CS3.301 Operating Systems and Networks

Persistence: Hard Disks

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Software Engineering Research Centre

Acknowledgement

The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

Sources:

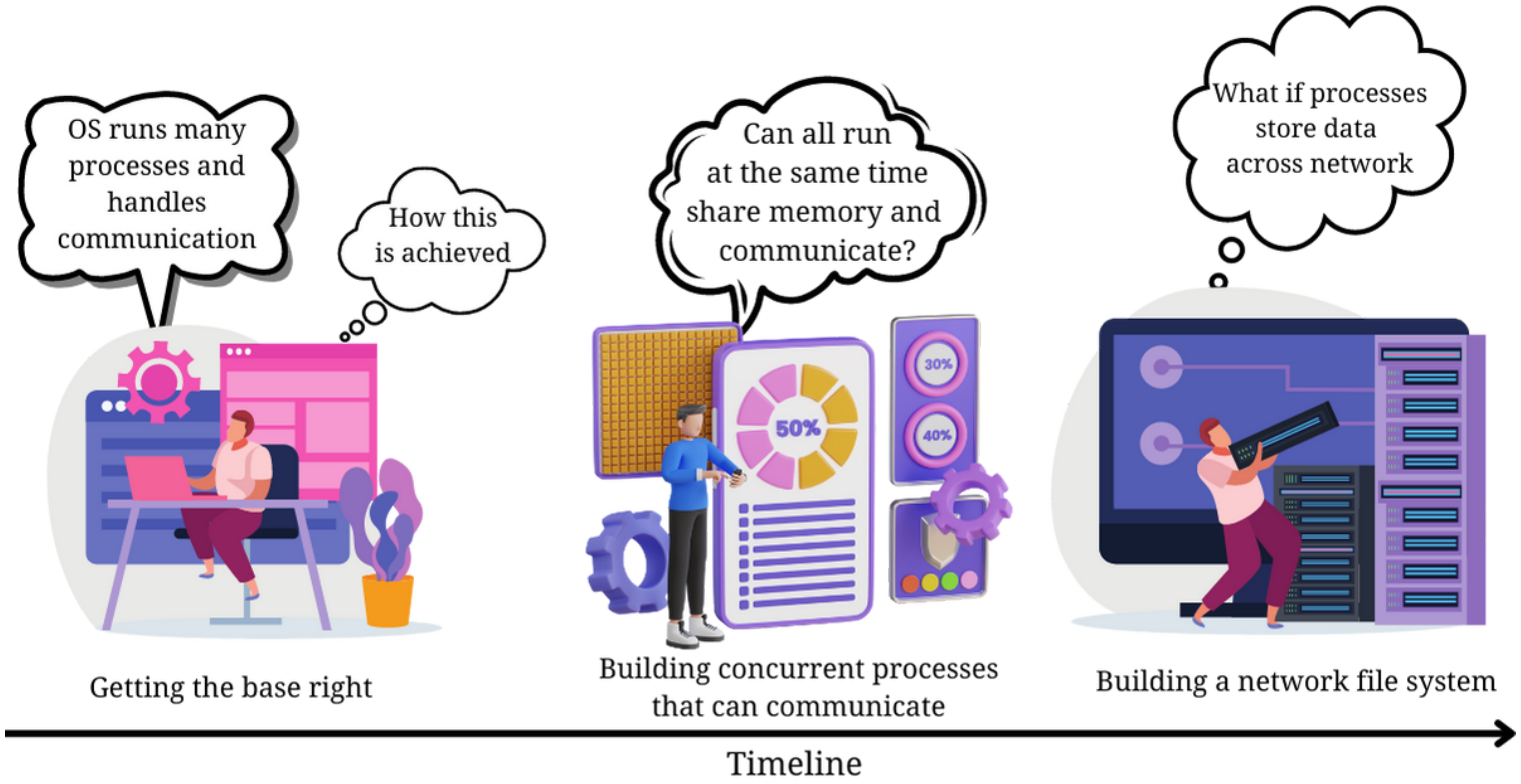
- Computer Networks, 6e by Tanebaum, Teamster and Wetherall
- Computer Networks: A Top Down Approach by Kurose and Ross
- Computer Networking essentials, Youtube Channel
- Other online sources which are duly cited

Putting it together for Networks

Wait we still have one main part

How does OS handles the data and how is it stored?

Lets go to the final part



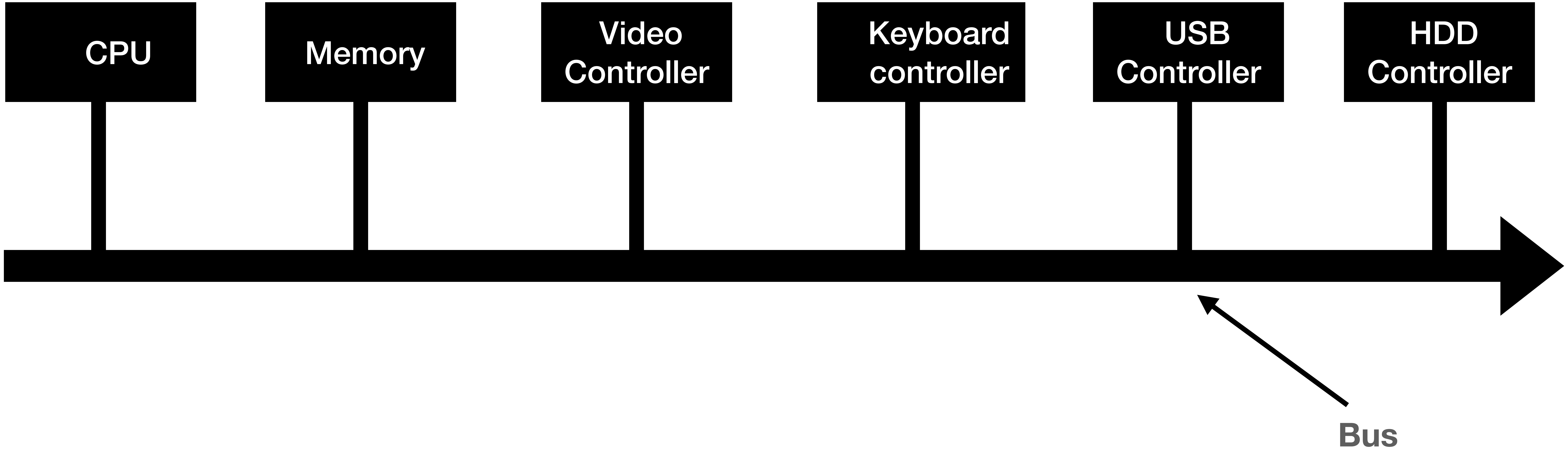
This Course

Process and Memory Virtualization	Concurrency	Persistence
Networking intro	Addressing and Routing	Network file Systems

