CS 471 Operating Systems

Yue Cheng

George Mason University

Fall 2019

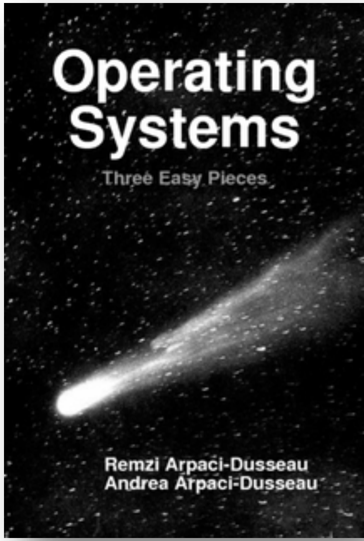
Introduction

- Instructor of Section 002
  - Dr. Yue Cheng (web: cs.gmu.edu/~yuecheng)
  - Email: yuecheng@gmu.edu
  - Office: 5324 Engineering
  - Office hours: T 15:00pm-16:30pm
  - Research interests: Distributed and storage systems, serverless and cloud computing, operating systems

Introduction

- Instructor of Section 002
  - Dr. Yue Cheng (web: cs.gmu.edu/~yuecheng)
  - Email: yuecheng@gmu.edu
  - Office: 5324 Engineering
  - Office hours: T 15:00pm-16:30pm
  - Research interests: Distributed and storage systems, serverless and cloud computing, operating systems
- Teaching assistant
  - Zhemin An
  - Email: zan2@gmu.edu
  - Office hours:
    - W 10am-12pm; R: 3:30pm-5:30pm

Administrivia



- **Required** textbook
  - **Operating Systems: Three Easy Pieces,**
    By Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau
- Recommended textbook
  - **Operating Systems Principles & Practices**
    By T. Anderson and M. Dahlin
- **Prerequisites are enforced!!**
  - **CS 310** Data Structures
  - **CS 367** Computer Systems & Programming -OR-
  - **ECE 445** Computer Organization
  - **Be comfortable with C programming language**
- Class web page
  - https://cs.gmu.edu/~yuecheng/teaching/cs471_fall19/index.html
  - Class materials will all be available on the class web page

Administrivia (cont.)

- Syllabus
  - https://cs.gmu.edu/media/syllabi/Fall2019/CS_471ChengY002.html
- Grading
  - 30% projects
  - 10% homework
  - 15%+15% two midterm exams
  - 30% final exam
- Reminders
  - Honor code
  - Late policy: 15% deducted each day. No credit after 3 days

Course format

- (Review) + lecture + (worksheets)
  - A short overview of the previous lecture to make sure the old content is not completely forgotten
  - Worksheet practices to make sure the lecture is well understood

OS/161 Projects

- Three coding projects
  - Project 0: Intro to OS/161 (due Sep 13) – 6%
  - Project 1: Synchronization – 12%
  - Project 2: Syscalls and processes – 12%

Homework and system programming assignments

- Two written homework assignments

- + Extra credits for some system programming assignments
  - For those who are ahead of the schedule and are interested in more work :)
  - Expect more as the semester approaches…
    - A baseline hash table based key-value store implemented using C
    - Adding multi-threading and concurrency support
    - Adding memory management

What is an OS?

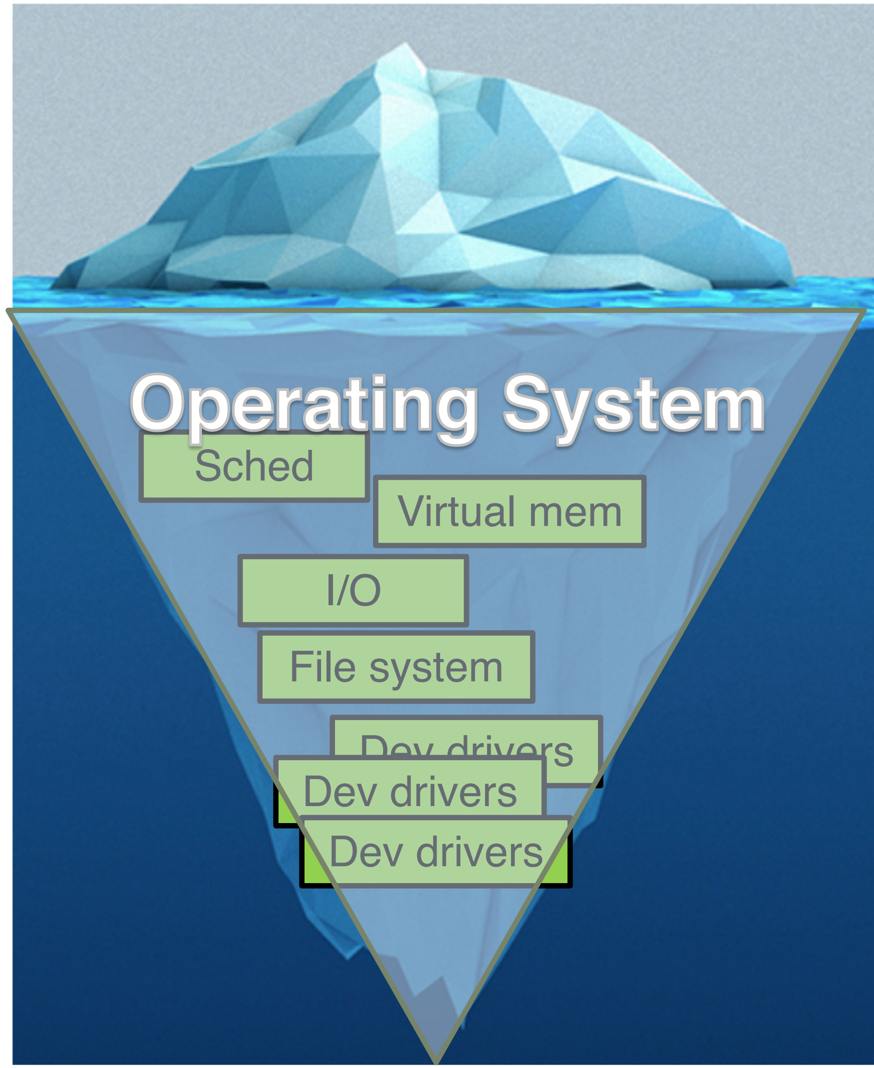
What is an OS?

