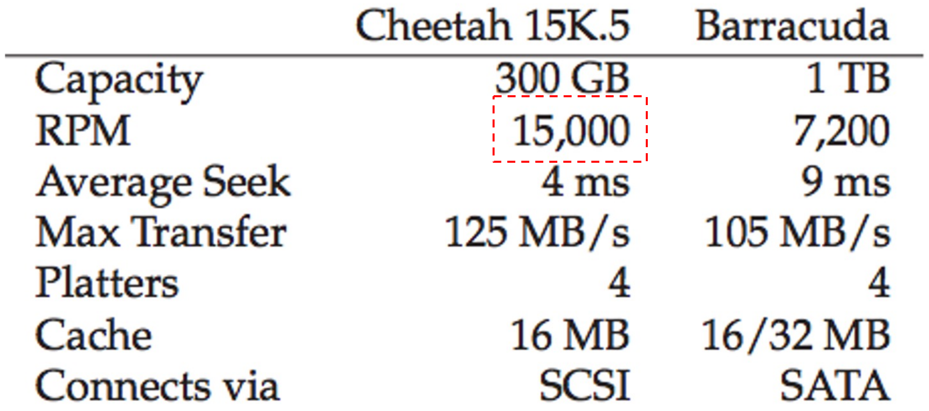
$$\boxed{4ms} \quad \boxed{} \quad \boxed{}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Next, let us figure out the average rotational delay. This requires some dimensional analysis!

Time (ms) / 1 rotation
= 1 min / 15,000 rot *



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

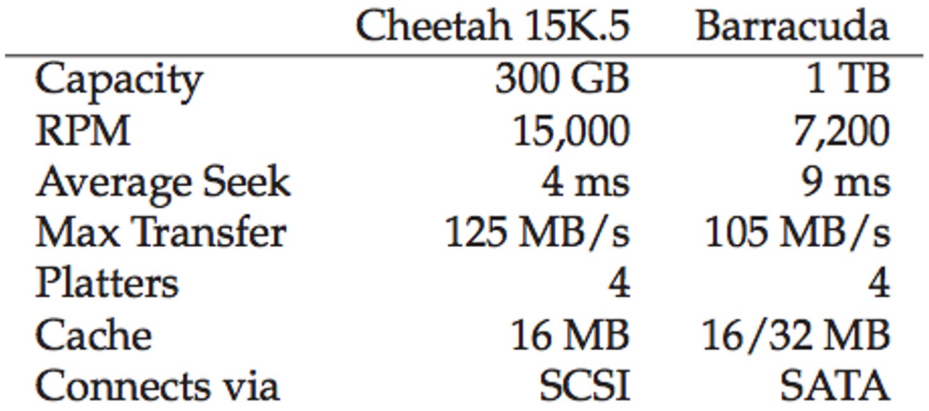
$$T_{I/O} = \underset{\boxed{4ms}}{T_{seek}} + \underset{\boxed{}}{T_{rotation}} + \underset{\boxed{}}{T_{transfer}}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Next, let us figure out the average rotational delay. This requires some dimensional analysis!

Time (ms) / 1 rotation
= 1 min / 15,000 rot *
60 sec / 1 min *



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

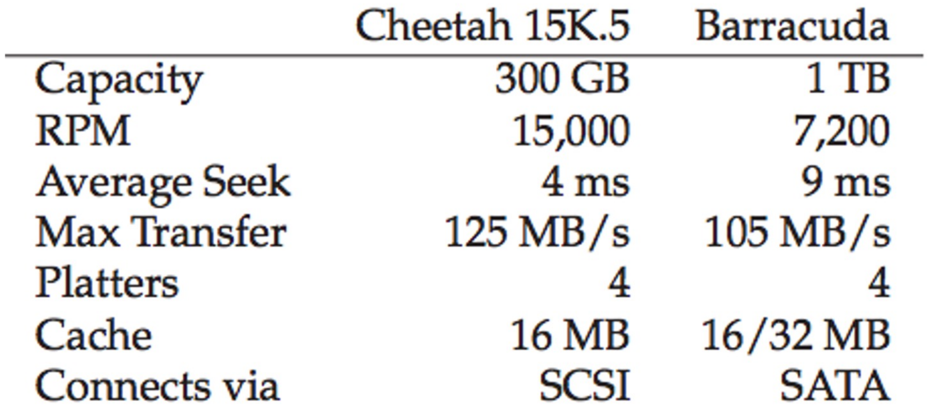
$$T_{I/O} = \underset{\boxed{4\text{ms}}}{T_{seek}} + \underset{\boxed{}}{T_{rotation}} + \underset{\boxed{}}{T_{transfer}}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Next, let us figure out the average rotational delay. This requires some dimensional analysis!