Persistence

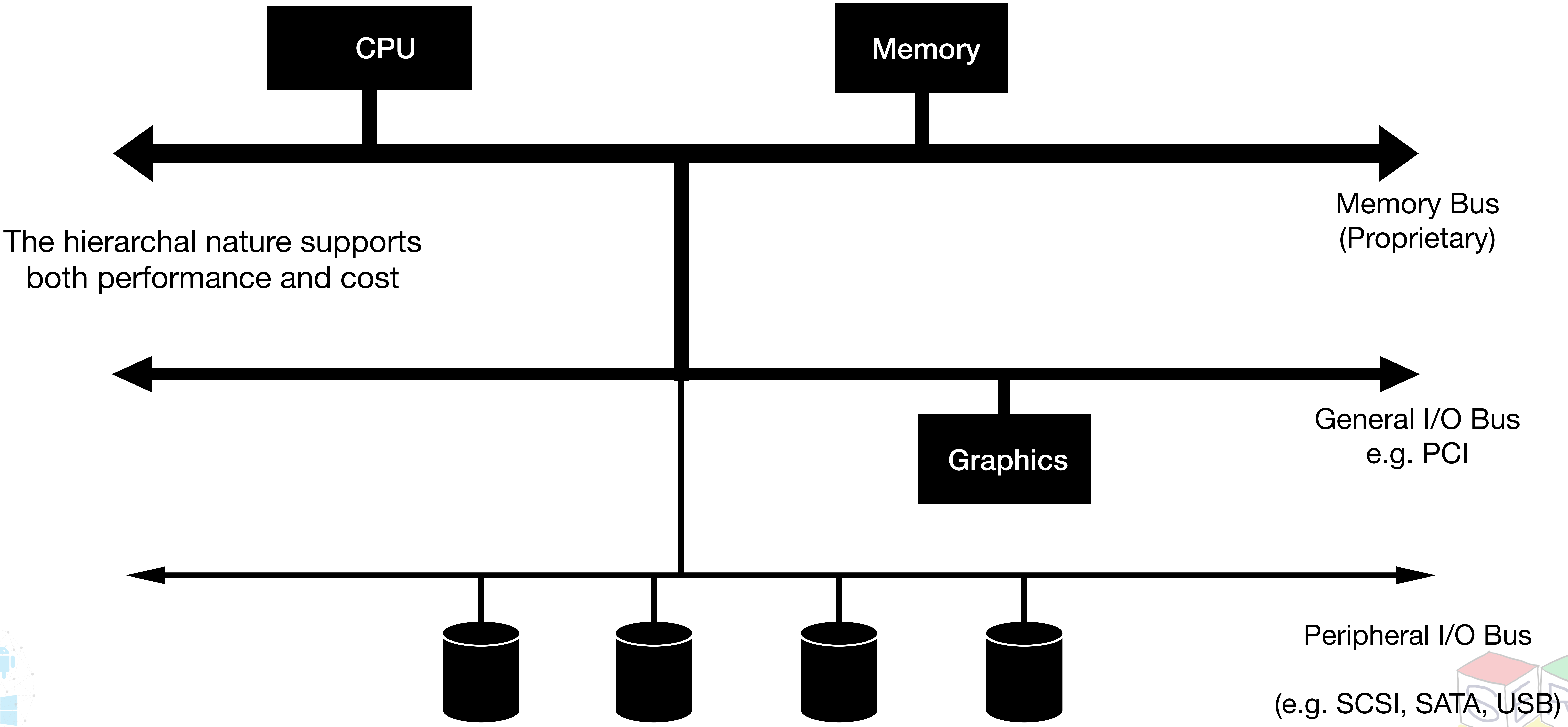
- RAM is Volatile - Again we need two parts here
- **Hardware** and **software** are needed to store data persistently
  - Hardware: I/O devices such as hard drive, SSDs, etc.
  - Software:
    - File system manages the disk
    - File system is responsible for any files that the user creates
    - Read, writes are handled by file system which interacts with low level device drivers

