CS 1217

Deadlock Wrap, Introduction to Disks and File Systems

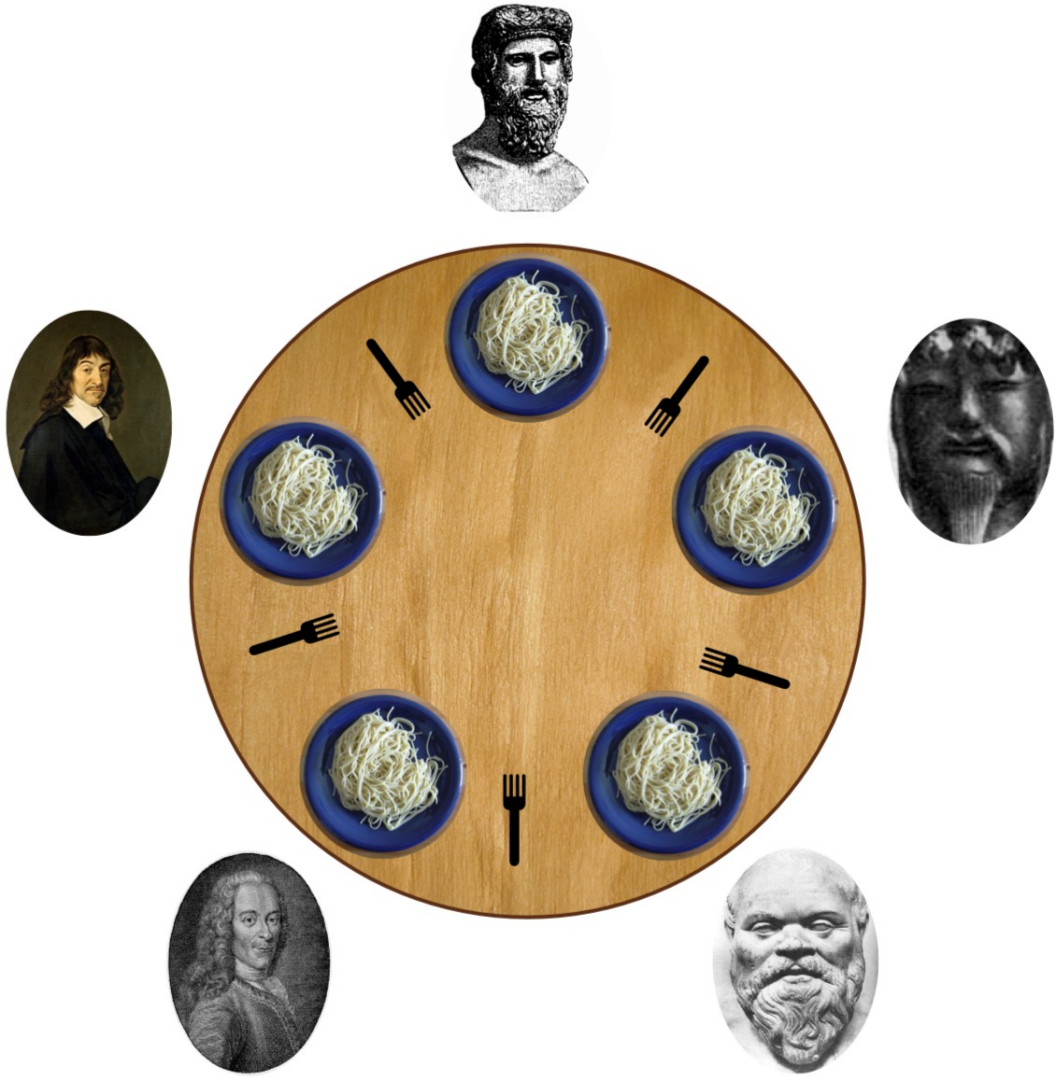
Deadlock

- **Deadlock** occurs when a thread or set of threads are waiting for each other to finish and thus nobody ever does.