Time (ms) / 1 rotation
= 1 min / 15,000 rot *
  60 sec / 1 min *
  1,000 ms / 1 sec



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

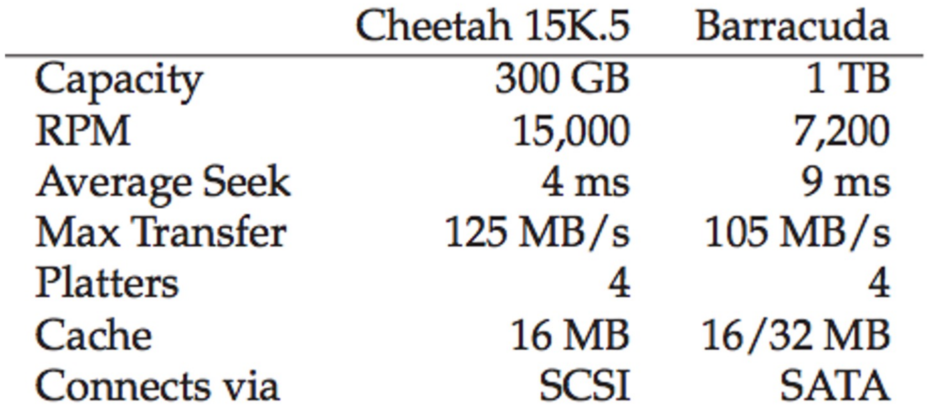
$$\boxed{4ms} \quad \boxed{} \quad \boxed{}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Next, let us figure out the average rotational delay. This requires some dimensional analysis!

Time (ms) / 1 rotation
= 1 min / 15,000 rot *
60 sec / 1 min *
1,000 ms / 1 sec
= 60,000 ms / 15,000 rot = 4ms



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = \underset{\boxed{\text{4ms}}}{T_{seek}} + \underset{\boxed{}}{T_{rotation}} + \underset{\boxed{}}{T_{transfer}}$$

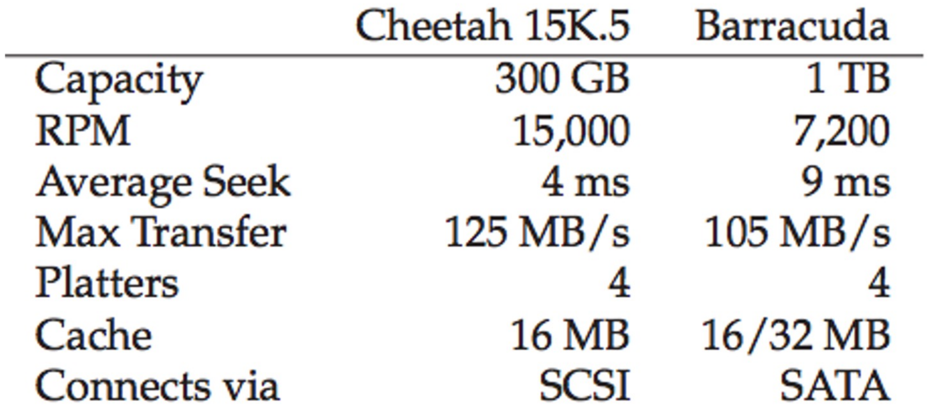
Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Next, let us figure out the average rotational delay. This requires some dimensional analysis!

Time (ms) / 1 rotation
= 1 min / 15,000 rot *
  60 sec / 1 min *
  1,000 ms / 1 sec
= 60,000 ms / 15,000 rot = 4ms

Average time per rotation is 4ms/2 = 2ms



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = \underset{\boxed{4ms}}{T_{seek}} + \underset{\boxed{2ms}}{T_{rotation}} + \underset{\boxed{}}{T_{transfer}}$$

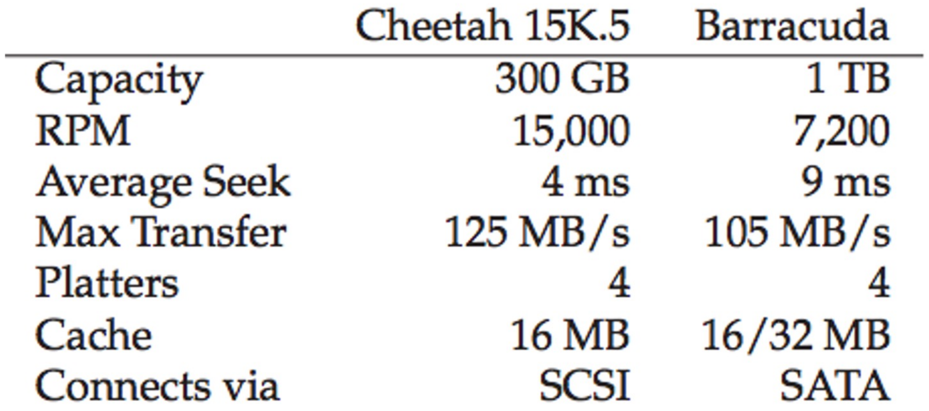
Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Lastly, let us calculate the *transfer time*.

The transfer time is the size of the transfer over the peak transfer rate.