- OS manages resources
  - Memory, CPU, storage, network
  - Data (file systems, I/O)

- Provides low-level abstractions to applications
  - Files
  - Processes, threads
  - Virtual machines (VMs), containers
  - …

OS abstracts away low-level details

OS abstracts away low-level details

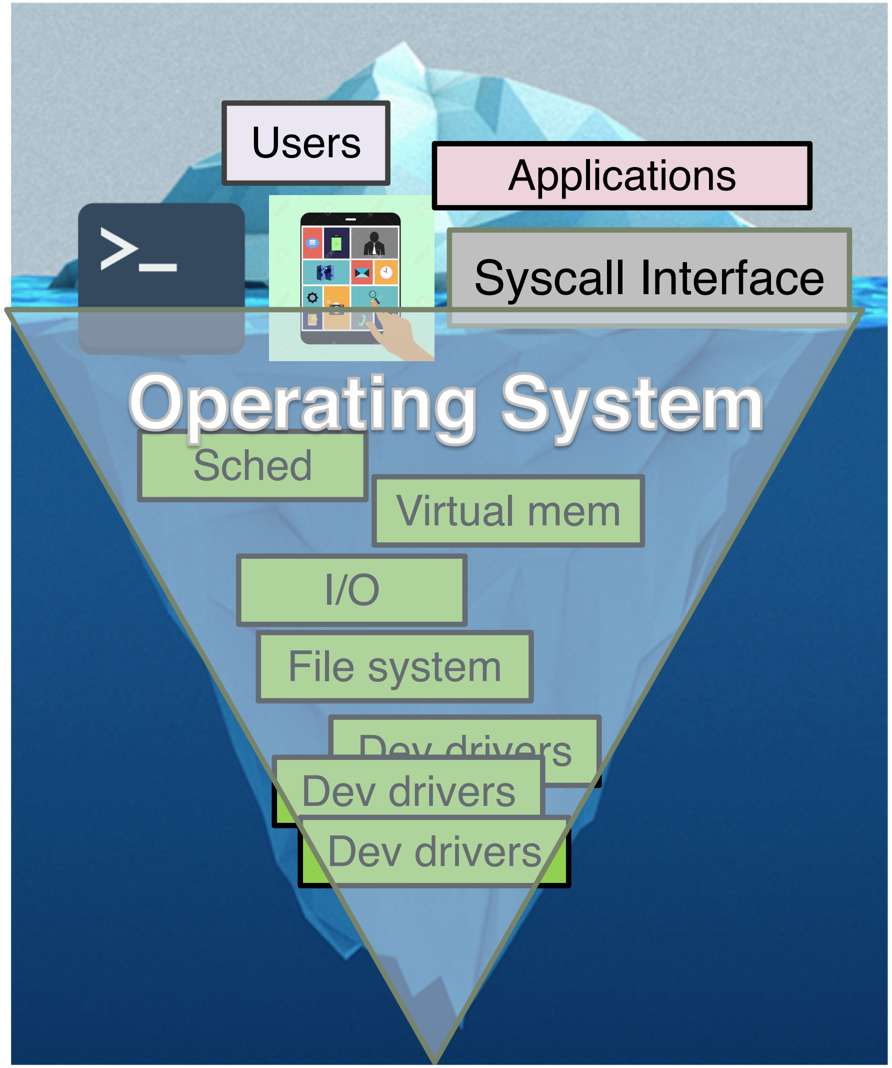


- Under the surface
  - Complex and dirty implementations of abstractions and a lot more…

OS abstracts away low-level details

- User's perspective
  - User interface:
    - Terminal, GUI
  - Application interface:
    - System calls

- Under the surface
  - Complex and dirty implementations of abstractions and a lot more…



The goals of an OS

- OS manages resources
  - Memory, CPU, storage, network
  - Data (file systems, I/O)

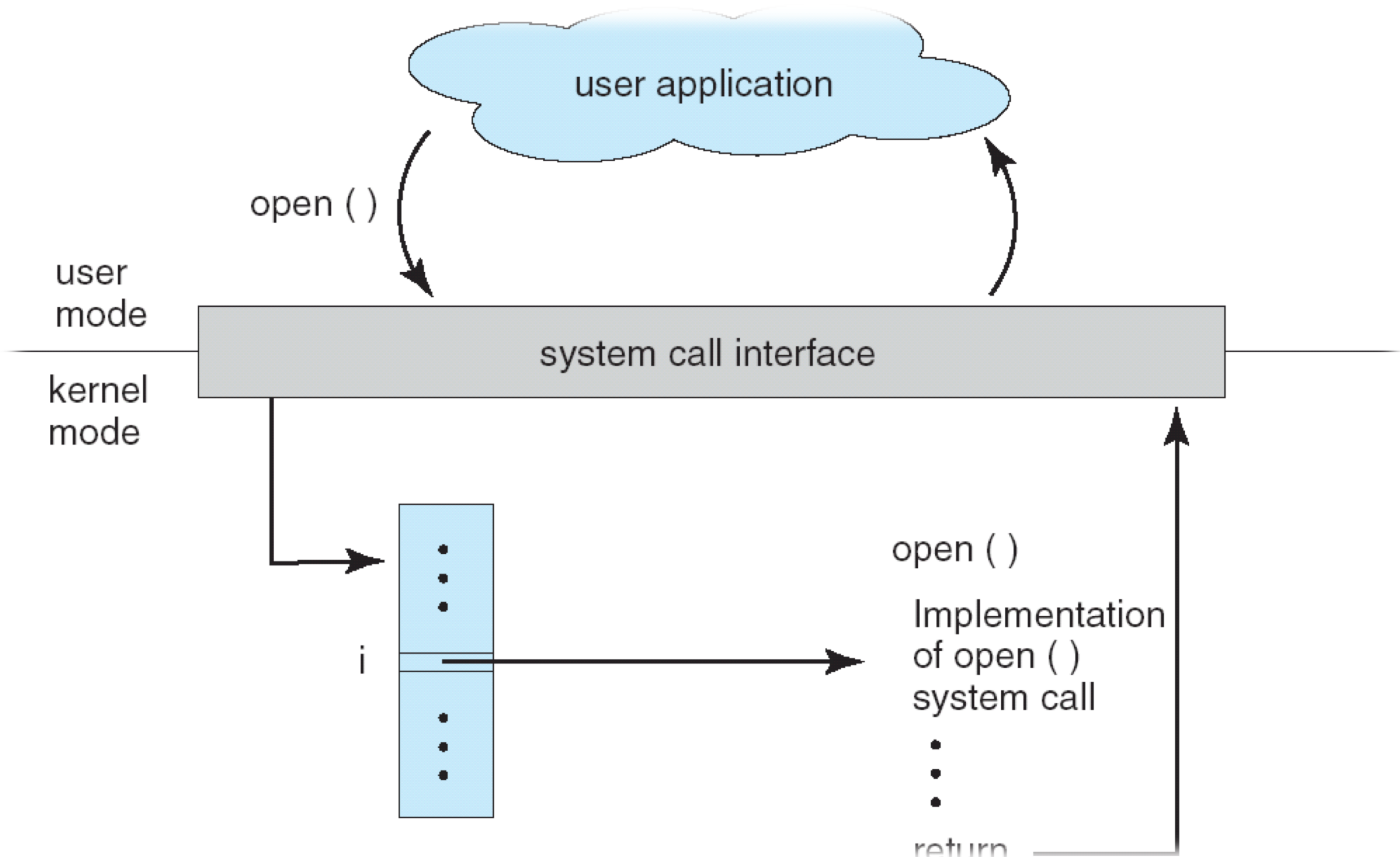
- Provides low-level abstractions to applications
  - Files
  - Processes, threads
  - Virtual machines (VMs), containers
  - …

- **Goals**
  - **Resource efficiency** (resource virtualization)
  - **Ease-of-use** (interfaces)
  - **Reliability** (user-kernel space separation)

System Calls

- System calls provide the interface between a running program and the operating system
  - Generally available in routines written in C and C++
  - Certain low-level tasks may have to be written using assembly language
- Typically, application programmers design programs using an application programming interface (API)
- The runtime support system (runtime libraries) provides a system-call interface, that intercepts function calls in the API and invokes the necessary system call within the operating system
- Major differences in how they are implemented (e.g., Windows vs. Unix)

