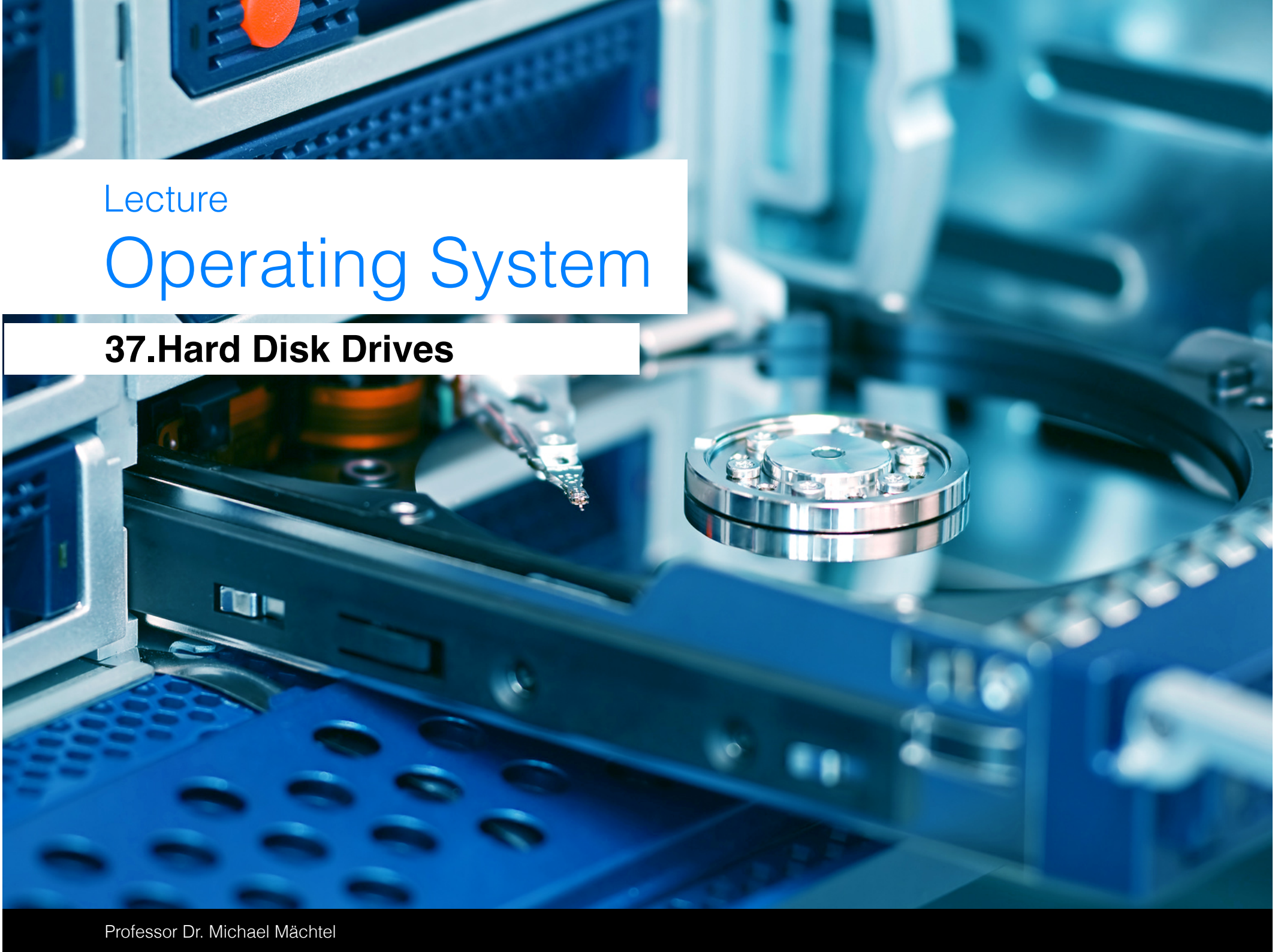


Lecture

Operating System

37.Hard Disk Drives

Professor Dr. Michael Mächtel

37. Hard Disk Drives

1. **Hard Discs**
2. **Time To Read/Write**
3. **I/O Schedulers**
4. **Other Scheduling Issues**

37. Hard Disk Drives

1. **Hard Discs**
2. Time To Read/Write
3. I/O Schedulers
4. Other Scheduling Issues

Hard Disk Driver

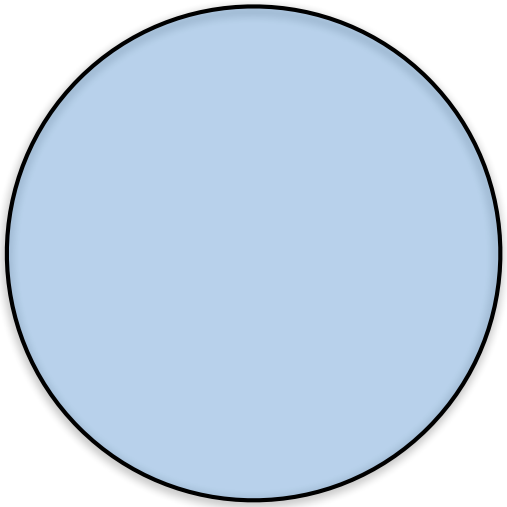
- Hard disk driver have been the **main form of persistent data storage** in computer systems for decades.
  - The drive consists of a large number of **sectors** (512-byte blocks).
  - **Address Space** :
    - We can view the disk with n sectors as an array of sectors; 0 to n-1.