4KB/(125MB/1sec)



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = \underset{\boxed{4ms}}{T_{seek}} + \underset{\boxed{2ms}}{T_{rotation}} + \underset{\boxed{}}{T_{transfer}}$$

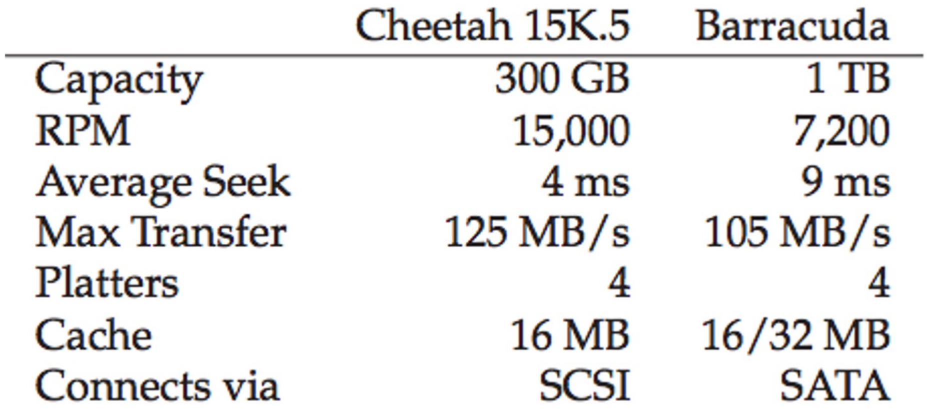
Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Lastly, let us calculate the *transfer time*.

The transfer time is the size of the transfer over the peak transfer rate.

4KB/(125MB/1sec)
= 4KB/1 * 1sec/125MB



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = \underbrace{T_{seek}}_{\boxed{4ms}} + \underbrace{T_{rotation}}_{\boxed{2ms}} + \underbrace{T_{transfer}}_{\boxed{}}$$

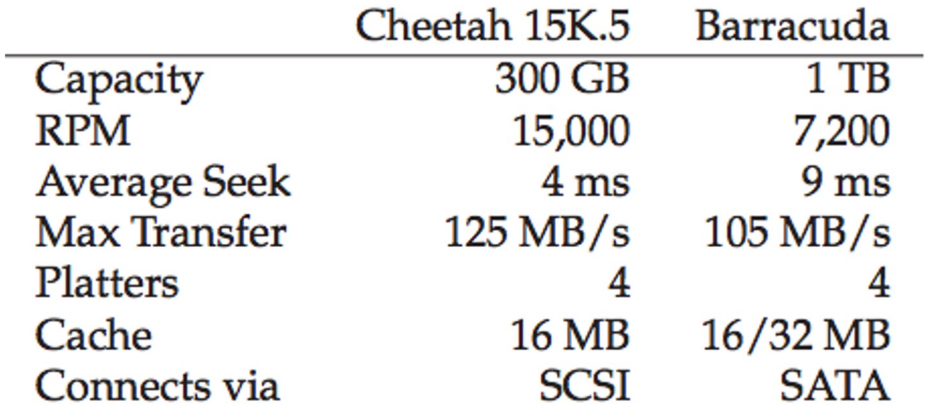
Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Lastly, let us calculate the *transfer time*.

The transfer time is the size of the transfer over the peak transfer rate.

4KB/(125MB/1sec)
= 4KB/1 * 1sec/125MB *
   1MB/1024KB *



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = \underbrace{T_{seek}}_{\boxed{4ms}} + \underbrace{T_{rotation}}_{\boxed{2ms}} + \underbrace{T_{transfer}}_{\boxed{\sim 30\mu s}}$$

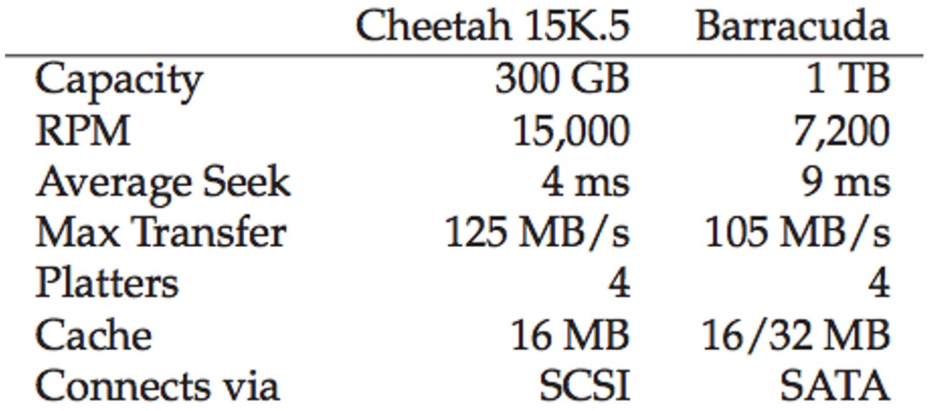
Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Lastly, let us calculate the *transfer time*.