Some Prerequisite



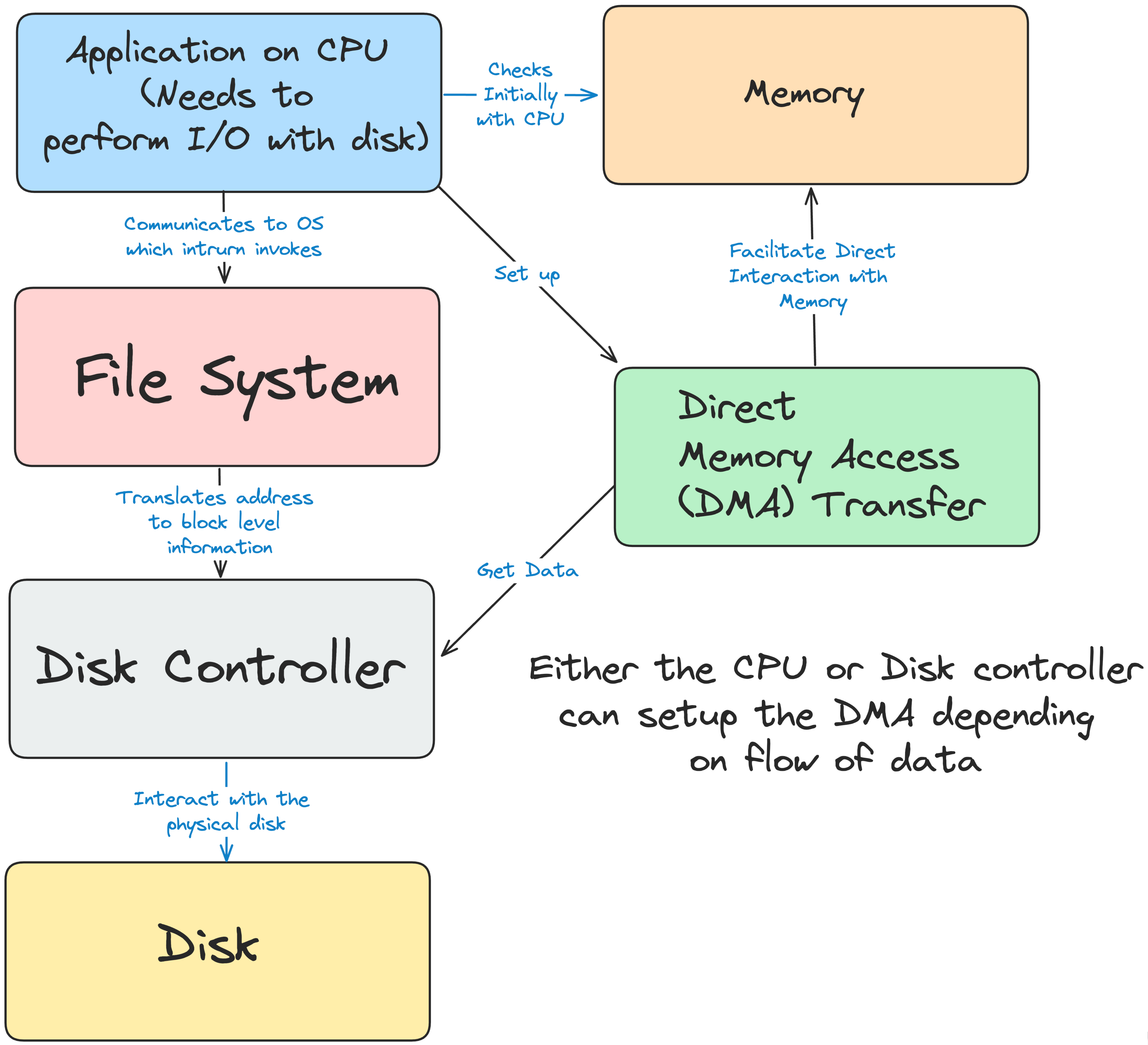
As we go more away from CPU, the more time it takes

Prototypical System Architecture



The flow of access

- Application performs read or write to a file
- CPU communicates to OS which invokes the File System (FS)
- The OS may check in its cache if its already there
- FS prepares block level information to disk controller
- A Direct Memory Access (DMA) is set up
- Disk controller performs the physical read or write based on commands from DMA and file system
- If its read, Disk -> DMA, for writes, DMA -> Disk



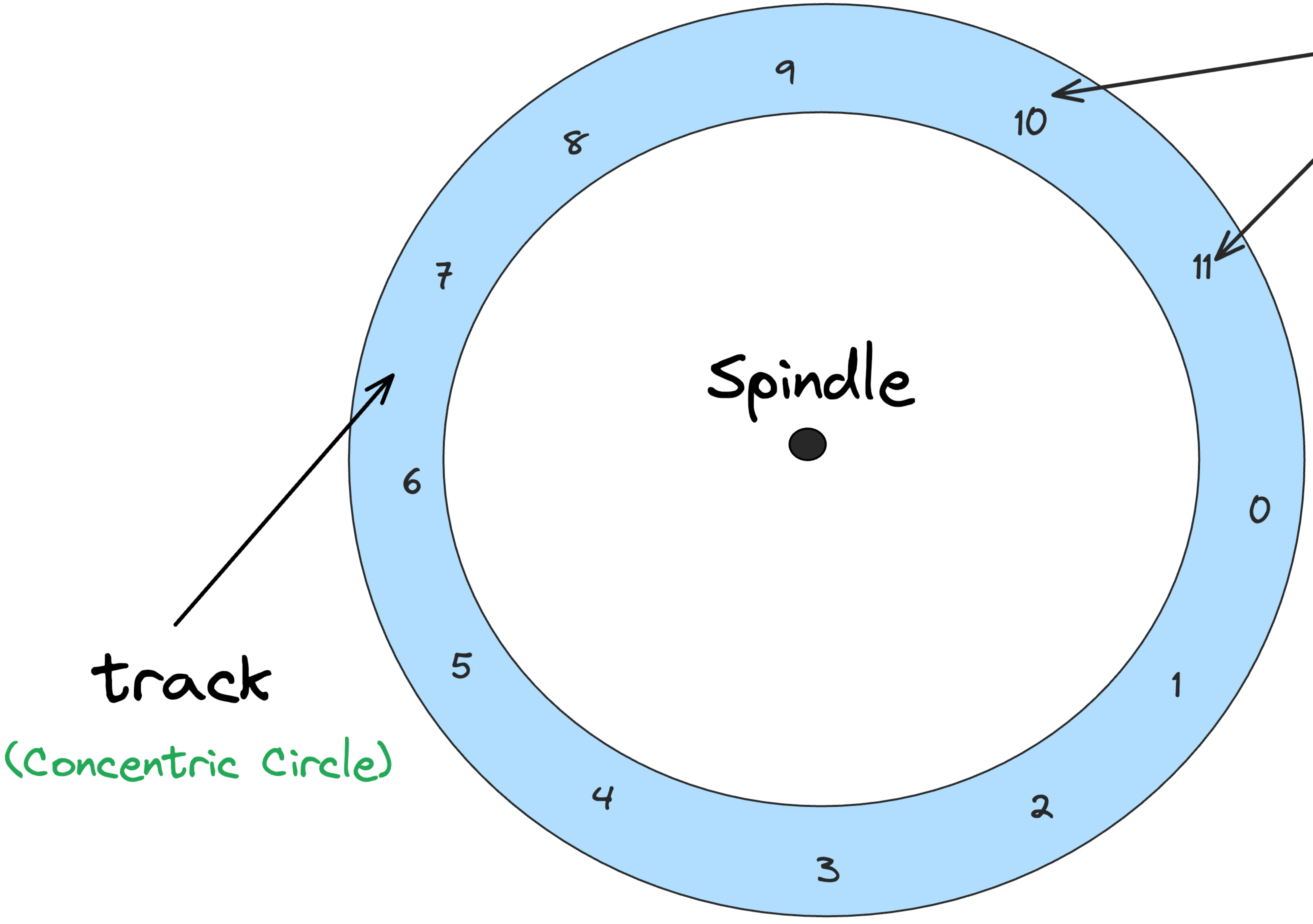
Hard Disk and How it works!