Conditions for Deadlock

- A deadlock **cannot occur** unless the following conditions are met:
  - **Protected access** to shared resources, which implies waiting.
  - **No resource preemption**, meaning that the system cannot forcibly take a resource from a thread holding it.
  - **Multiple independent requests**, meaning a thread can hold some resources while requesting others.
  - **Circular dependency graph**, meaning that Thread A is waiting for Thread B which is waiting for Thread C which is waiting for Thread D which is waiting for Thread A.

Dining Philosophers “Problem”



Making sure the Philosophers Eat

- Breaking deadlock conditions usually requires eliminating one of the **requirements** for deadlock.
- **Don’t wait**: don’t sleep if you can’t grab the second fork + put down the first.
- **Break cycles**: usually by acquiring resources in a **well-defined order**. Number forks 0–4, always grab the higher-numbered fork first.
- **Break out**: detect the deadlock cycle and forcibly take away a resource from a thread to break it. (Requires a new mechanism.)
- **Don’t make multiple independent requests**: grab **both** forks at once. (Requires a new mechanism.)

Deadlock vs. Starvation

- **Starvation** : condition in which one or more threads do not make progress.
- Starvation differs from deadlock in that **some** threads make progress and it is, in fact, those threads that are preventing the "starving" threads from proceeding.

Deadlock vs. Race Condition

- What is better: a **deadlock** (perhaps from overly careful synchronization) or a **race condition** (perhaps from a lack of correct synchronization)?

Choice of Tool Matters

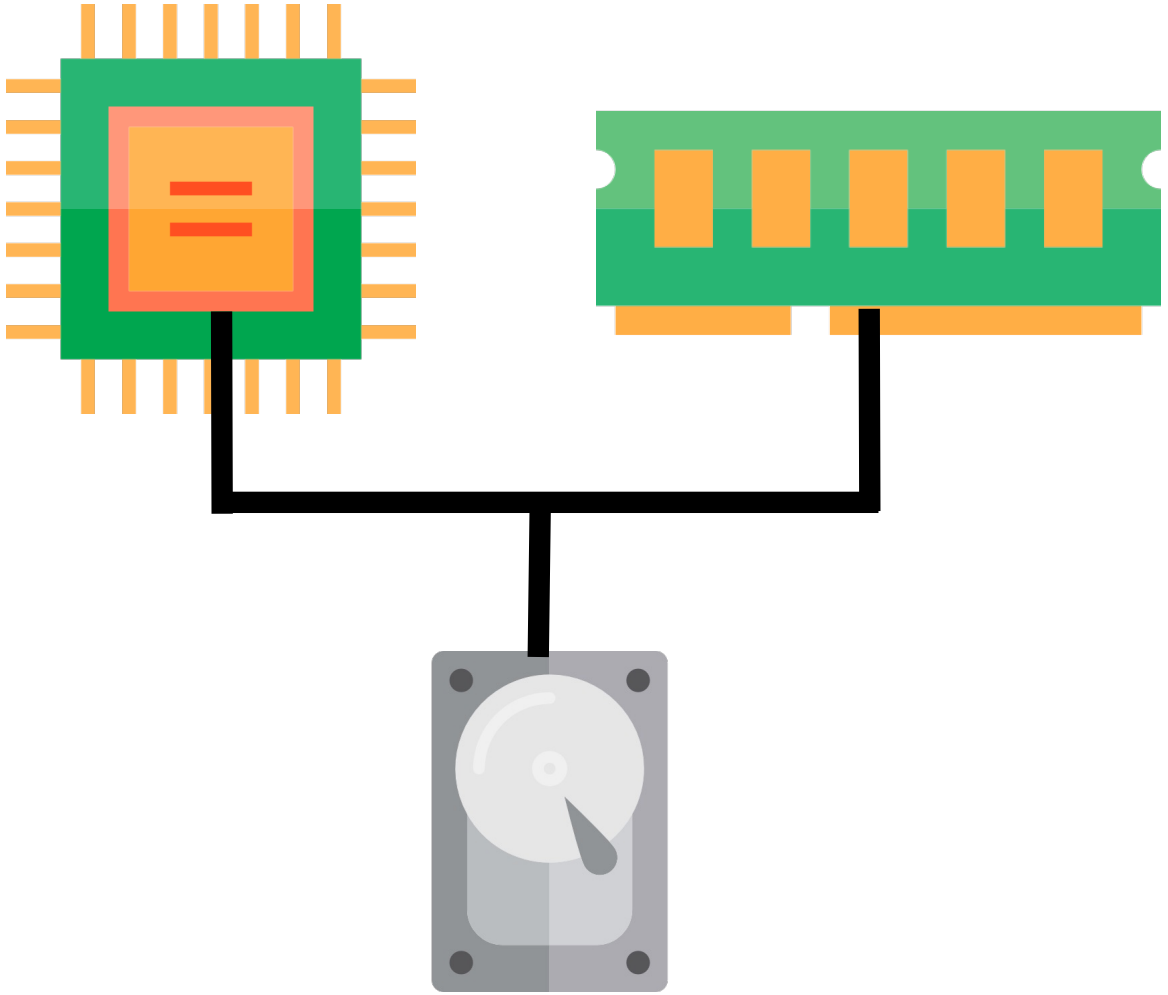
- Most problems can be solved with a **variety** of synchronization primitives.