The transfer time is the size of the transfer over the peak transfer rate.

4KB/(125MB/1sec)
= 4KB/1 * 1sec/125MB *
  1MB/1024KB *
  1000ms/1sec * 1000µs/1ms
= about 30µs



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

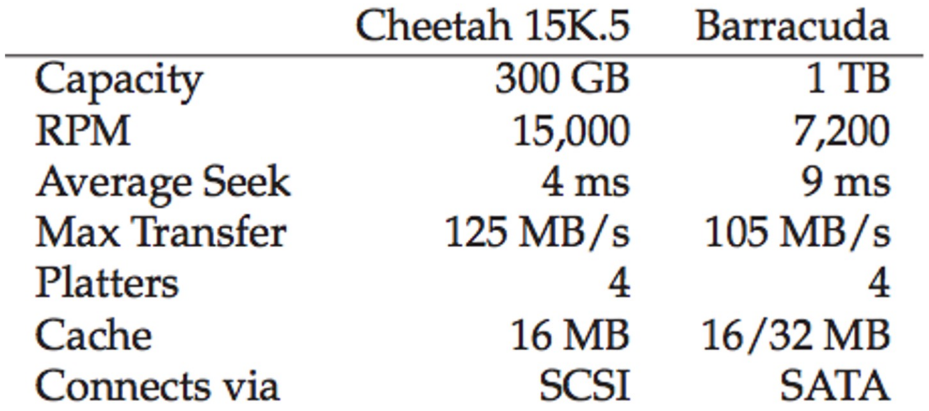
$$T_{I/O} = \underbrace{T_{seek}}_{\boxed{4\text{ms}}} + \underbrace{T_{rotation}}_{\boxed{2\text{ms}}} + \underbrace{T_{transfer}}_{\boxed{\sim 30\mu\text{s}}}$$

Example: Calculate I/O Time Random WL (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

Now, we have all the parts we need to compute the total I/O time for the Cheetah under a random workload transferring 4KB requests.

The transfer time is so small it isn't even worth including. So, the total time is about 6ms.



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

$$\boxed{\sim 6\text{ms}} \quad \boxed{4\text{ms}} \quad \boxed{2\text{ms}} \quad \boxed{\sim 30\mu\text{s}}$$

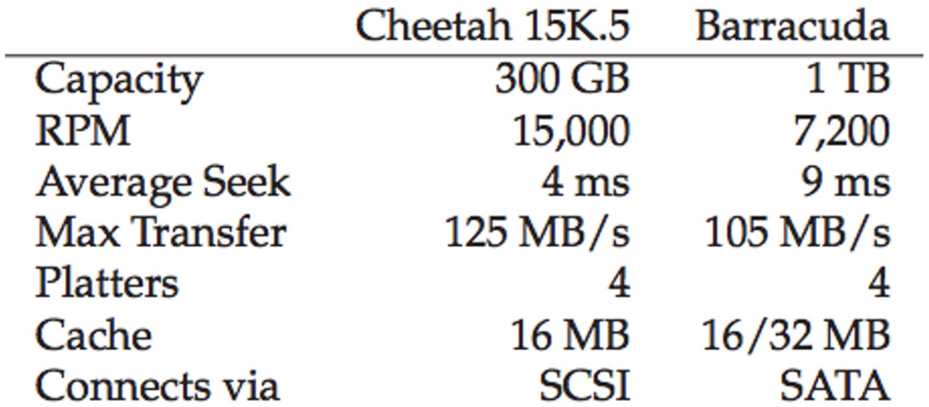
Example: Calculate Rate of I/O Random (Cheetah)

How well does the Cheetah perform under a random workload - random 4KB requests from across disk.

We can also compute the *rate of I/O* which is used as a comparison number between drives.