- Main form of storage in computer systems over decades
- How to store and access data?
    - How do modern hard disks store data?
        - What is the interface?
    - How is the **data laid out** and **accessed?**
    - How does **disk scheduling** improve performance?

Interface to Hard disk

- Drive consists of a number of 512 byte blocks each of which can be read or written
- Sectors are numbered from *0* to *n-1* on disk with n sectors - **Address space**
- Many file systems read or write **4 KB** at a time (or more)
- The main guarantee that is provided is that a single 512 byte write is **atomic**
    - Either it happens completely or not at all
- Power loss in between can result in a portion of write becoming incomplete **(torn write)**

Basic Geometry



Sector
(Data is encoded here)

Platter
(Circular Hard Surface)

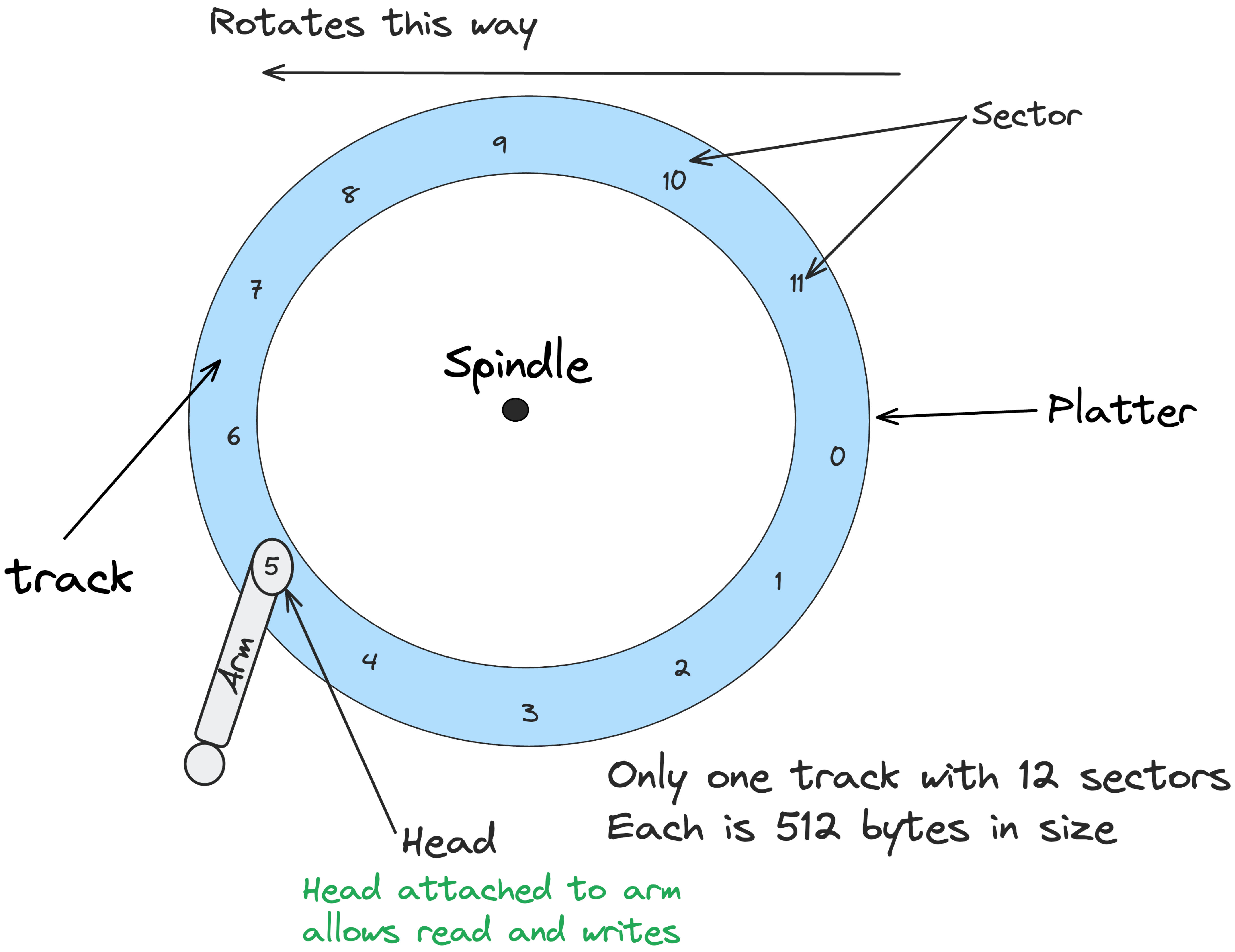
Spindle is connected to a motor => spins the platter around

1. Platter has two sides: Surface

2. Each surface has thousands of tracks tightly packed together

3. Each track is just having width of human hair

Simple Disk Drive



- Disk arm moves across to support reads and writes
- Spindle connected to a motor, spins the platter around at a constant fixed rate
- The rate is measured in rotations per minute (RPM)
- Modern values in range **7200 to 15000 RPM**
- For drive that rotates at 10,000 RPM, single rotations takes at **6ms**