Interface

- The only guarantee is that a single 512-byte write is **atomic**.
- Multi-sector operations are possible.
  - Many file systems will read or write 4KB at a time.
  - **Torn write**:
    - If an untimely power loss occurs, only a portion of a larger write may complete.
- Accessing blocks in a **contiguous chunk** is the fastest access mode.
  - A *sequential* read or write
  - Much faster than any more *random* access pattern.

Basic Geometry

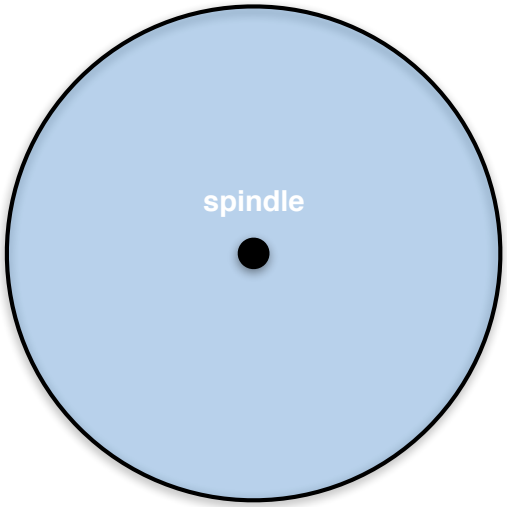
- **Platter** (Aluminum coated with a thin magnetic layer)
  - A circular hard surface
  - Data is stored persistently by inducing magnetic changes to it.
  - Each platter has 2 sides, each of which is called a surface.

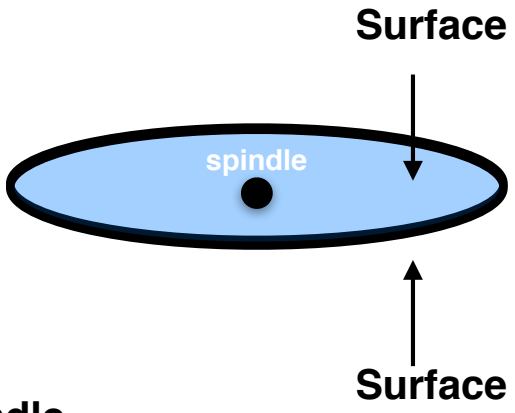


Platter

Basic Geometry (Cont.)

- **Spindle**
  - Spindle is connected to a motor that spins the platters around.
  - The rate of rotations is measured in **RPM** (Rotations Per Minute).
    - Typical modern values : 7,200 RPM to 15,000 RPM.
    - E.g., 10000 RPM : A single rotation takes about 6 ms.

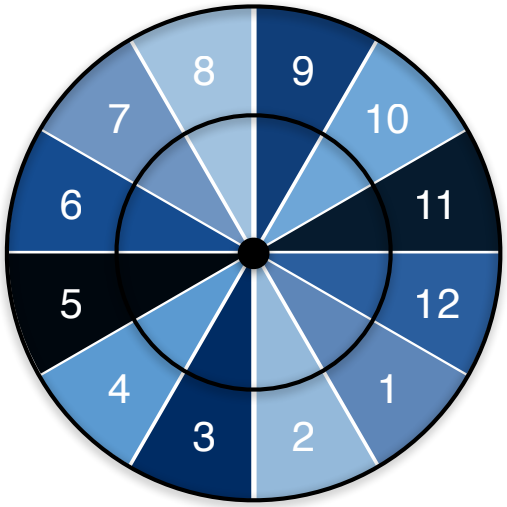




Platter with spindle

Basic Geometry (Cont.)