$$R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$$

4KB/6MS = 1000ms/1sec * 1MB/1024KB
= ~0.66MB/sec



	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

$\boxed{\sim 6ms}$ $\boxed{4ms}$ $\boxed{2ms}$ $\boxed{\sim 30\mu s}$

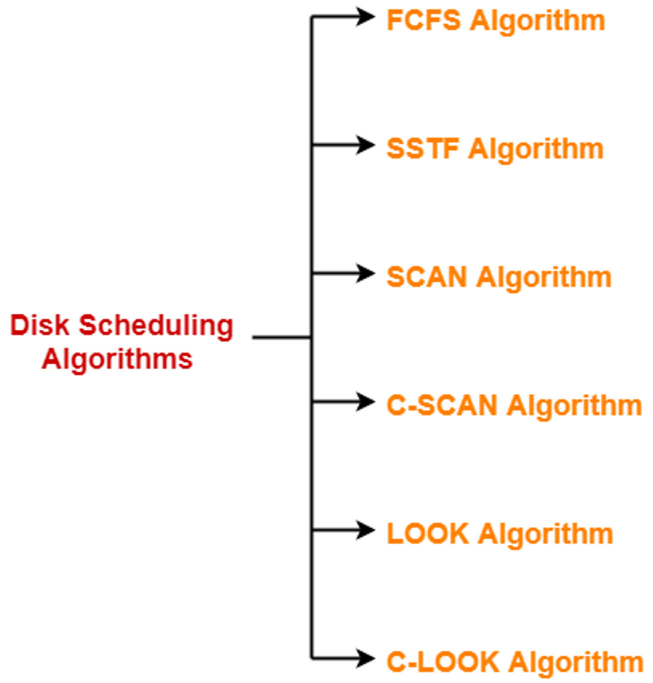
Disk Scheduling

- The overall workload seen by the OS is not as simple as Random or Sequential
- It is actually a mix of requests (reads and writes) from all of the processes
- Given a queue of requests, the OS can reorder requests however it likes
- Such reordering algorithms are generally called disk scheduling

Disk Scheduling

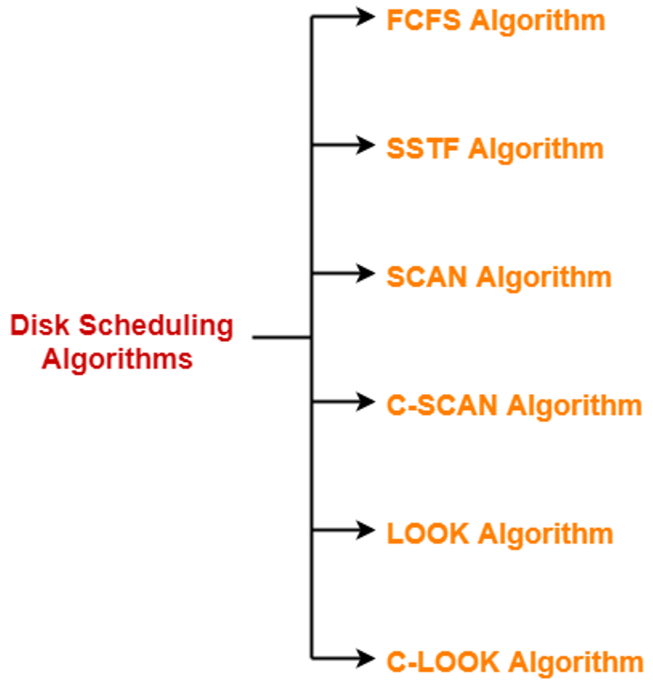
Because of the high cost of I/O, the OS has historically played a role in deciding the order of I/O operations issued to the disk.

More specifically, given a set of I/O requests, the **disk scheduler** examines the requests and decides which one to schedule next.



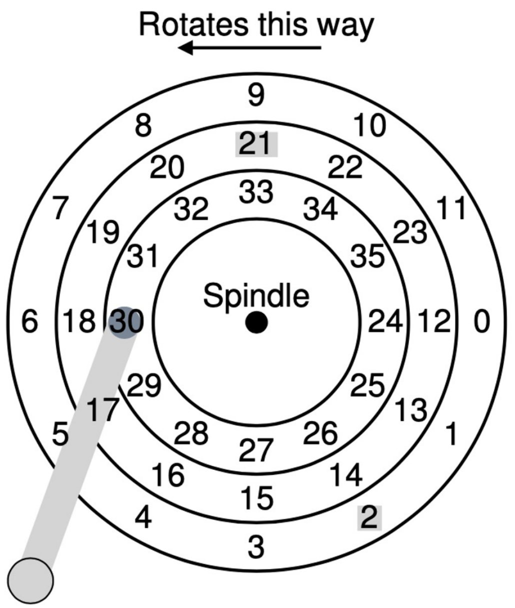
Disk Scheduling

- With CPU scheduling, we do not know how long a particular job/process will run.
- With **disk scheduling**, we can make a good guess at how long a "job" (i.e., disk request) will take.
    - We can do this by estimating the seek and possible rotational delay of a request
    - In doing so, the disk scheduler can know how long each request will take, and thus (greedily) pick the one that will take the least time to service first.
    - Thus, the disk scheduler will try to follow the **principle of SJF (shortest job first)** in its operation.