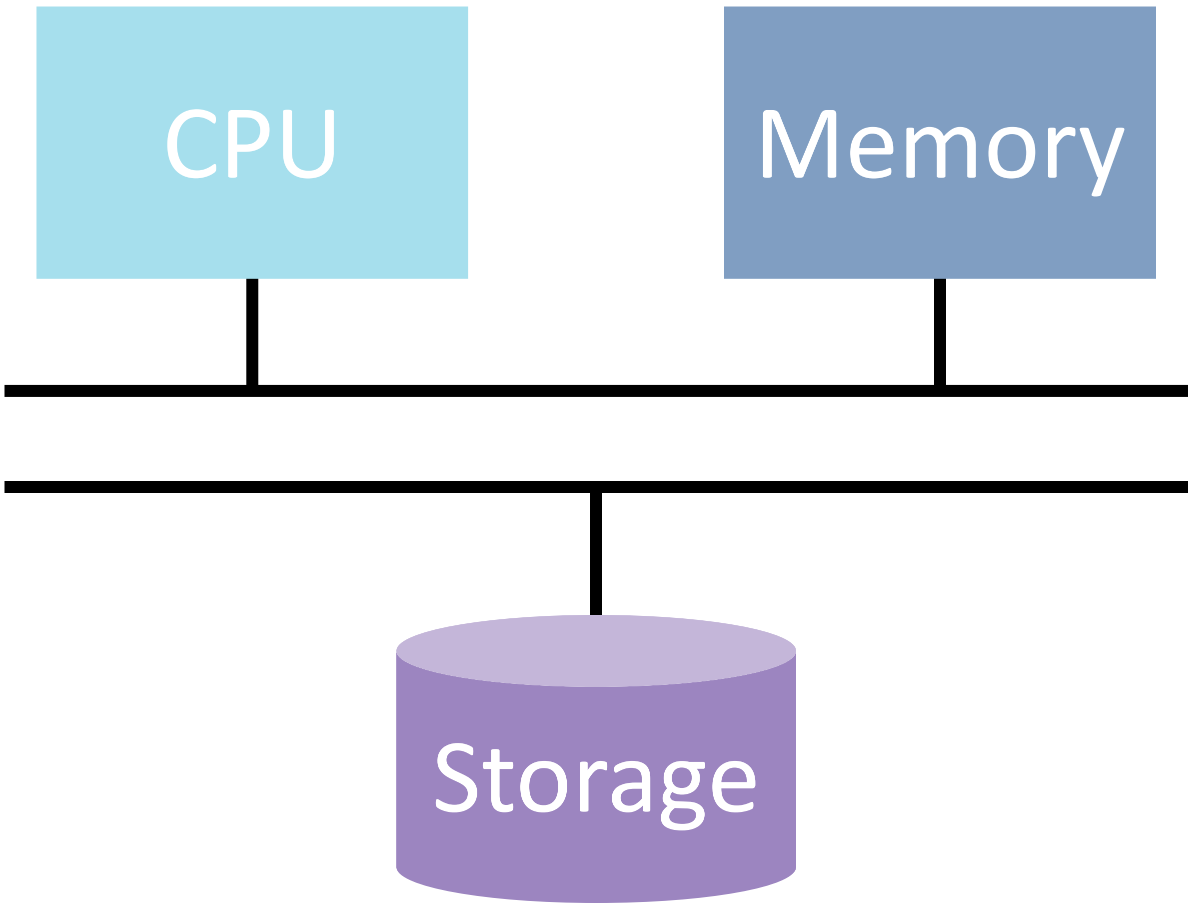
Example System Call Processing



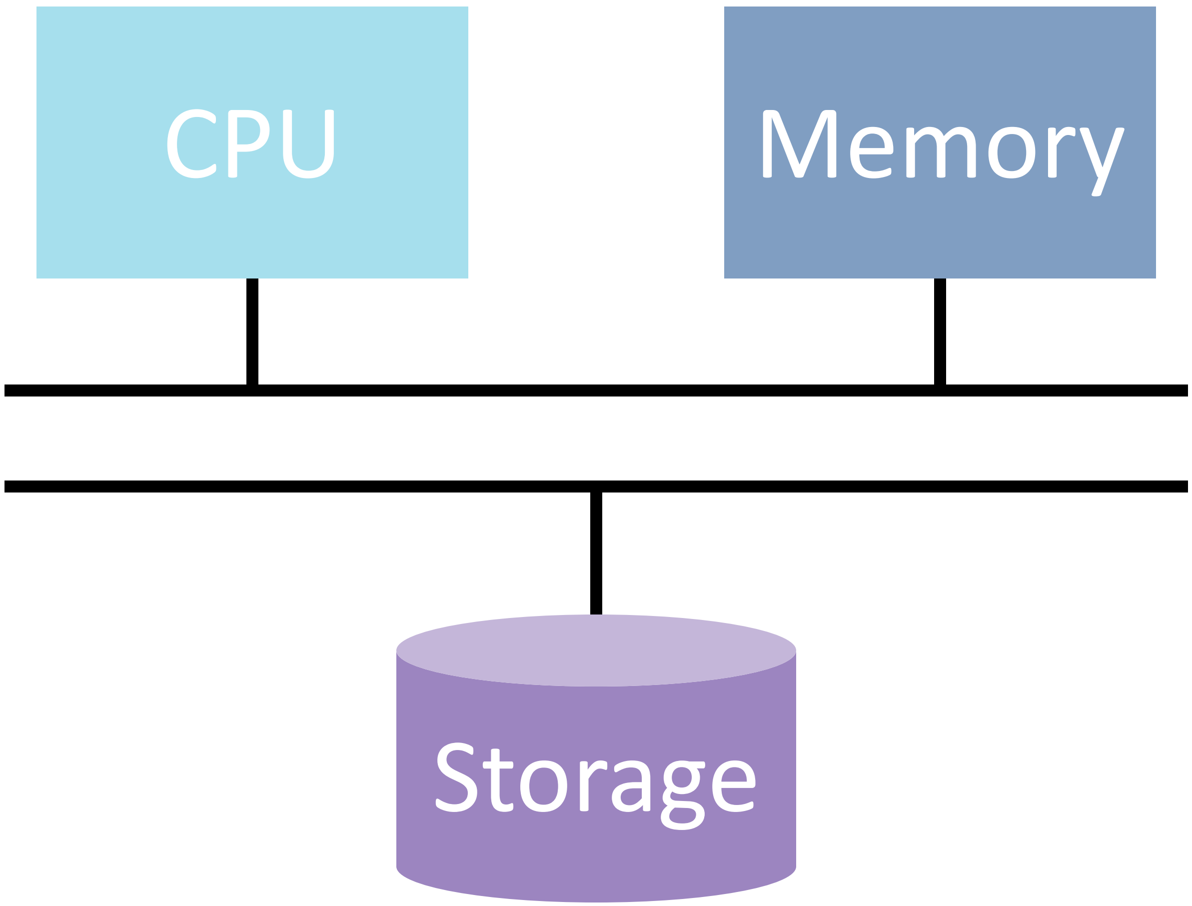
Major System Calls in Linux: File Management

- `fd = open(file, how, …)`
  - Open a file for reading, writing, or both
- `s = close(file)`
  - Close an open file
- `n = read(fd, buf, nbytes)`
  - Read data from a file into a buffer
- `n = write(fd, buf, nbytes)`
  - Write data from a buffer into a file
- `pos = lseek(fd, offset, whence)`
  - Move the file pointer
- `s = stat(name, &buf)`
  - Get a file's status info