- However, there is usually **one primitive** that is more appropriate than the others.

General Approach to Synchronization Problems

- Identify the constraints.
- Identify shared state.
- Choose a primitive.
- Pair waking and sleeping/ acquiring and releasing
- Look out for multiple resource allocations: can lead to deadlock.
- Walk through simple examples and corner cases **before** beginning to code.

The Computer System



Introduction to Disks

- **Persistent storage**: storage that does not lose its contents when power is turned off/is unavailable.

- Two types are used today:
- Hard Disk Drive (HDD): Uses spinning disks made of magnetic material
- Solid State Drive (SSD): Uses non-volatile memory (Flash)

HDDs

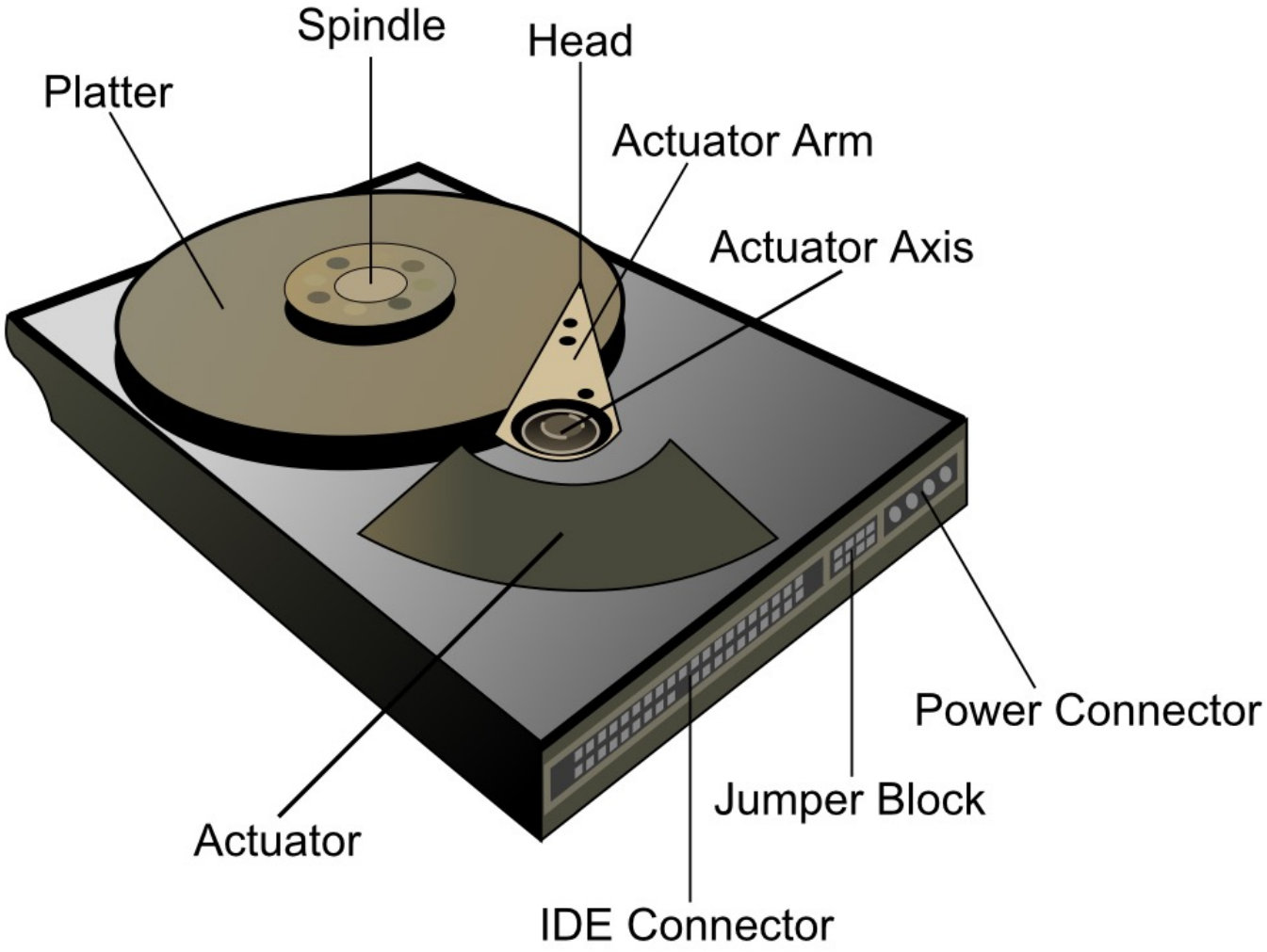
- Why study HDDs when SSD is the future?
- Bulk of world's storage resides in HDDs, even true for datacenters