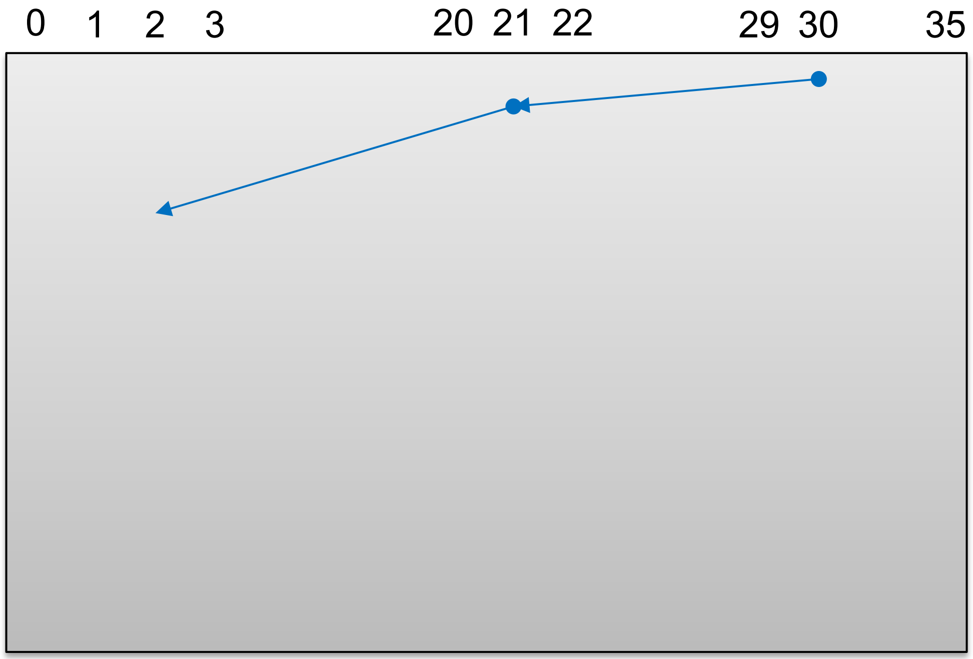


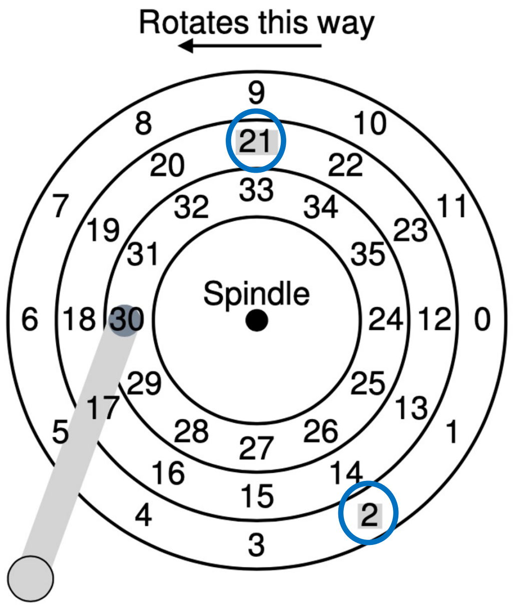
SSTF: Shortest Seek Time First

- One early disk scheduling approach is known as **shortest-seek-time-first** (**SSTF**)
- SSTF orders the queue of I/O requests by track, picking requests on the nearest track to complete first.
- For example, assuming the current position of the head is over the inner track, and we have requests for sectors 21 (middle track) and 2 (outer track), we would then issue the request to 21 first, wait for it to complete, and then issue the request to 2



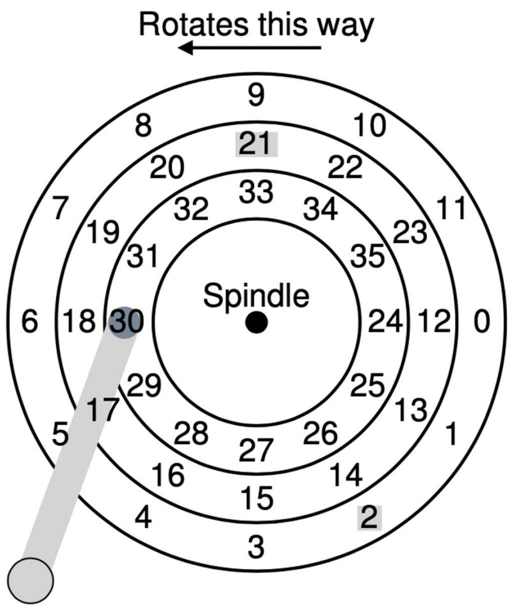
SSTF: Shortest Seek Time First





SSTF: Shortest Seek Time First (Problem #1)

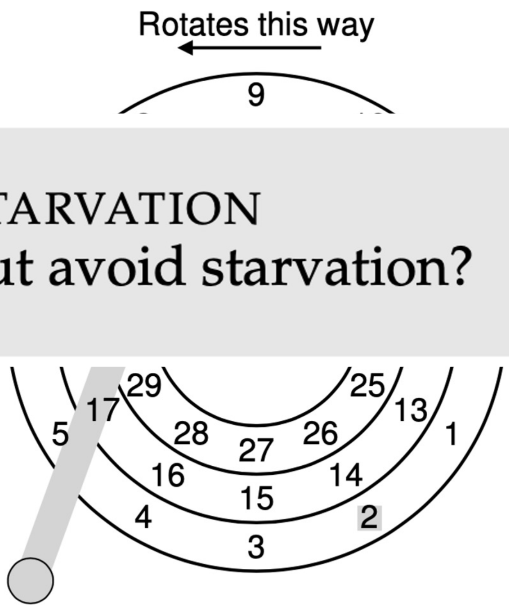
- SSTF works well in this example, seeking to the middle track first and then the outer track.
- However, SSTF is not a panacea, for the following reasons. First, the drive geometry is not available to the host OS; rather, it sees an array of blocks.
- Fortunately, this problem is rather easily fixed. Instead of SSTF, an OS can simply implement **nearest-block-first** (**NBF**), which schedules the request with the nearest block address next.