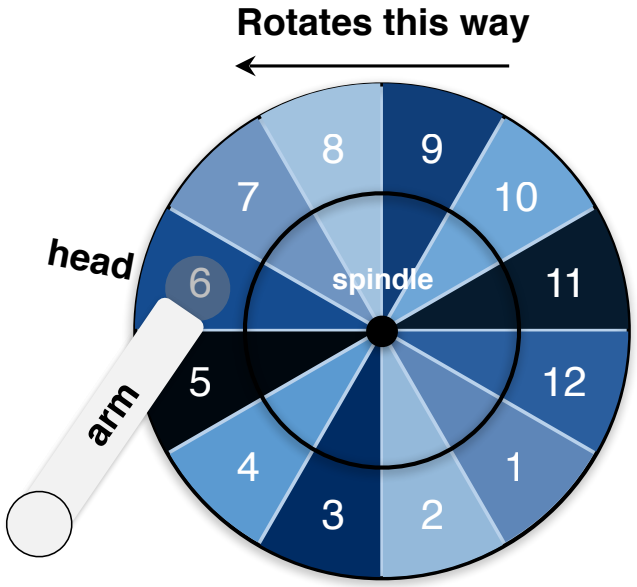
- **Track**
  - Concentric circles of sectors
  - Data is encoded on each surface in a track.
  - A single surface contains many thousands and thousands of tracks.



A Disk with Just A Single Track (12 sectors)

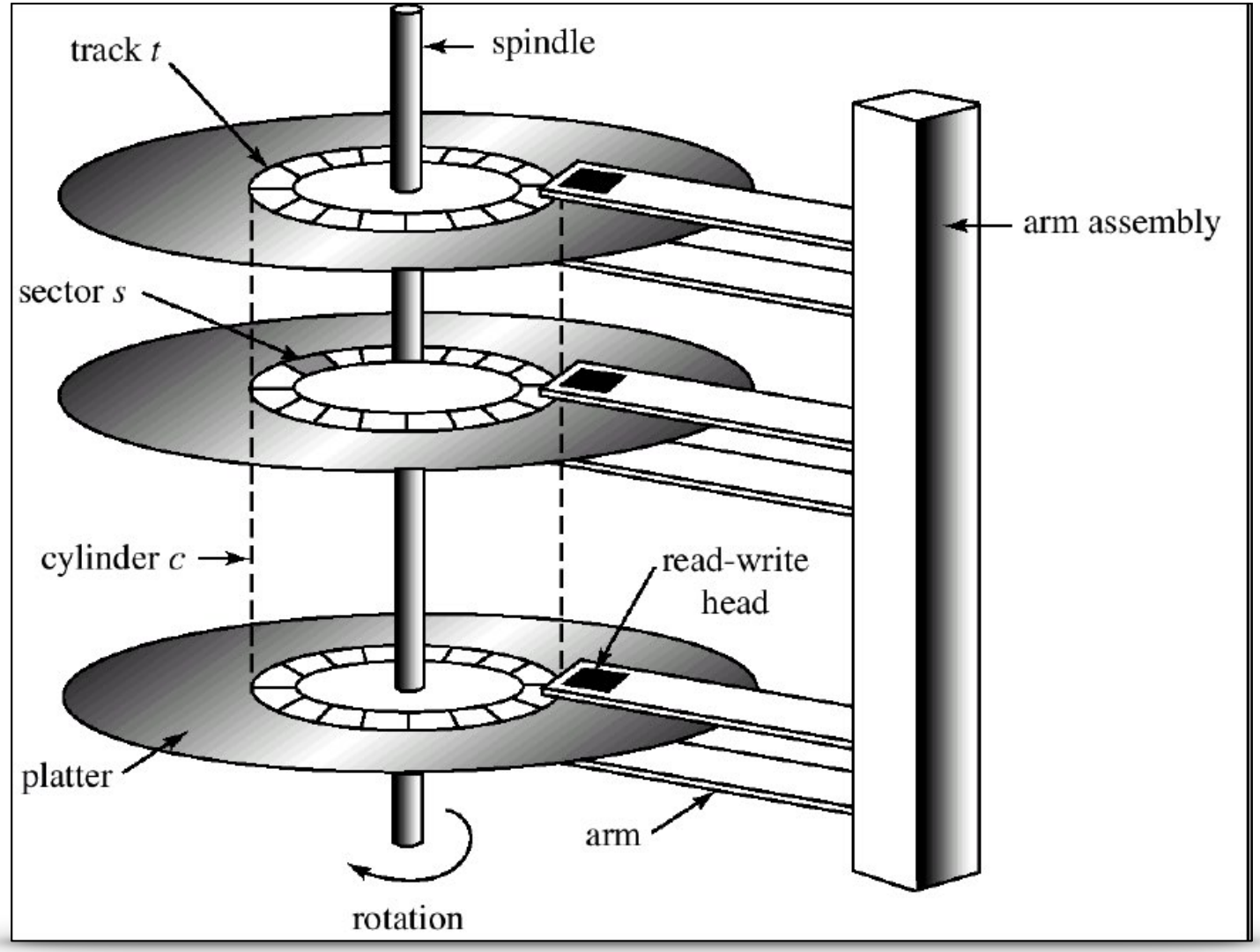
Basic Geometry (Cont.)

- **Disk head** (One head per surface of the drive)
  - The process of *reading* and *writing* is accomplished by the **disk head**.
  - Attached to a single disk **arm**, which moves across the surface.