- There is a lot to be learnt from **systems design principles** at work here and admire the maturity of file system design

Disk Components

- **Platter**
  - A circular flat disk on which magnetic data is stored
  - Constructed of a rigid non-magnetic material coated with a *very thin* layer of magnetic material
  - Can have data written on both sides.
- **Spindle**
  - the drive shaft on which multiple **platters** are mounted and spun between 4200–15,000 RPM.
- **Head**
  - the actuator that reads and writes data onto the magnetic surface of the **platters** while rotating at tens of nanometers over the platter surface.

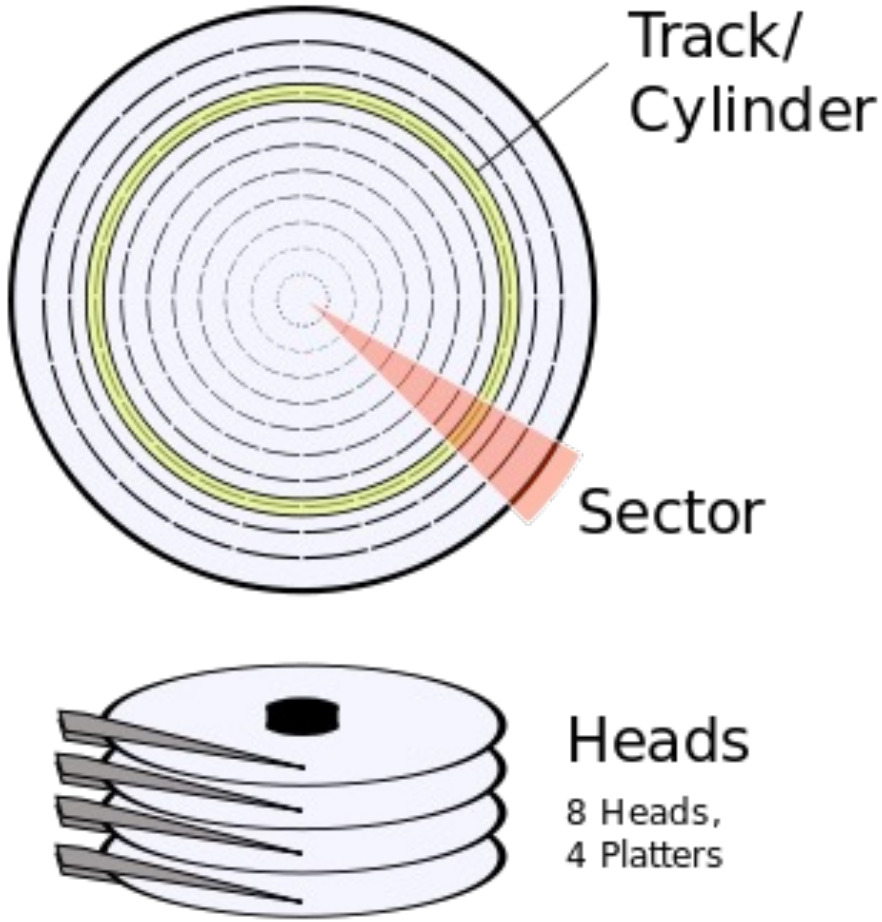
Disk Parts


Spindle
Head
Platter
Actuator Arm
Actuator Axis
Power Connector
Jumper Block
Actuator
IDE Connector

More Terminology

- **Track:** think of a lane on a race track running around the platter.
- **Sector:** resembles a slice of pie cut out of a single platter.
- **Cylinder:** imagine the intersection between a cylinder and the set of platters. Composed of a set of vertically-aligned tracks on all platters.

Disk Organization



HDD in Action

https://www.youtube.com/watch?v=9eMWG3fwiEU

Challenges with Disks

- They have mechanical moving parts => **slow**

- Flash is fundamentally faster, since information storage and retrieval is governed by how fast electrons can do their job.

Disks becoming irrelevant?

Disks for Data Centers

White paper for FAST 2016

Eric Brewer, Lawrence Ying,
Lawrence Greenfield, Robert Cypher, and Theodore Ts'o
Google, Inc.

February 23, 2016
Version 1.1, revised February 29, 2016

Online at: http://research.google.com/pubs/pub44830.html

"Disks form the central element of Cloud-based storage, whose demand far outpaces the considerable rate of innovation in disks. Exponential growth in demand, already in progress for 15+ years, implies that most future disks will be in data centers and thus part of a large collection of disks."