SSTF: Shortest Seek Time First (Problem #2)

- The second problem is more fundamental: **starvation**. Imagine in this

be ignored completely by a pure SSTF approach. Yuck!

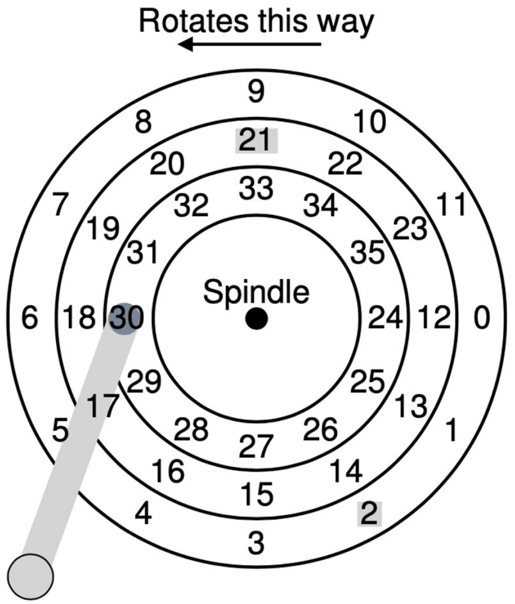


CRUX: HOW TO HANDLE DISK STARVATION

How can we implement SSTF-like scheduling but avoid starvation?

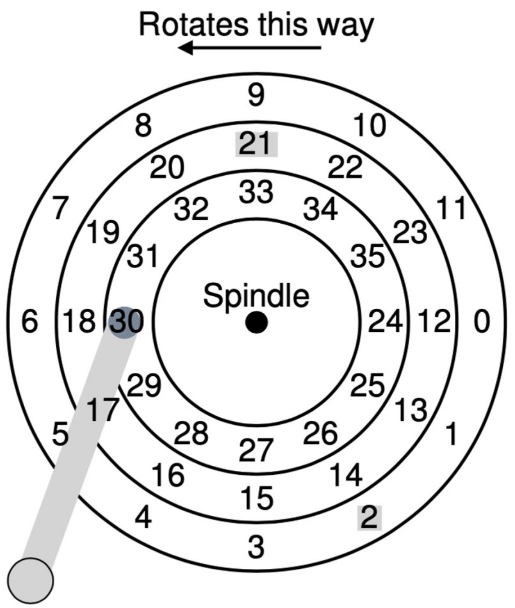
SCAN / C-SCAN: Elevator

- The answer to the starvation problem was developed some time ago. The algorithm, originally called **SCAN**, simply moves back and forth across the disk servicing requests in order across the tracks.
- Let's call a single pass across the disk (from outer to inner tracks, or inner to outer) a *sweep*.
- Thus, if a request comes for a block on a track that has already been serviced on this sweep of the disk, it is not handled immediately, but rather queued until the next sweep (in the other direction).



SCAN / C-SCAN: Elevator

- **C-SCAN** is another common variant, short for **Circular SCAN**.
- Instead of sweeping in both directions across the disk, the algorithm only sweeps from outer-to-inner, and then resets at the outer track to begin again.
- Doing so is fairer to inner and outer tracks, as pure back- and-forth SCAN favors the middle tracks, i.e., after servicing the outer track, SCAN passes through the middle twice before coming back to the outer track again.



SCAN / C-SCAN: Elevator

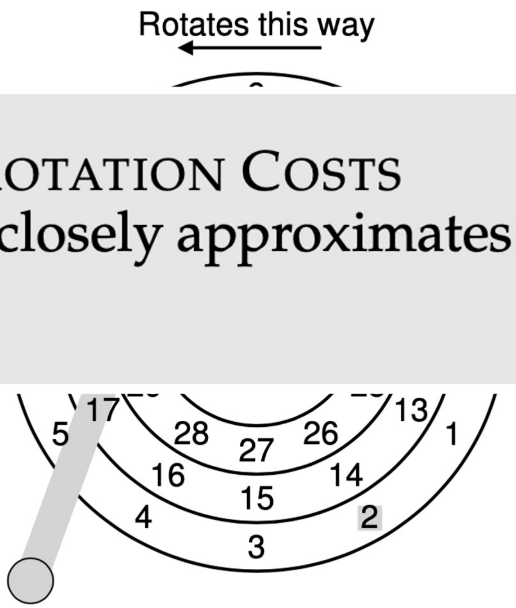
- For reasons that should now be clear, the SCAN algorithm (and its cousins) is sometimes referred to as the **elevator** algorithm, because it behaves like an

CRUX: HOW TO ACCOUNT FOR DISK ROTATION COSTS
How can we implement an algorithm that more closely approximates SJF by taking *both* seek and rotation into account?