Single-Track Latency: Rotational Delay

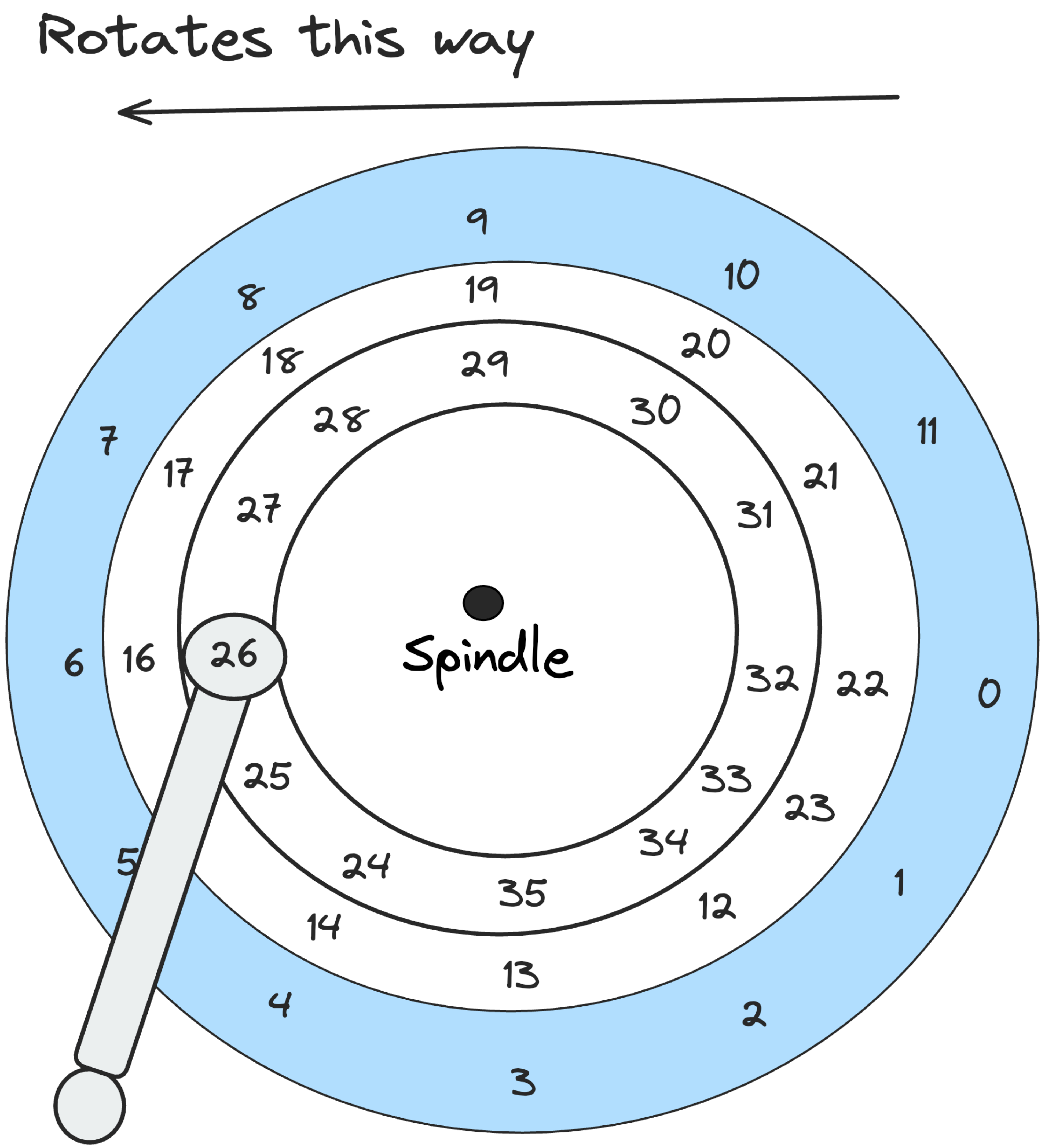
- Consider in the previous case that the access has to be done in block 0
  - Remember that the disk rotates and as soon as the head reaches the desired sector (block), it can read
  - The position of head was at sector 5
  - It has to wait for the disk to rotate and the head to reach at sector 0
  - This is known as **rotational delay** or **rotation delay**
  - What if the read access request arrived for sector 4?

Almost full rotation!!