3 Major Topics

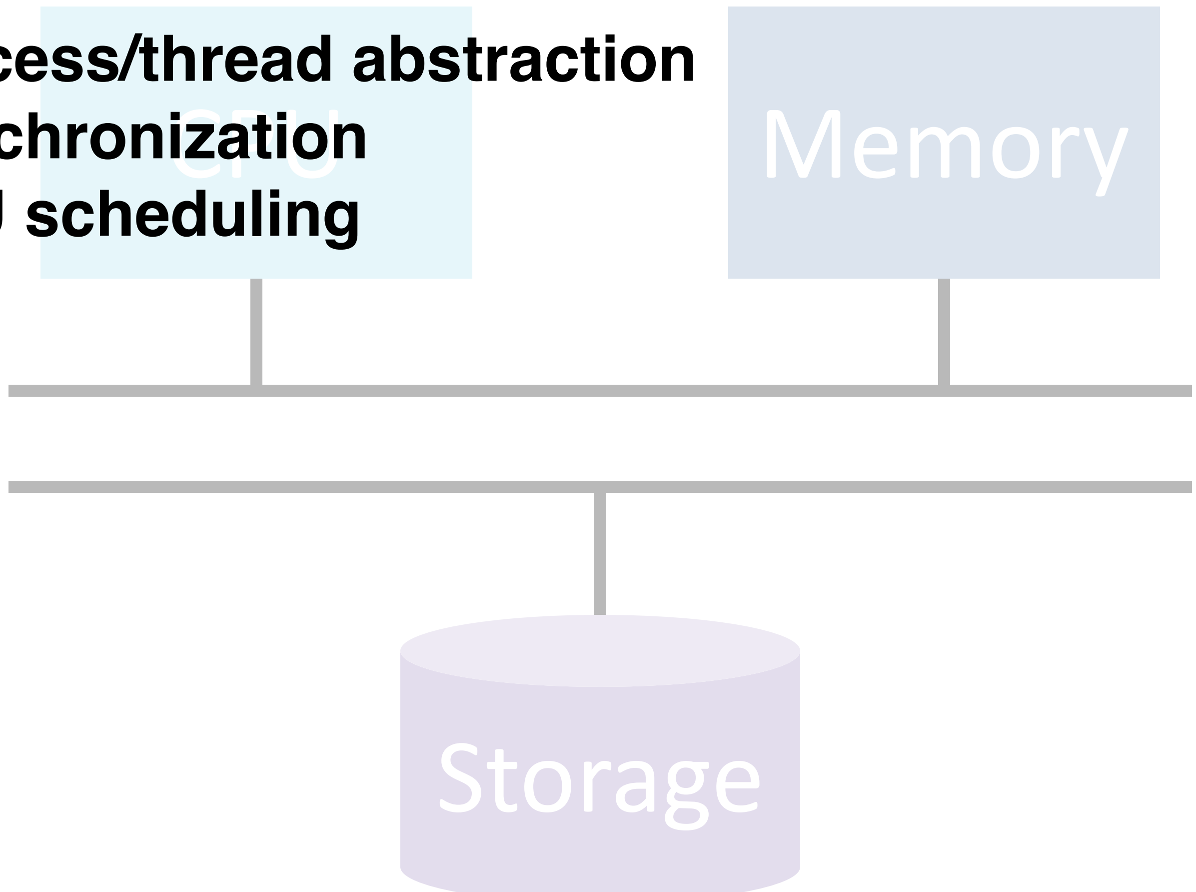


OS Provides Virtualization on Hardware



Topic 1: Concurrency, Synchronization, and CPU Scheduling

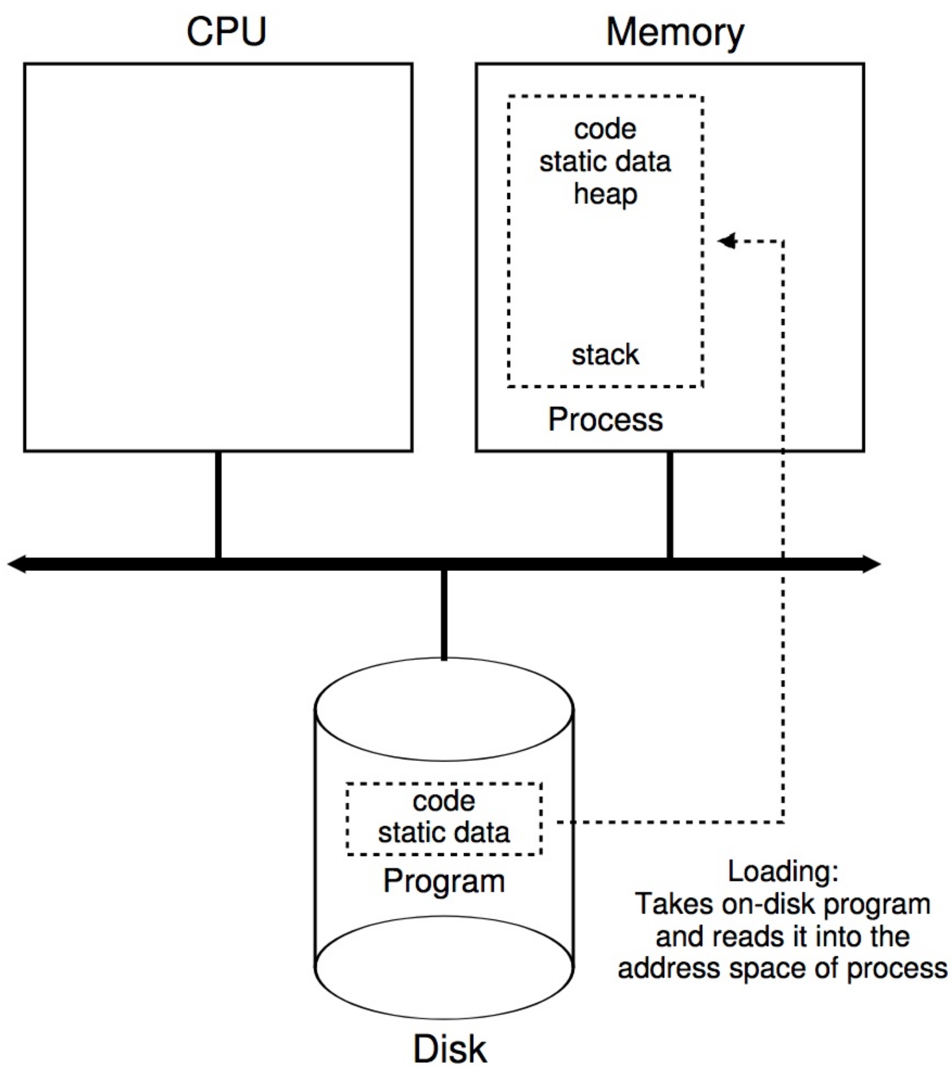
- **Process/thread abstraction**
- **Synchronization**
- **CPU scheduling**



Process Abstraction

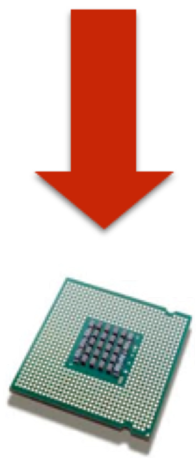
- **A process is a program in execution**
  - It is a unit of work within the system. A program is a **passive entity**, a process is an **active entity**.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- A software system may have many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads

Loading from Program to Process



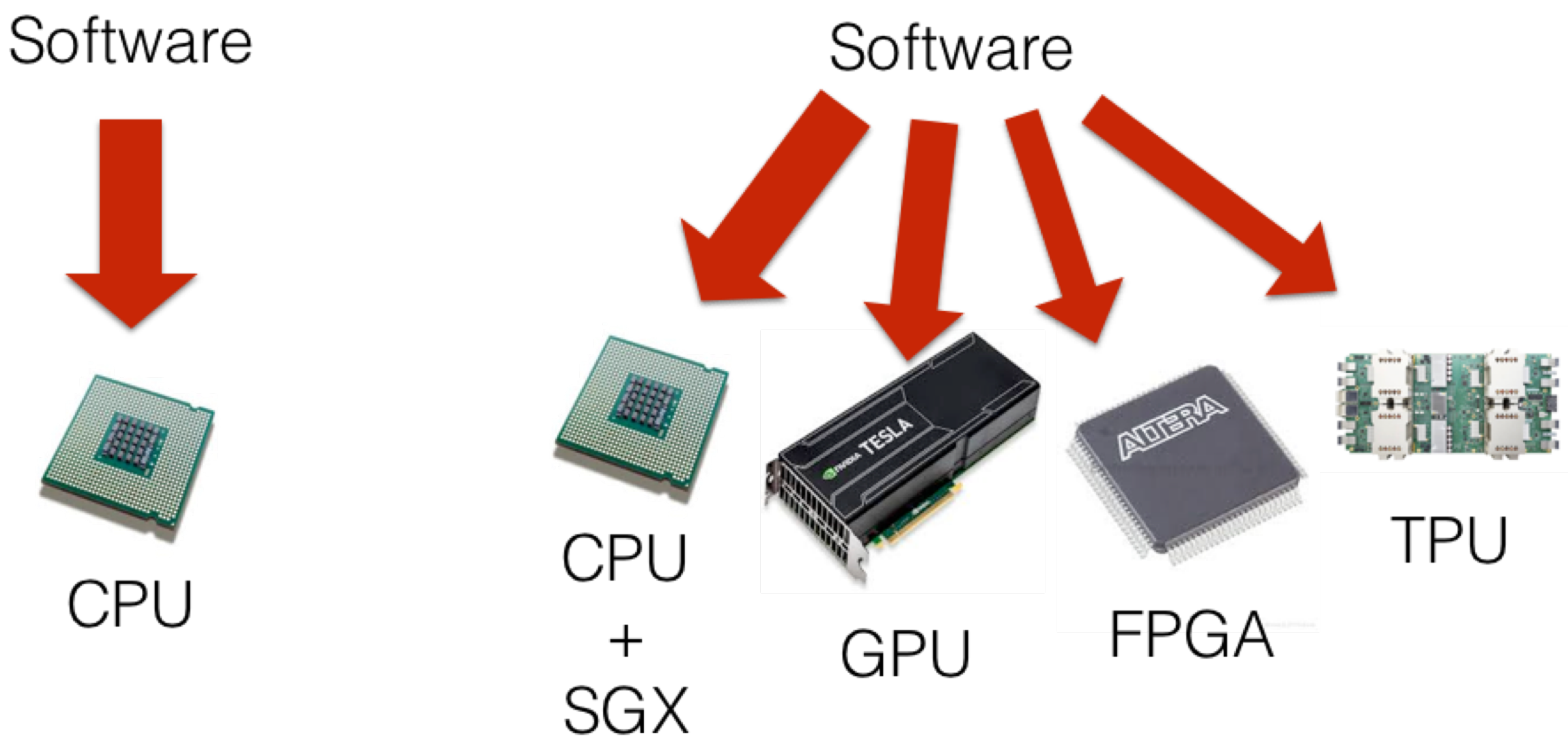
Computation – Increased Complexity

Software



CPU

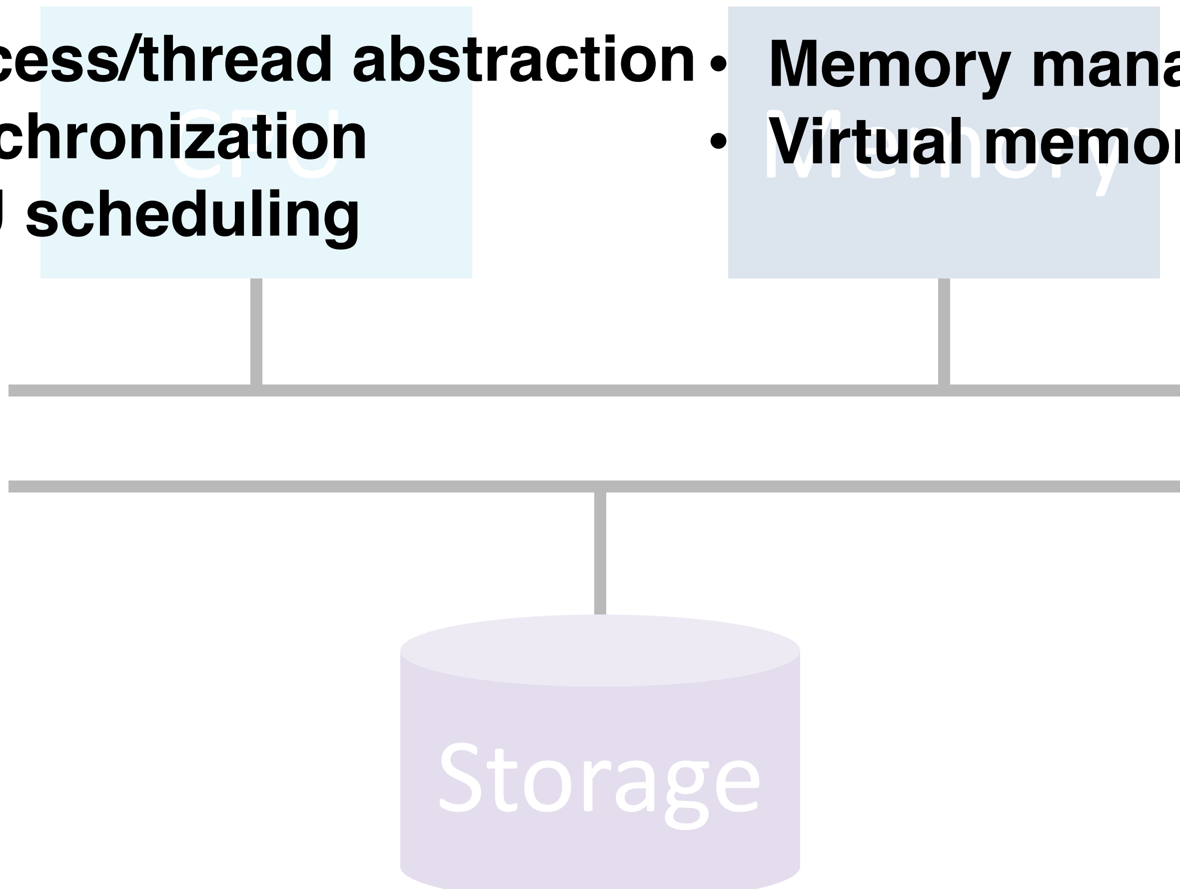
Computation – Increased Complexity



Topic 2: Memory Management and Virtual Memory

- **Process/thread abstraction**
- **Synchronization**
- **CPU scheduling**

- **Memory management**
- **Virtual memory**



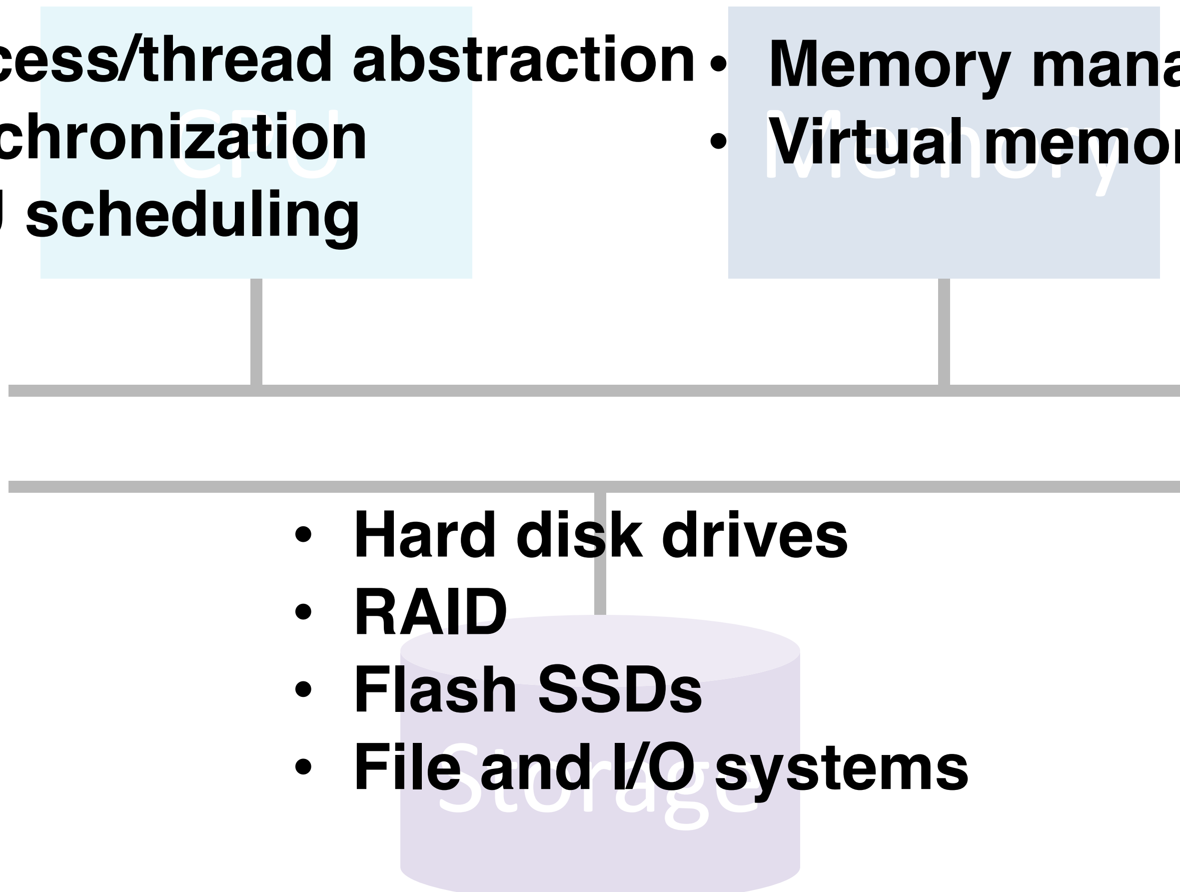
Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed
- **Virtual memory** management is an essential part of most operating systems