Disks are Slow

- Disks frequently bottleneck other parts of the operating system.

- Operating systems play some of our usual games with disks to try and hide disk latency.
  - Use the **past to predict the future**.
  - Use a **cache**.

- Memory latency is hidden **transparently** by the processor.
  - OS is directly involved in hiding disk latency

Disk Latency Distribution

- Disk operations require many steps for reads/writes
- ***Command Issue***
  - Tell the device what needs to be done; which platter? Which head?
- Seek Time
  - Time for the drive head to move to appropriate track
- ***Stabilization/Settle Time***
  - latency for the heads to stabilize
- Rotation Delay
  - Latency for the right sector to come under the head
- Transfer Time
  - Latency to Read / Write data

Need for File Systems

- Low-level disk interface is very limited:
- Requires reading and writing entire 512-byte blocks. **Why**?
- No notion of files, directories, etc.

- File systems take what disks offer, and create the ***file system abstraction*** in software

Flash is Great, right?

- Not really
- The underlying technology has its own set of limitations

- Before writing, large chunks might need to be “erased”
- In-place writing is not possible!
- Cannot write to a cell more than a fixed number of times!

- Lesson: Every technology has its limitations, need to write s/w with that in mind

Concepts of Files

- The semantics of file have a variety of sources what are worth separating:
- **Just a file:** the minimum it takes to be a file.
- **About a file:** what other useful information do most file systems *typically* store about files?
- **Files and processes:** what additional properties does the UNIX file system interface introduce to allow user processes to manipulate files?
- **Files together:** given *multiple* files, how do we organize them in a useful way?