But Disk is not just about Single Tracks

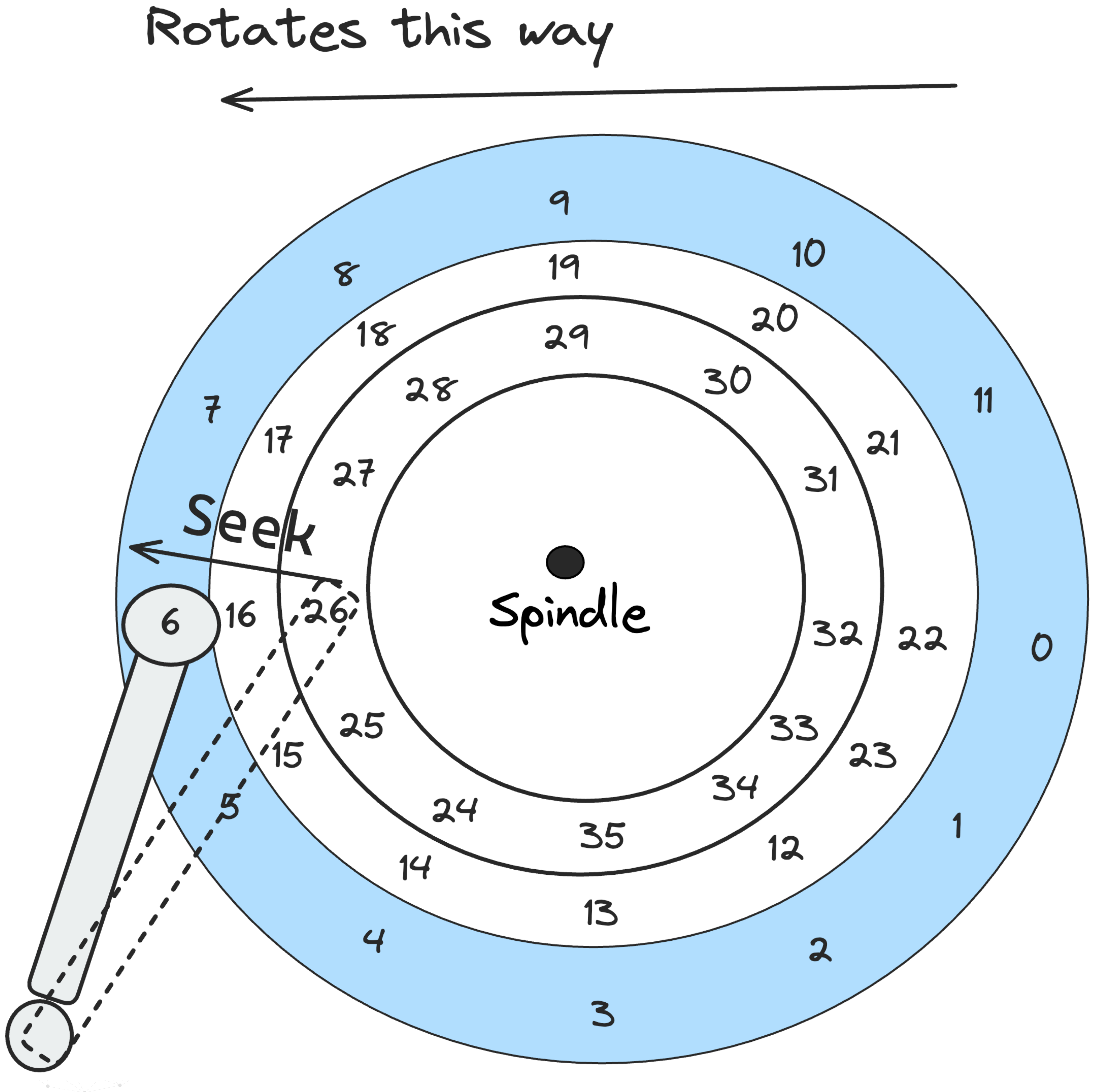
Source: Times of India

Multiple Tracks: Seek Time



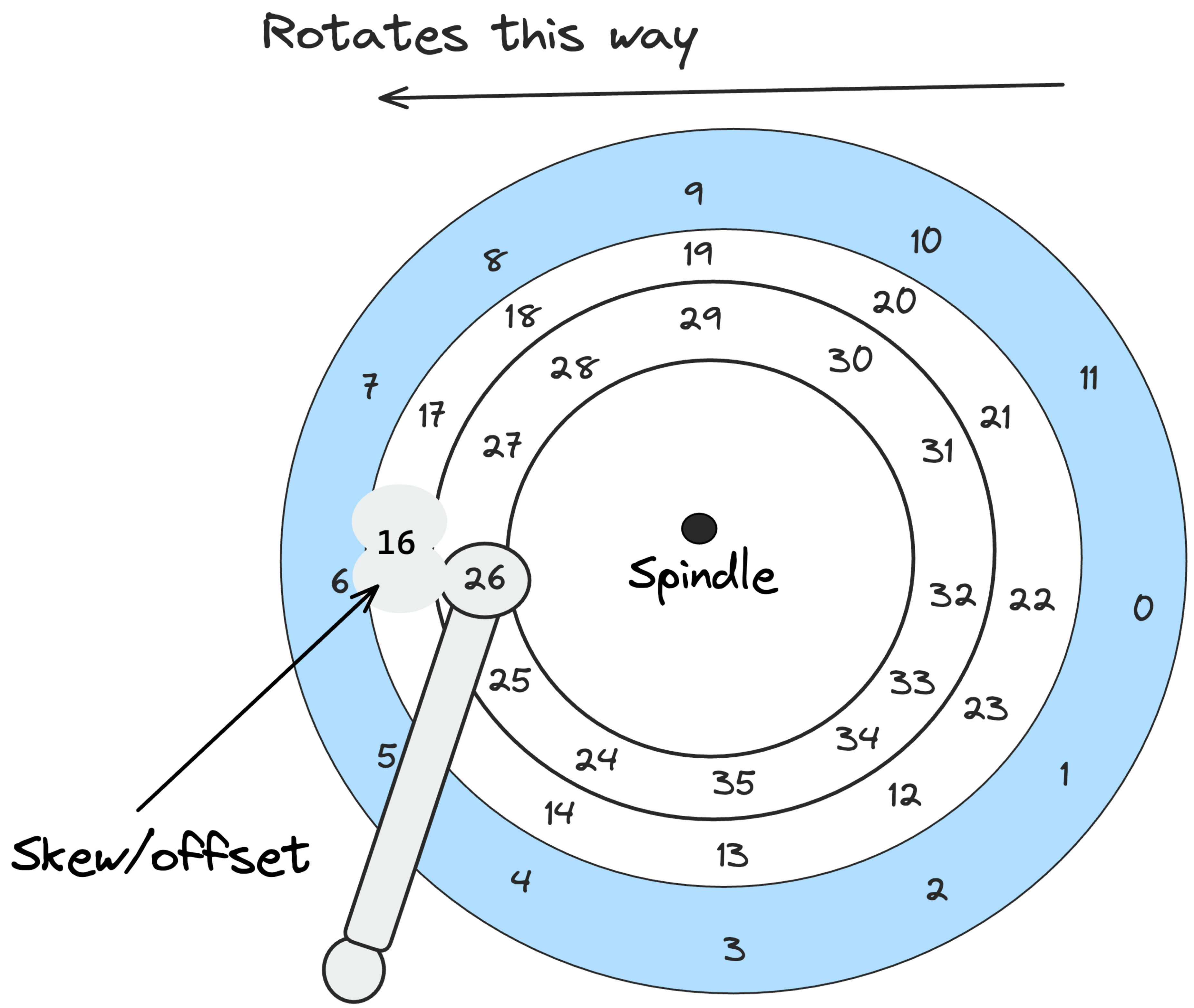
- Its not just about rotational delay
- In a real setting, a disk surface has 100s of tracks
- Read or write may happen at any block located in different tracks
- Rotation will only help in movement within a track
- Across tracks, the operation performed is **Seek**

Multiple Tracks: Seek Time



- Seek is a costly operation in terms of time
- Seek has multiple phases
  - **Acceleration:** Arm starts moving
  - **Coasting:** Arm moving at full speed
  - **Settling:** Head over correct track
- Settling time is 0.5 - 2ms High!
- Final phase of I/O is **transfer** - Read or write from surface
- **Seek, rotate and transfer** - three key phases

We may also have to consider skew



- Many drives some kind of **track skew** to make sure that sequential reads can be properly serviced
- When head moves from one track to another:
  - By the time the head moves, the desired block in the track would have got rotated
  - Head would now have to wait for a longer rotational delay
- To avoid this beginning of next track is slightly offset or **skewed**
- This is done to optimize performance

Modern Disk Drives also have Cache!