Topic 3: Storage, I/O, and Filesystems

- **Process/thread abstraction**
- **Synchronization**
- **CPU scheduling**

- **Memory management**
- **Virtual memory**

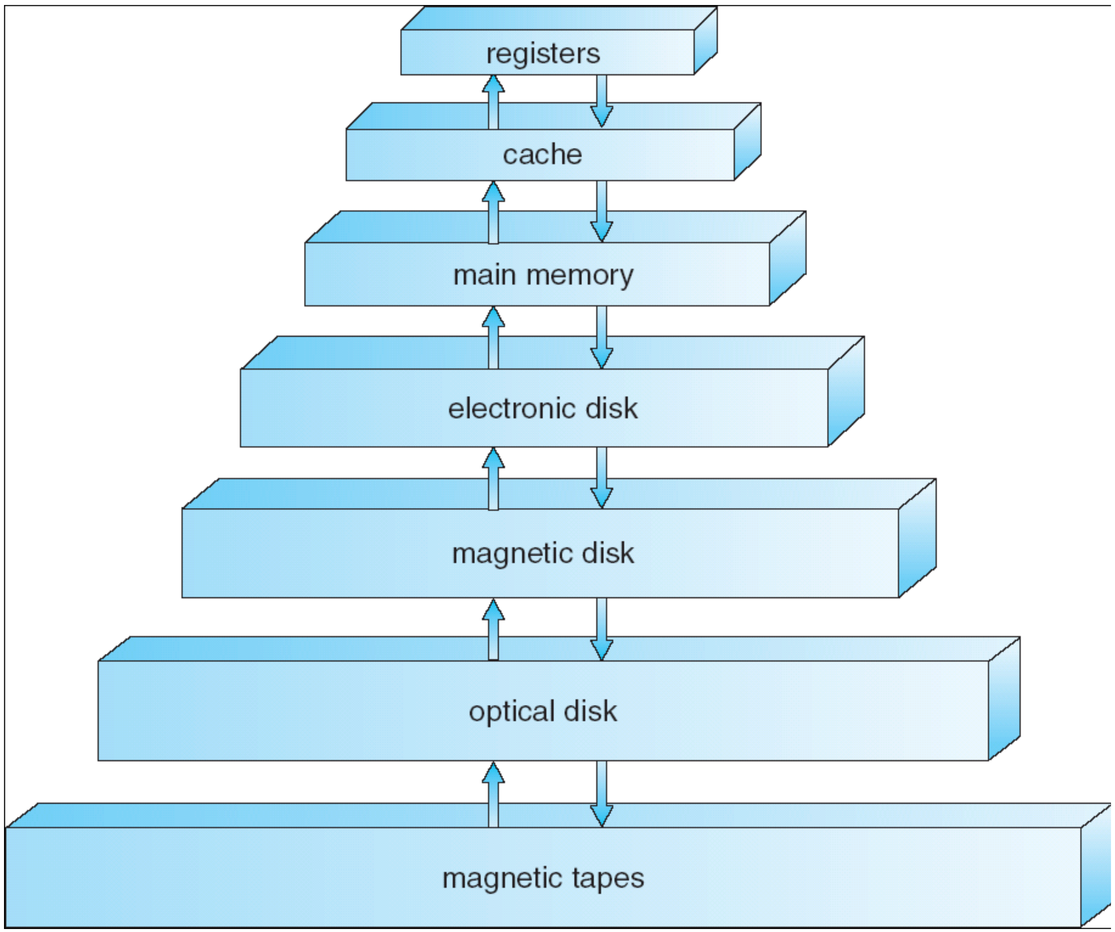


- **Hard disk drives**
- **RAID**
- **Flash SSDs**
- **File and I/O systems**

Storage Management

- OS provides a uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device type (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- Filesystem management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Storage Hierarchy



