IT308: Operating Systems

Hard Disks

Disks

- Disks are a crucial part of computing systems
- This is where we save persistent data and swap memory pages to

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)
  - Solid Sate Drives (SSD)
- SSDs are plainly better and are taking over ...

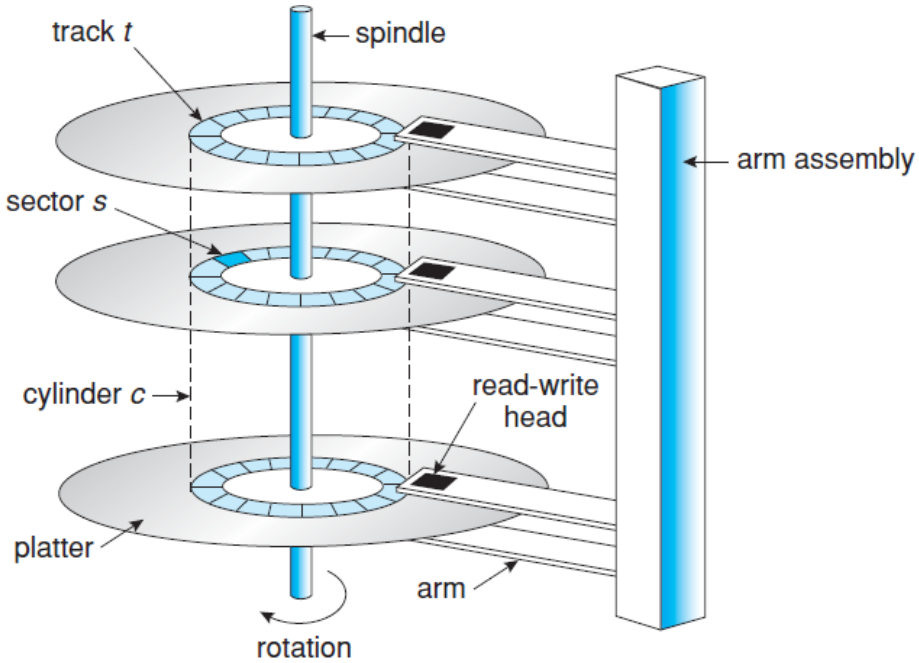
Disks (2)

- The disk gives us an address space abstraction, much like physical memory
- The drive consists of N sectors (also called blocks) of some size, typically 512 Bytes

Disks (2)

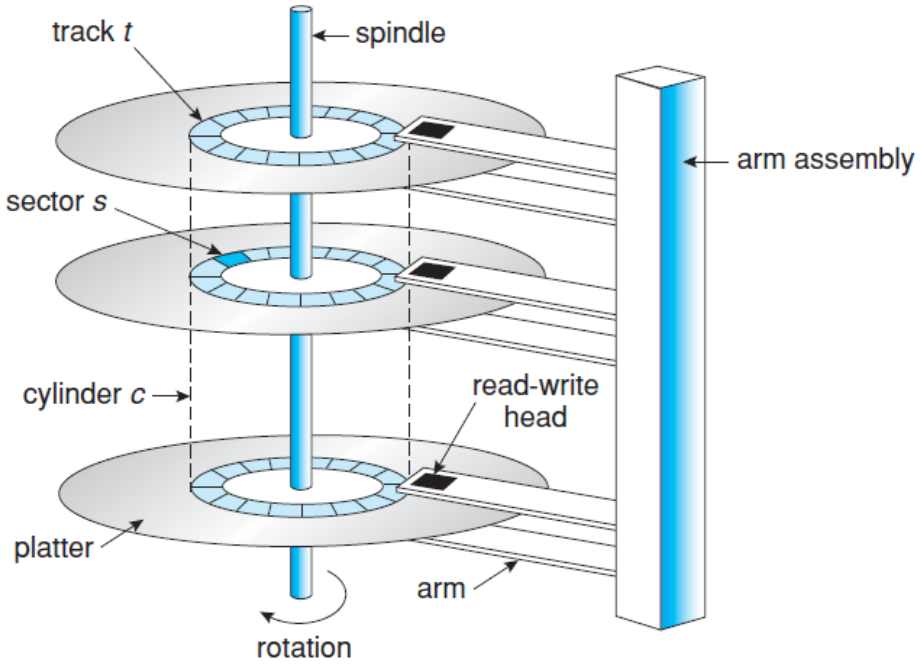
- The disk gives us an address space abstraction, much like physical memory
- The drive consists of N sectors (also called blocks) of some size, typically 512 Bytes
- Unlike RAM, you have to read/write the whole block
- A single block write is atomic (either complete all or nothing)