- The cache is often referred to as **Track buffer**
  - Allow the drive to quickly respond to requests
  - Small amount of memory (usually around 8 to 16 MB)
  - Can be used by drive to read from/write to the disk
- **Reads:** When reading from one sector, read all sectors in that tack and cache
  - Subsequent reads can be very fast
- **Writes:** Two choices: **Write through** and **Write back**

Write on Cache

- **Writeback** (Immediate reporting)
  - Acknowledge a write has completed as soon as the data reaches cache memory
  - It makes drives appear very fast but it can be dangerous
  - Especially if order needs to be preserved - **This can lead to problems!**
- **Writethrough**
  - Acknowledge when the write has been written to the disk
  - Here data written to cache is also written to cache and disk simultaneously
  - Overall performance here might be an issue

Some Analysis

- Disk rotates at **10,000 RPM** and has transfer rate of **100 MB/sec**
  - **How much milliseconds does a single rotation take?**
    - 1 minute = 60 seconds = 60,000 ms
    - 10,000 RPM in 60,000 ms => **6 ms for 1 rotation**
  - **How much time to transfer 512 KB blocks of data?**
    - 0.5 MB of data
    - 100 MB/sec => 0.1 MB/ms => **5 ms for 0.5 MB of data**

I/O Time of Disk

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

Rate of I/O $$R_{I/O} = \frac{Size_{transfer}}{T_{I/O}}$$

- We can perform different analysis given some characteristics
- Assume that there are two different workloads
  - Random workload
  - Sequential workload

I/O Time of Disks

- **Random Workload**
  - Issues small (4 KB) reads to random locations on the disk
  - Very common in applications like Database management systems
- **Sequential Workload**
  - Reads large number of sectors consecutively from disk
  - These are also quite common!
- Given workload, can we perform some comparison on the disk performance
  - We would also need some disk characteristics

Disk Performance Analysis

Characterestic	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

- What are some observations about the disks here?

Some Observations

- One is about performance - Cheetah
  - Drives are engineered to spin as fast as possible
  - Delivers low seek time and fast transfer rate
- Another is about capacity of the storage - Barracuda
  - Cost per byte is important
  - Drives are slow but packs as many bits into given space

Some Analysis

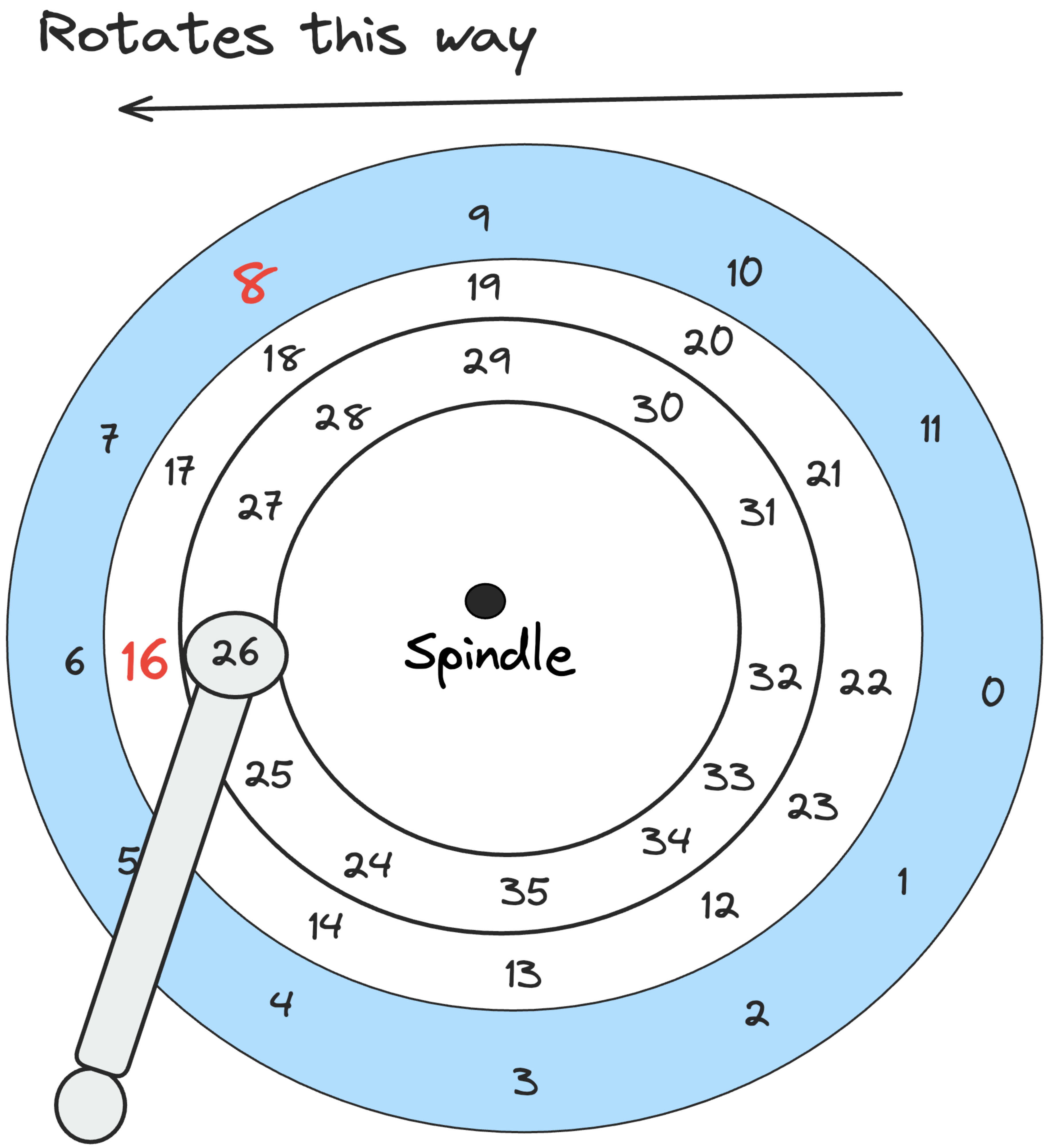
Workload	Metric	Cheetah	Barracuda
	Tseek	4 ms	9 ms
	Trotation	2 ms	4.2 ms
Random (4 KB reads)	Ttransfer	30 micros	38 micros
	TI/O	6 ms	13.2 ms
	RI/O	0.66 MB/s	0.31 MB/s
Sequential (100 MB reads)	Ttransfer	800 ms	950 ms
	TI/O	806 ms	936.2 ms
	RI/O	125 MB/s	105 MB/s