Storage Structure

- Main memory – relatively large storage media that the CPU can access directly
  - Small CPU cache memories are used to speed up average access time to the main memory at run-time
  - Volatile (data loss at power-off)
  - Byte-addressable

- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
  - Magnetic disks
  - Electronic disks -- Solid state disks (SSDs)
  - Non-volatile (i.e., persistent)
  - Non byte-addressable

Storage Systems Tradeoffs

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
  - Density

- Faster access time, greater cost per bit
- Greater capacity (density), lower cost per bit
- Greater capacity (density), slower access speed

Storage hierarchy – Increased Complexity

2015

L1/L2 cache	~1ns
L3 cache	~10ns
Main memory	~100ns / ~80GB/s / ~100GB
NAND SSD	~100us / ~10GB/s / ~1TB
Fast HDD	~10ms / ~100MB/s / ~10TB

Storage hierarchy – Increased Complexity

2015

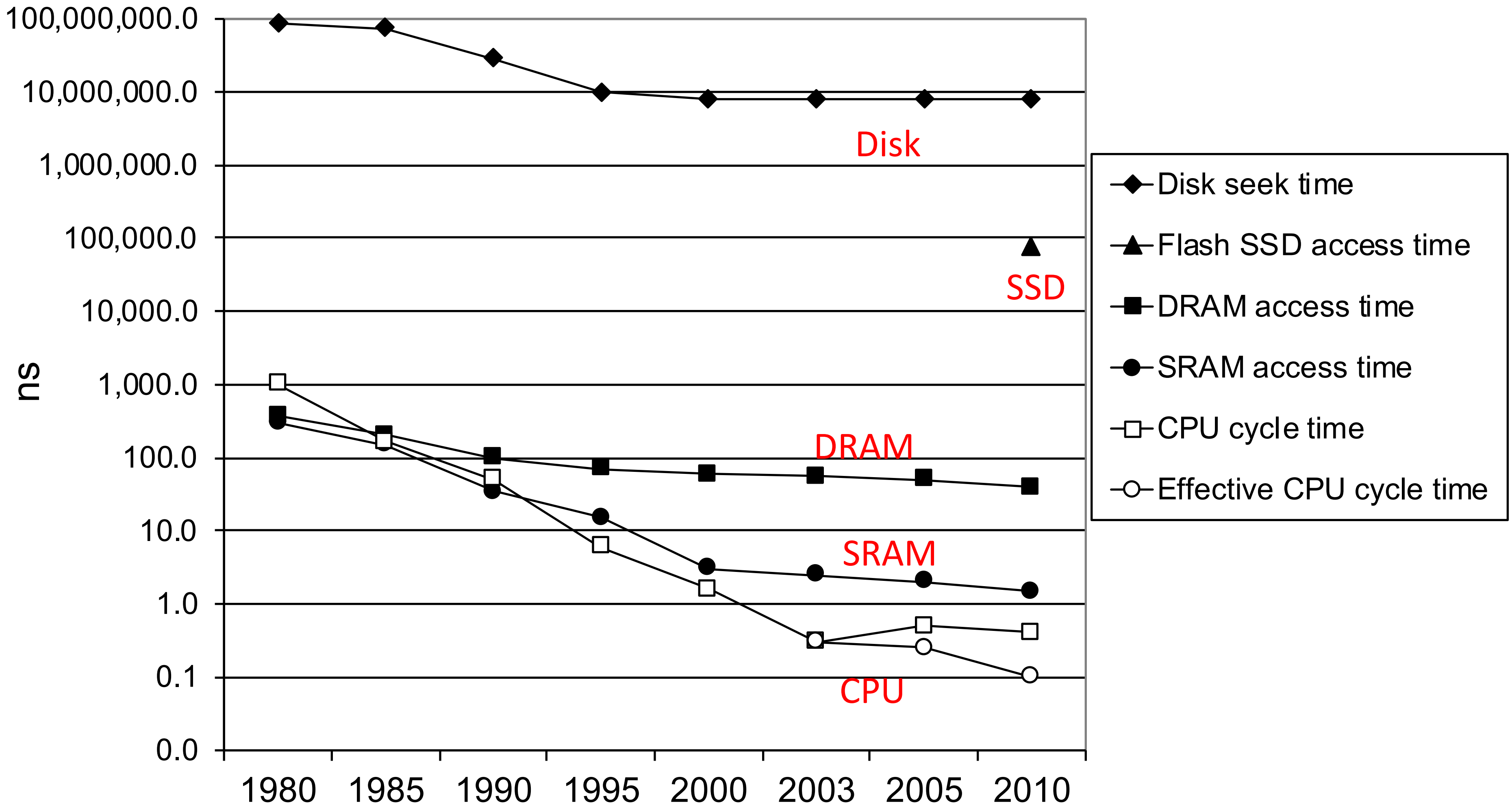
- L1/L2 cache ~1ns
- L3 cache ~10ns
- Main memory ~100ns / ~80GB/s / ~100GB
- NAND SSD ~100us / ~10GB/s / ~1TB
- Fast HDD ~10ms / ~100MB/s / ~10TB

2020

- L1/L2 cache ~1ns
- L3 cache ~10ns
- High-bandwidth memory ~10-100ns / ~1TB/s / ~10GB
- Main memory ~100ns / ~80GB/s / ~100GB
- NVM (3D XPoint) ~1us / ~10GB/s / ~1TB
- NAND SSD ~100us / ~10GB/s / ~10TB
- Fast HDD ~10ms / ~100MB/s / ~100TB

The CPU-Memory Gap

The gap widens between memory, disk, and CPU speeds.