Disk Organization



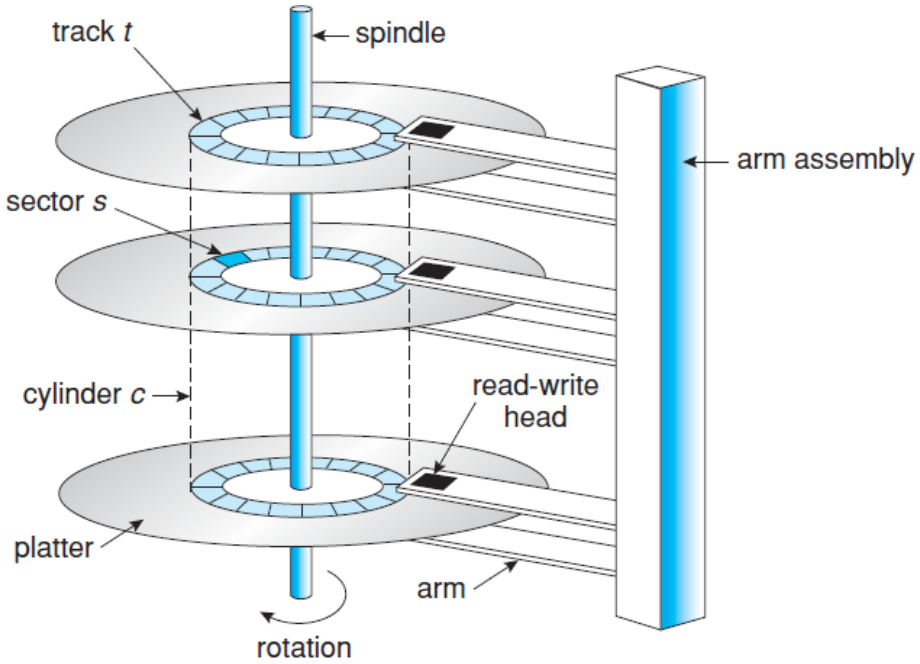
- Platter: a circular hard surface
- Surface: one side of the platter

Disk Organization



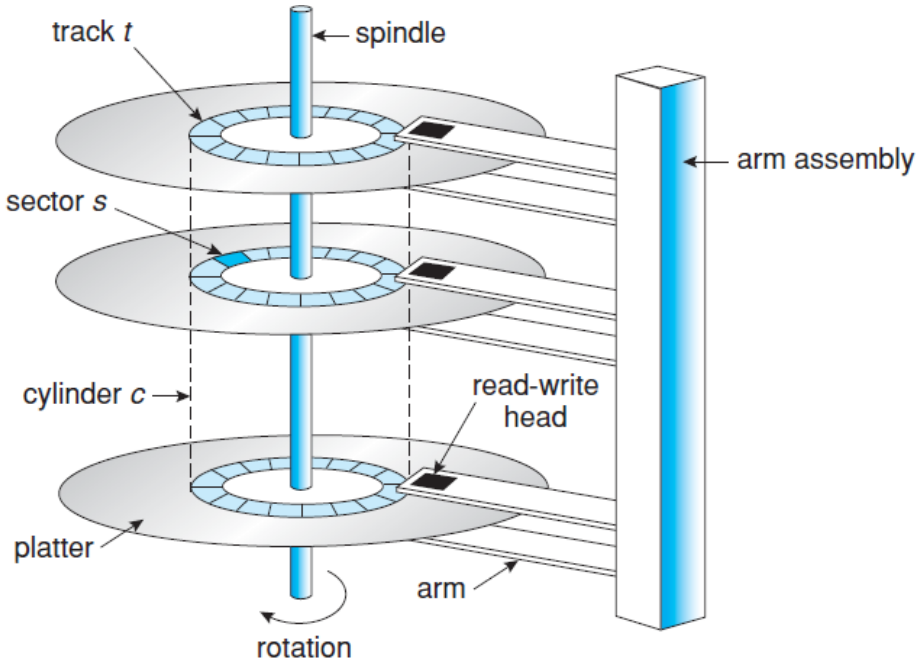
- Platter: a circular hard surface
- Surface: one side of the platter
- Spindle: the spinning motor shaft, typical 5k-10k RPM

Disk Organization



- Platter: a circular hard surface
- Surface: one side of the platter
- Spindle: the spinning motor shaft, typical 5k-10k RPM
- Track: Concentric circles of sectors