A Single Track Plus A Head

Example of a Disk



37. Hard Disk Drives

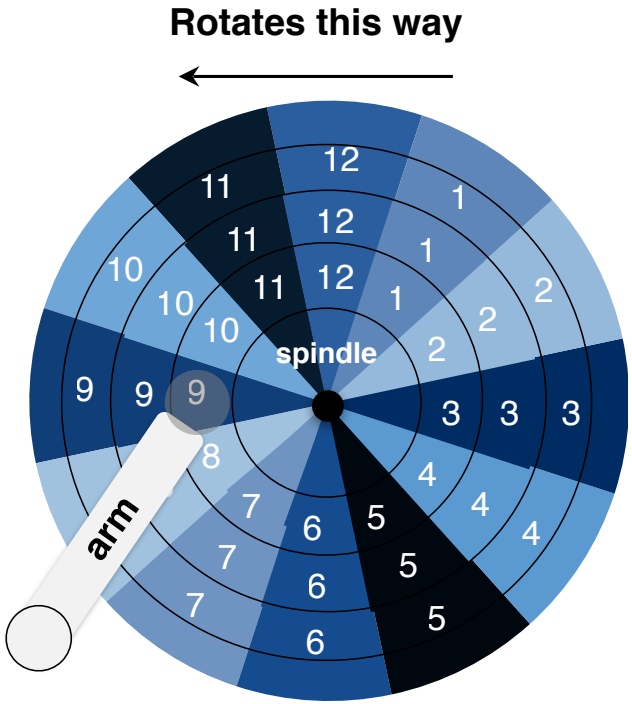
1. Hard Discs
2. **Time To Read/Write**
3. I/O Schedulers
4. Other Scheduling Issues

Time to Read/write

- **Complete I/O time**:
  - **Seek**
  - Waiting for the **rotational delay**
  - **Transfer**

Seek

- **Seek**: Move the disk arm to the correct track
  - **Seek time**: Time to move head to the track contain the desired sector.
  - One of the most costly disk operations.

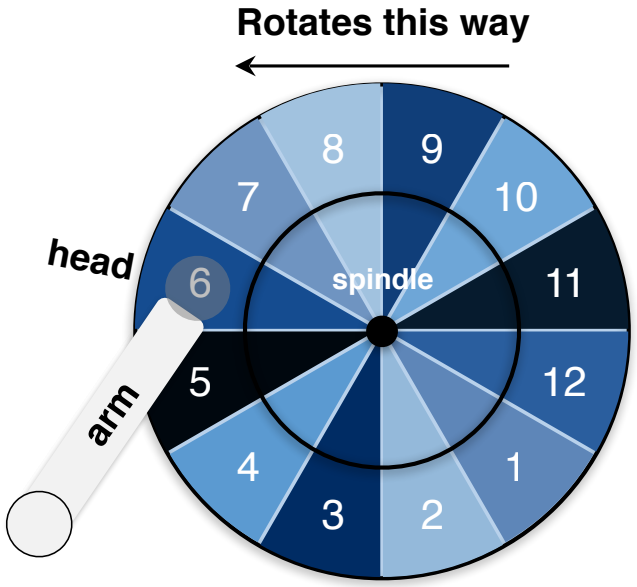


Seek (Cont.)

- Seek cost: Function of cylinder distance
  - Not purely linear cost
- Acceleration → Coasting → Deceleration → Settling
  - **Acceleration**: The disk arm gets moving.
  - **Coasting**: The arm is moving at full speed.
  - **Deceleration**: The arm slows down.
  - **Settling**: The head is *carefully positioned* over the correct track.
    - The settling time is often quite significant, e.g., 0.5 to 2ms.
- Entire seeks often takes several milliseconds
  - 4 - 10 ms
- Approximate average seek distance = 1/3 max seek distance

Rotation delay

- **Rotational delay**: Time for the desired sector to rotate
  - Ex) Full rotational delay is R and we start at `sector 6`
    - Read sector 12: Rotational delay $= \frac{R}{2}$
    - Read sector 5: Rotational delay $= R - 1$



A Single Track Plus A Head

Rotation Delay (Cont.)

- Depends on rotations per minute (RPM)
  - 7200 RPM is common, 15000 RPM is high end.
- With 7200 RPM, how long to rotate around?