There is large difference in performance between **high-end performance** drives and **low-end capacity** drives

Disk Scheduling

- OS plays a role in the order of I/O issued to disk
- Given a set of I/O requests, is it possible to schedule them in an order
  - Disk scheduler decides which I/O request to schedule next!
  - Decides which request to schedule to improve performance
- How can disk scheduler make a guess on what request shall be better?
  - Remember: Disk scheduler can estimate how much time each request can take
  - Parameters like rotational delay, transfer, seek are known/easily estimated

SSTF: Shortest Seek Time First



Scheduling Requests: 8, 16

- Orders the queue of I/O requests by track
  - Pick the block in the queue from nearest track to complete first
- In this case
  - The closest one is 16
  - Followed by 8
  - Schedule: 16 -> 8

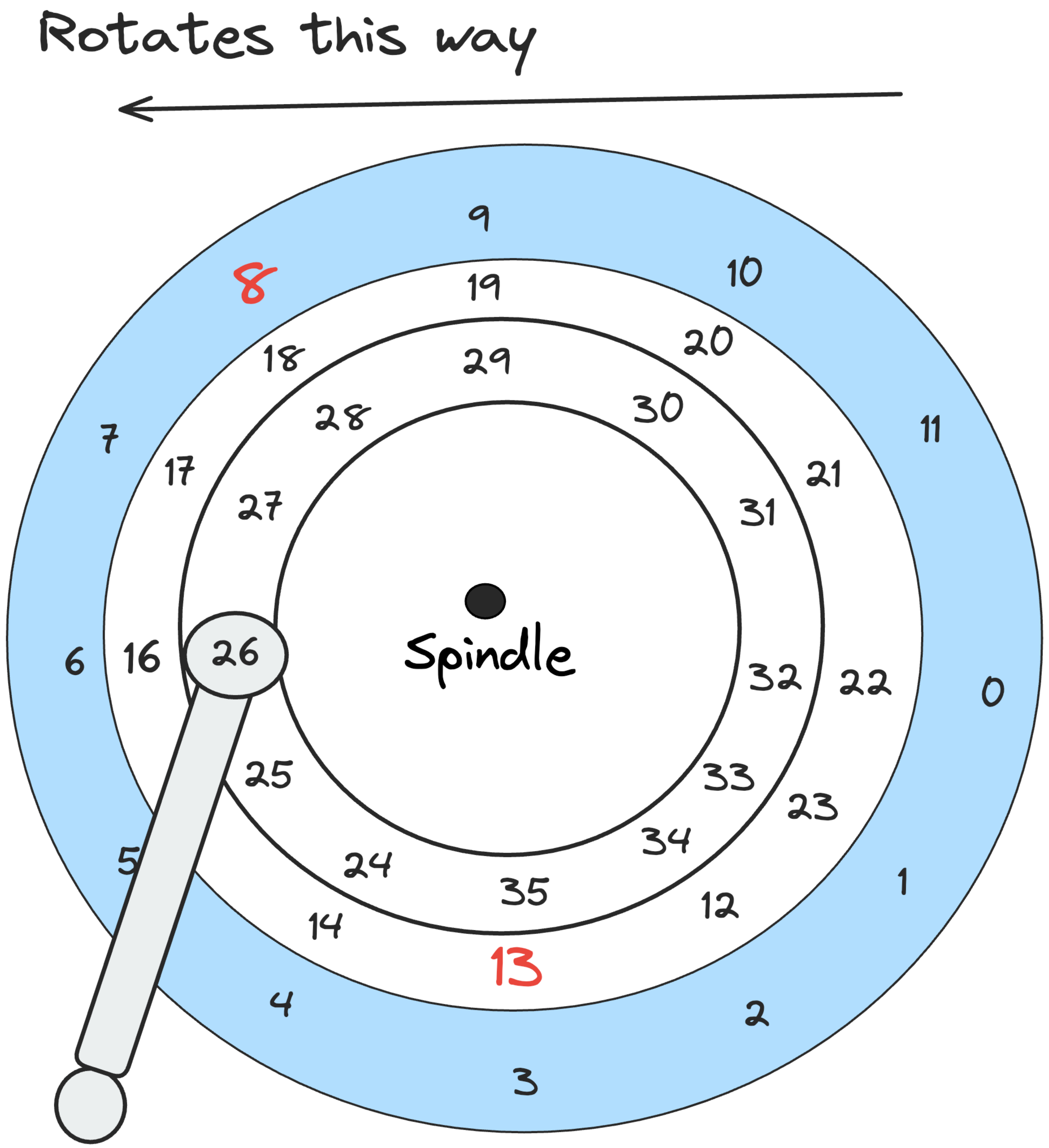
Two main issues

- Drive geometry is not available to the OS
  - It sees everything as blocks
  - One way out: Implement something like **Nearest Block First (NBF)**
- There is another main issue: **Starvation!**
  - If there is a steady stream of requests to inner tracks
  - Request to outer tracks may be ignored completely leading to starvation
- Can we do something better to avoid starvation?

Elevator (a.k.a SCAN or C-SCAN)

- Simply move back and forth, servicing the requests in order
  - **Sweep:** A Single Pass across the disk
  - If a request comes for a block on a track that has already been serviced in this sweep, it has to wait in a queue till the next sweep
- **F-SCAN**
  - Freeze the queue to be serviced when doing a sweep
  - Avoid starvation of far-away requests
- **C-SCAN (Circular scan)**
  - Sweep from inner-to-outer and outer-to-inner, etc.

Shortest Positioning Time First



- Rotation and seek needs to be considered
- Which to give preference - **It depends!**
- Assume I/O requests to 8 and 13
  - Seek and rotation both are time consuming
  - Assume that seek is faster than rotation
    - 8 -> 13 makes more sense
  - If rotation is faster than seek
    - 13 -> 8 makes more sense

SCAN ME

Thank you

Course site: karthikv1392.github.io/cs3301_osn
Email: karthik.vaidhyanathan@iiit.ac.in
Twitter: @karthi_ishere