Disk Organization



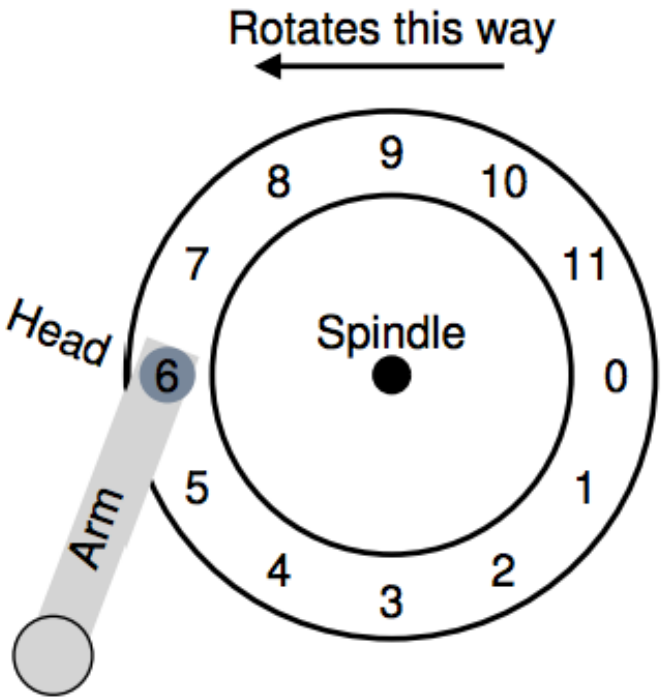
- Platter: a circular hard surface
- Surface: one side of the platter
- Spindle: the spinning motor shaft, typical 5k-10k RPM
- Track: Concentric circles of sectors
- Head: Read and write (one head per surface)
  - attached to an arm, which moves across the surface

Disk Organization (2)

- When the disk drive is operating, the disk is rotating at constant speed
- To read or write, the disk head must be positioned on the desired track and at the beginning of the desired sector
- Once the head is in position, the read or write operation is then performed as the sector moves under the head

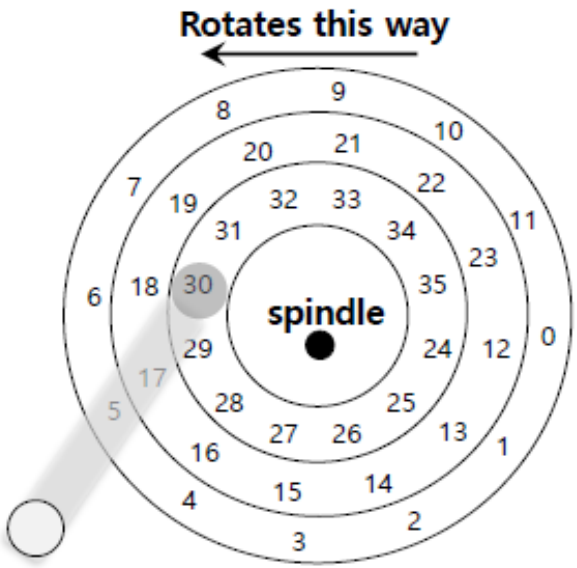
Rotational Delay

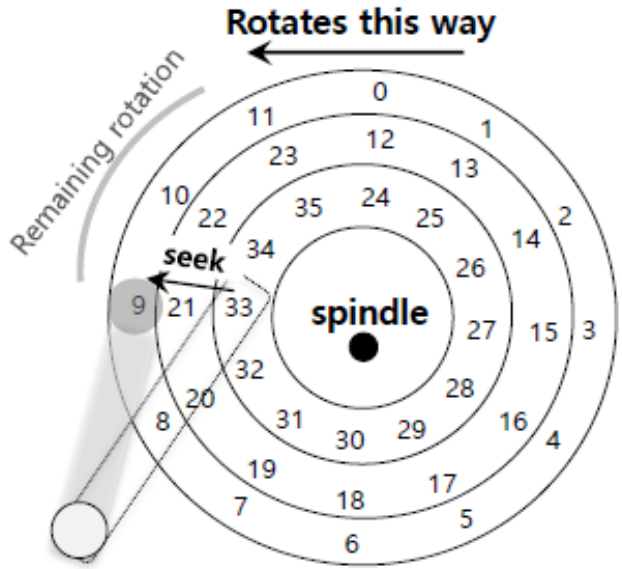
The time it takes tor the desired sector to rotate under the head.



Seek Time

The time it takes to position the head on the desired track.





Question

If it is a 10k RPM disk, how long is the rotational time in the worst case?

A. 1 us

B. 0.1 ms

C. 1 ms

D. 6 ms

Transfer Time

- Once the disk seeks, it has to read and transfer the data
- So the total time is $T_{seek} + T_{rotation} + T_{transfer}$

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

One is a "performance" disk vs a capacity disk. The one on the left is more $$$.

Drive performance

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

Question

Under the random workload how long does the Cheetah disk take to find and transfer 4kB?