  1 / 7200 RPM =

  1 minute / 7200 rotations =

  1 second / 120 rotations =

  8.3 ms / rotation
- Average rotation?
  - 8.3 ms / 2 = 4.15 ms

Transfer

- The final phase of I/O
  - Data is either *read from* or *written to* the surface.
- Pretty fast — depends on RPM and sector density.
- 100+ MB/s is typical for maximum transfer rate
- How long to transfer 512-bytes?
  - 512 bytes * (1s / 100 MB) = 5 us

I/O Time: Doing The Math

- I/O Time: $T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$
- The rate of I/O: $R_{I/O} = \frac{Size_{transfer}}{T_{I/O}}$

	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

Disk Drive Specs: SCSI Versus SATA

Workload Performance

- So…
  - seeks are slow
  - rotations are slow
  - transfers are fast
- What kind of workload is fastest for disks?
  - **Sequential**: access sectors in order (transfer dominated)
  - **Random**: access sectors arbitrarily (seek+rotation dominated)

I/O Time Example

	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

- Random workload: Issue 16KB read to random locations on the disk
- Sequential workload: Read 100MB consecutively from the disk

Summary: I/O Time Example

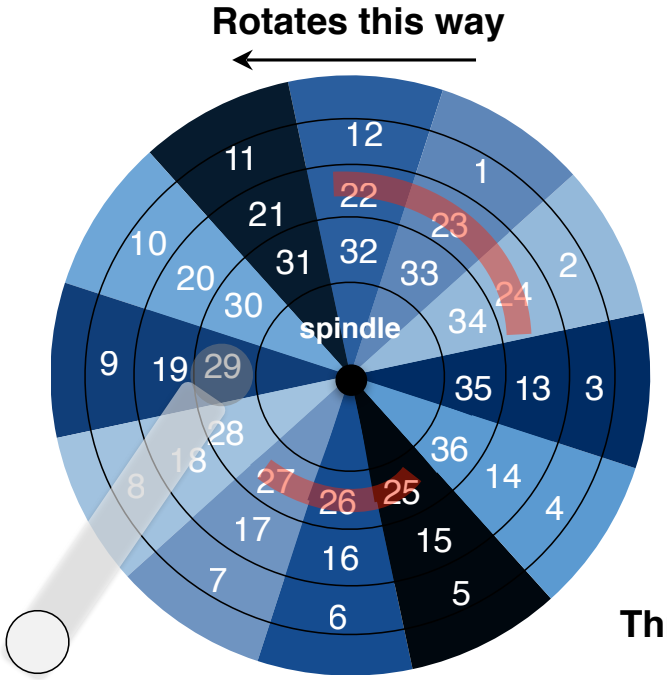
- Random workload: Issue 16KB read to random locations on the disk
- Sequential workload: Read 100MB consecutively from the disk

		Cheetah 15K.5	Barracuda
	∅ T_{seek}	4 ms	9 ms
	∅ $T_{rotation}$	2 ms	4,1 ms
Random	$T_{transfer}$	125 microsecs	149 microsecs
	$T_{I/O}$	6,1 ms	13.2 ms
	$R_{I/O}$	2,5 MB/s	1,2 MB/s
Sequential	$T_{transfer}$	800 ms	≈952 ms
	$T_{I/O}$	806 ms	965,2 ms
	$R_{I/O}$	≈124 MB/s	≈104 MB/s

There is a huge gap in drive performance between random and sequential workloads

Track Skew

- Make sure that sequential reads can be properly serviced even when crossing track boundaries.
  - Without track skew, the head would be moved to the next track but the desired next block would have already rotated under the head.



Three Tracks: Track Skew Of 2

Cache (Track Buffer)

- **Hold data** read from or written to the disk
  - Allow the drive to *quickly respond* to requests.
  - Small amount of memory (usually around 8 or 16 MB)
    - Disks contain internal memory used as cache
- **Write on cache**
  - Writeback (Immediate reporting)
    - Acknowledge a write has completed when it has **put the data in its memory.**
    - faster but dangerous
  - Write through
    - Acknowledge a write has completed after the write has **actually been written to disk**.

37. Hard Disk Drives

1. Hard Discs
2. Time To Read/Write
3. **I/O Schedulers**
4. Other Scheduling Issues

I/O Schedulers

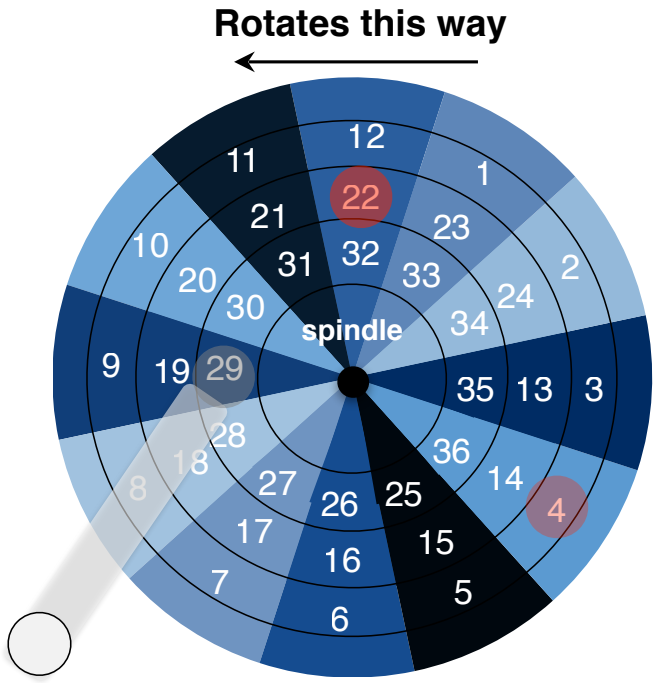
- **Disk Scheduler** decides which I/O request to schedule next.
- Given a stream of I/O requests, in what order should they be served?
- Much different than CPU scheduling
- Position of disk head relative to request position matters more than length of job