“closer” than 1!

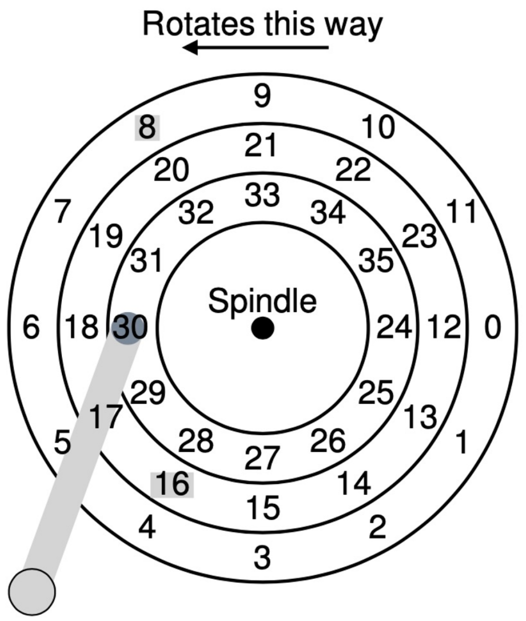
- As you can see, the elevator algorithm, when used in real life, prevents fights from taking place on elevators. In disks, it just prevents starvation.

- Unfortunately, these algorithms ignore rotation!



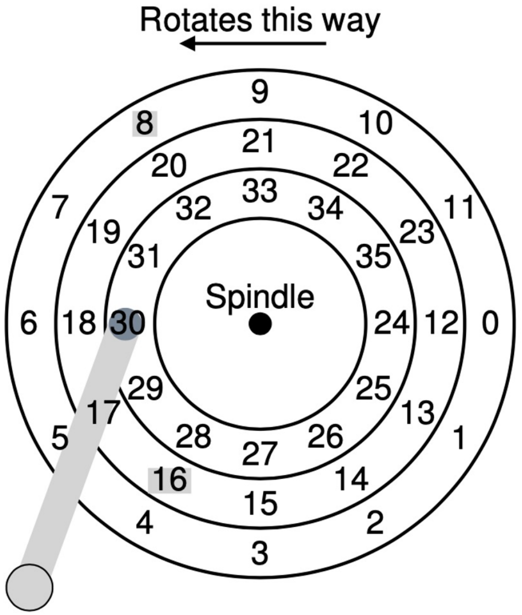
SPTF: Shortest Positioning Time First

- In this example, the head is currently positioned over sector 30 on the inner track. The scheduler must decide if it should schedule sector 16 (on the middle track) or sector 8 (on the outer track) for its next request.
- So, which should it service next?
- **The Answer:** it depends



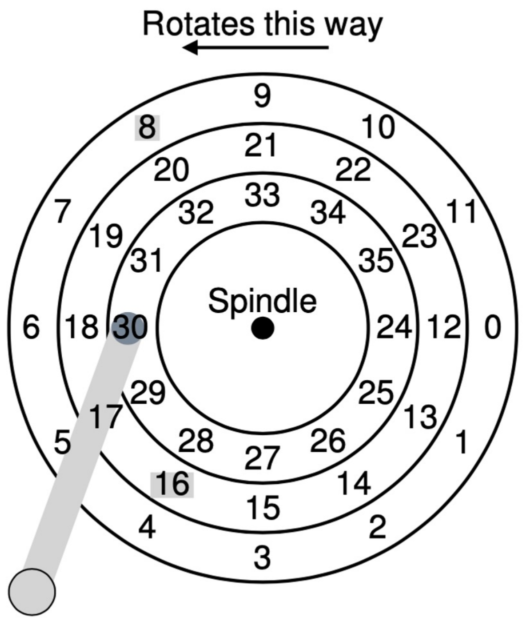
SPTF: Shortest Positioning Time First

- On modern drives, both seek and rotation are roughly equivalent
- Thus, SPTF is useful and improves performance.
- However, it is even more difficult to implement in an OS, which generally does not have a good idea where track boundaries are or where the disk head currently is (in a rotational sense).
- So, SPTF is typically performed in the drive.



SPTF: Shortest Positioning Time First

- At the end of the day, disk scheduling is a coordinated effort between the OS and the drive
- The OS schedules what it thinks the best ordering of requests are and issues them to disk
- The disk uses its internal knowledge of the disk geometry to service those requests in the best possible order



Examples

- Disk is at sector 30. Example queue: 65, 40, 18, 78
- SSTF Seeks: 40, 18, 65, 78
- SCAN (resets to 0 first): 18, 40, 65, 78
- C-SCAN: 40, 65, 78, 18
- All of these can create a lot of unfairness between processes that have lots of requests in one part of the disk