A. 6 ms + 30 μs

B. 4 ms + 30 μs

C. 2 ms + 100 μs

D. 6 ms + 100 μs

- $T_{seek} + T_{rotation} + T_{transfer}$

	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

Results

	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

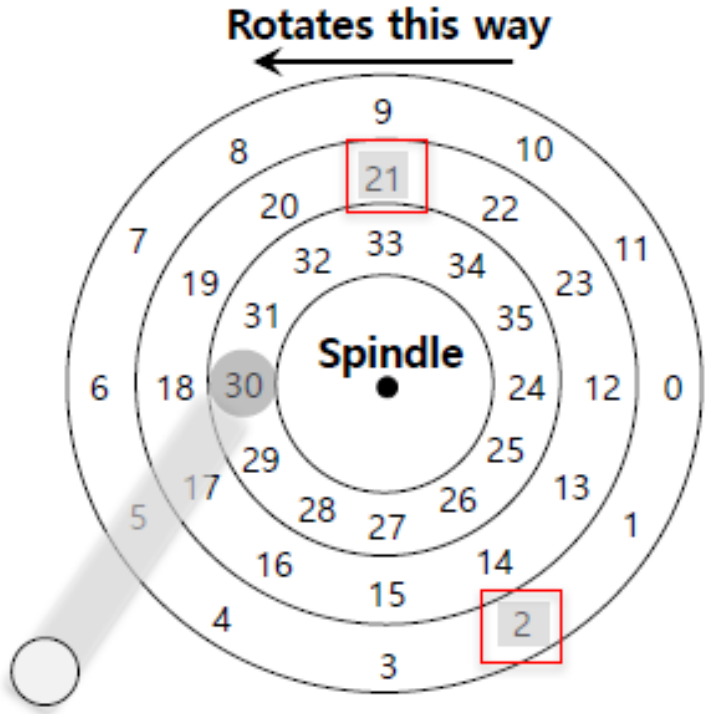
	Cheetah	Barracuda
$R_{I/O}$ Random	0.66 MB/s	0.31 MB/s
$R_{I/O}$ Sequential	125 MB/s	105 MB/s

Disk Scheduling

- The overall workload seen by the OS is 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

Shortest Seek Time First (SSTF)

- Select the request that requires the least movement of the disk arm from its current position
- Example: scheduling requests 21 and 2
  - request to 21 $\rightarrow$ request to 2



Shortest Seek Time First (SSTF)

- Always chooses the minimum seek time
- Requests for tracks far away from the current position may never be served, if requests for closer tracks are issued continuously (starvation)

SCAN (aka elevator algorithm)

- Move across the disk servicing requests in order across the tracks
- Sweep: A single pass across the disk
- If a request comes for a block on a track that has already been serviced on this sweep of the disk, it is queued until the next sweep

SCAN (aka elevator algorithm)

- Move across the disk servicing requests in order across the tracks
- Sweep: A single pass across the disk
- If a request comes for a block on a track that has already been serviced on this sweep of the disk, it is queued until the next sweep
- C-SCAN (C is circular)
  - Sweep only from outer-to-inner, or inner-to-outer, etc. (better fairness since middle tracks are no longer favored)

Examples

- Disk is at track 30. Example queue: 65, 40, 18, 78
- SSTF Seeks:

Examples

- Disk is at track 30. Example queue: 65, 40, 18, 78
- SSTF Seeks:
  - 40, 18, 65, 78

Examples

- Disk is at track 30. Example queue: 65, 40, 18, 78
- SSTF Seeks:
  - 40, 18, 65, 78
- SCAN seeks:

Examples

- Disk is at track 30. Example queue: 65, 40, 18, 78
- SSTF Seeks:
  - 40, 18, 65, 78
- SCAN seeks:
  - 40, 65, 78, 18

Examples

- Disk is at track 30. Example queue: 65, 40, 18, 78
- SSTF Seeks:
  - 40, 18, 65, 78
- SCAN seeks:
  - 40, 65, 78, 18
- C-SCAN (resets to 0 first) seeks:
  - 18, 40, 65, 78

Examples

- Disk is at track 30. Example queue: 65, 40, 18, 78
- SSTF Seeks:
  - 40, 18, 65, 78
- SCAN seeks:
  - 40, 65, 78, 18
- C-SCAN (resets to 0 first) seeks:
  - 18, 40, 65, 78
- All of them ignore rotation

In Practice

- In reality, disk scheduling is not nearly so simple
- Modern OSs also:
  - Issue multiple request simultaneously (say 16) because the HDD control does its own scheduling based on internal knowledge
  - Wait before issuing requests because a better one might come along (anticipatory scheduling)
  - Have to worry about fairness and starvation between processes