FCFS

- Assume seek+rotate = 10 ms for random request
- How long (roughly) does the below workload take?
  - Requests are given in sector numbers
- 300001, 700001, 300002, 700002, 300003, 700003
  - ~60ms
- Reorder: 300001, 300002, 300003, 700001, 700002, 700003
  - ~20ms

SSTF and Elevator Scheduler

- SSTF (Shortest Seek Time First)
  - Order the queue of I/O request by track
  - Pick requests on the nearest track to complete first



SSTF: Scheduling Request 4 and 22
Issue the request to 22 → issue the request to 4

SSTF is not a panacea.

- **Problem 1**: The drive geometry is not available to the host OS
  - Solution: OS can simply implement Nearest-block-first (NBF)
- **Problem 2**: Starvation
  - If there were a steady stream of request to the inner track, request to other tracks would then be ignored completely.

Elevator Schedulers

- **SCAN (Sweep)**: A single pass across the disk
  - If a request comes for a block on a track that has already been services on this sweep of the disk, it is queued until the next sweep.
- **F-SCAN**
  - Freeze the queue to be serviced when it is doing a sweep
  - Avoid starvation of far-away requests
- **C-SCAN**
  - Sweep from outer-to-inner, and then inner-to-outer, etc.

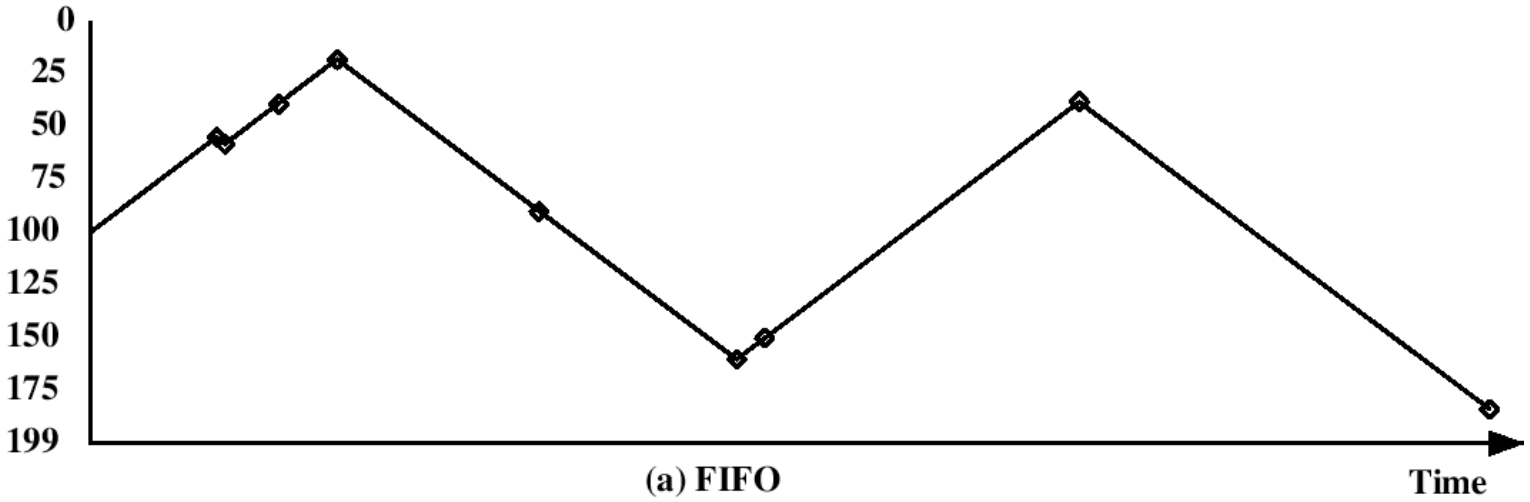
Example

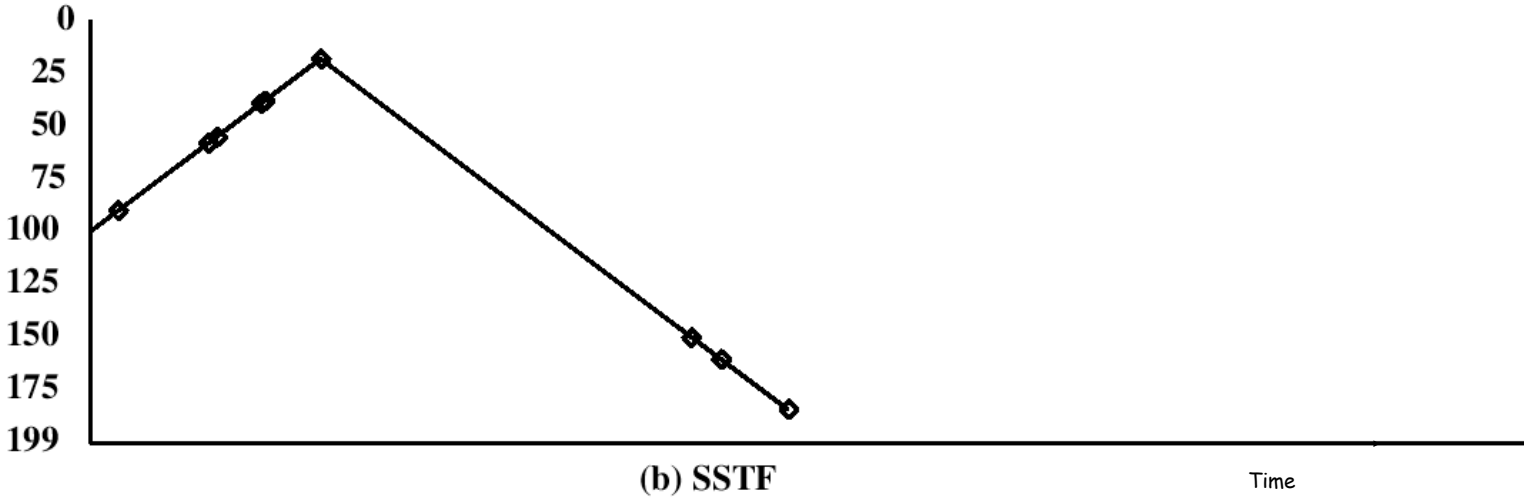
- Disk with 200 tracks, head on track 100
  - I/O Queue: 55,58,39,18,90,160,150,38,184

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Head Movement

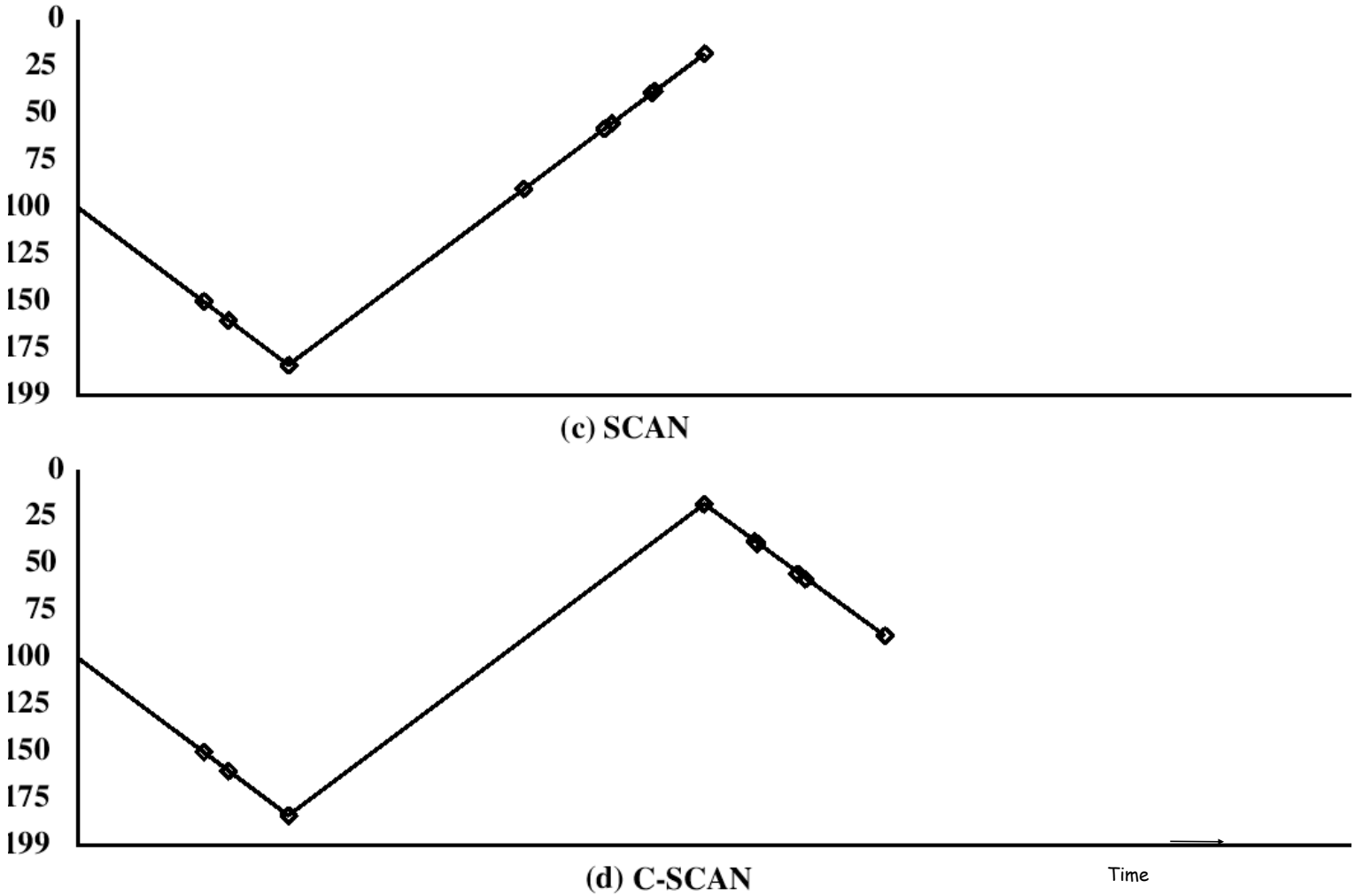
FIFO, SSTF





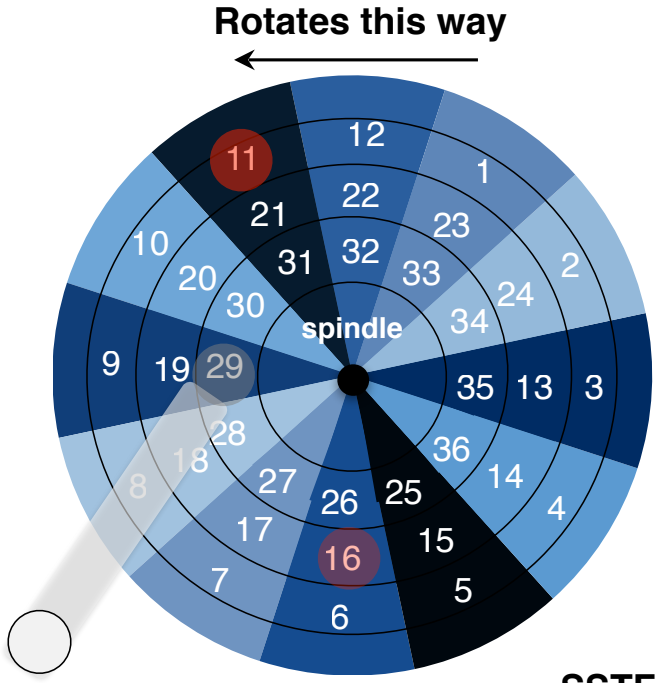
Head Movement

SCAN, C-SCAN



How to account for Disk rotation costs?

- If rotation is faster than seek : request 16 → request 11
- If seek is faster than rotation : request 11 → request 16



On modern drives, both seek and rotation are roughly equivalent:

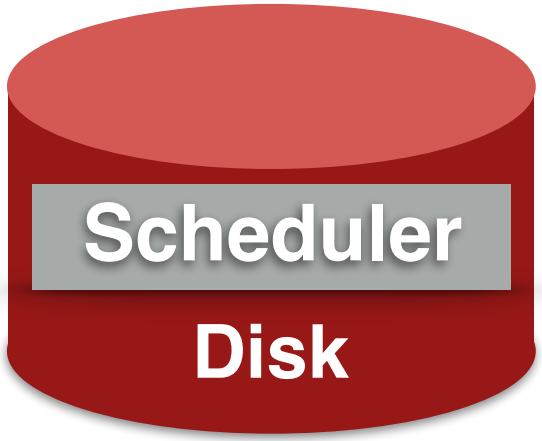
Thus, SPTF (Shortest Positioning Time First) is useful.

SSTF: Sometimes not good enough

37. Hard Disk Drives

1. Hard Discs
2. Time To Read/Write
3. I/O Schedulers
4. **Other Scheduling Issues**

Where should the scheduler go?



I/O merging

- Reduce the number of request sent to the disk and lowers overhead
  - E.g., read blocks 33, then 8, then 34:
    - The scheduler merge the request for blocks 33 and 34 into a single two-block request.

Work Conservation

- **Work conserving schedulers** always try to do work if there's work to be done
- Sometimes, it's better to wait instead if system **anticipates** another request will arrive
- Such **non-work-conserving schedulers** are called **anticipatory** schedulers

CFQ (Linux Default)

- **Completely Fair Queueing**
  - Queue for each process
    - Optimize order within queue
  - Weighted round-robin between queues, with slice time proportional to priority
  - Yield slice only if idle for a given time (anticipation)
  - I/O Requests have deadline, therefore no starvation appears.

Hard Disk Summary

- Storage devices provide common block interface
- On a disk: Never do random I/O unless you must!
  - e.g., Quicksort is a terrible algorithm on disk
- Spend time to schedule on slow, stateful devices

Thanks

Questions?