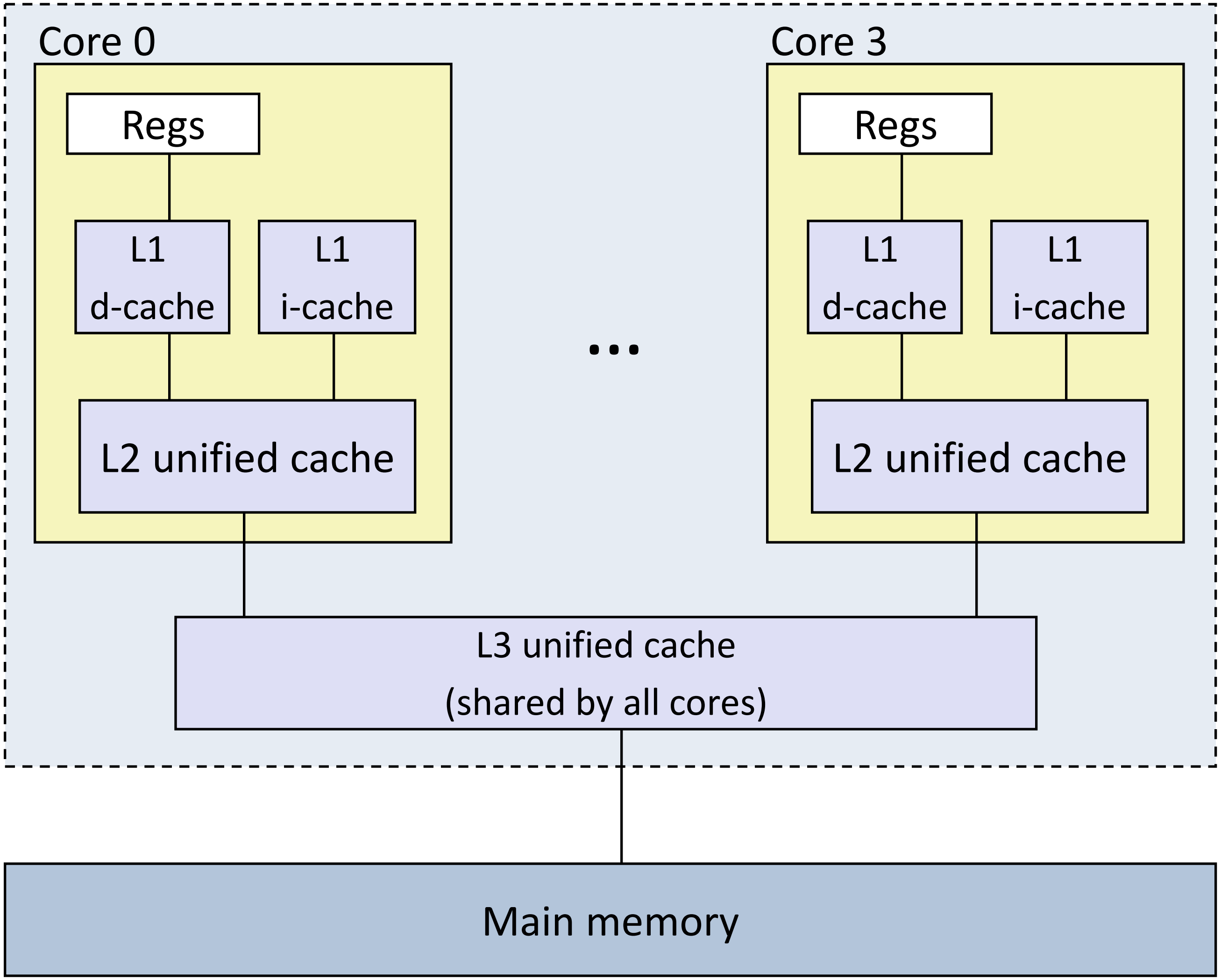


Data decades ago, but trends are the same

Caching

- Skew rule: 80% requests hit on 20% hottest data
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

Intel Core i7 Cache Hierarchy



L1 i-cache and d-cache:

32 KB, 8-way,

Access: 4 cycles

L2 unified cache:

256 KB, 8-way,

Access: 11 cycles

L3 unified cache:

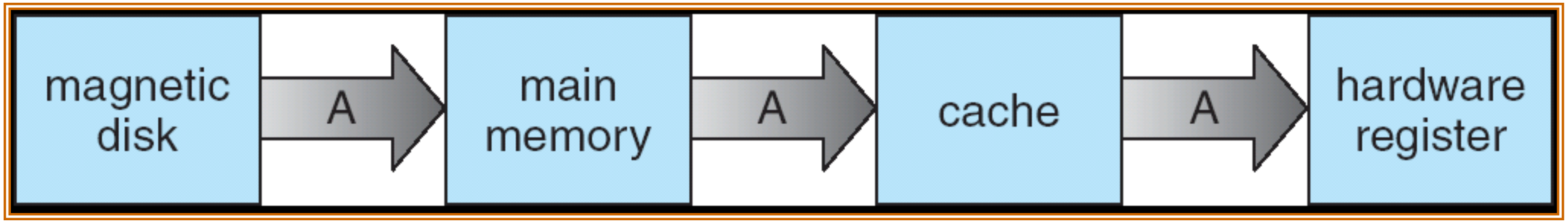
8 MB, 16-way,

Access: 30-40 cycles

Block size: 64 bytes

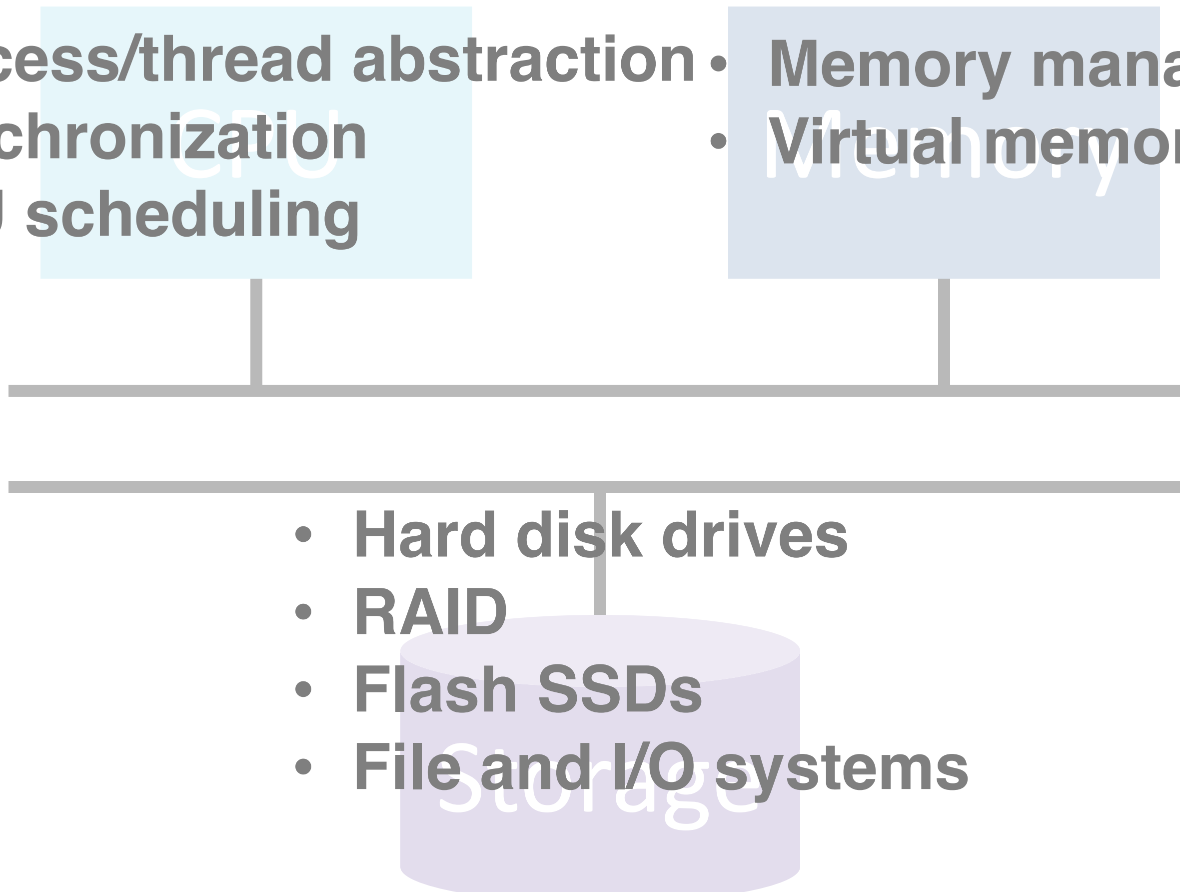
Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a piece of data can exist

Advanced Topics (Miscellaneous)



- Process/thread abstraction
- Synchronization
- CPU scheduling

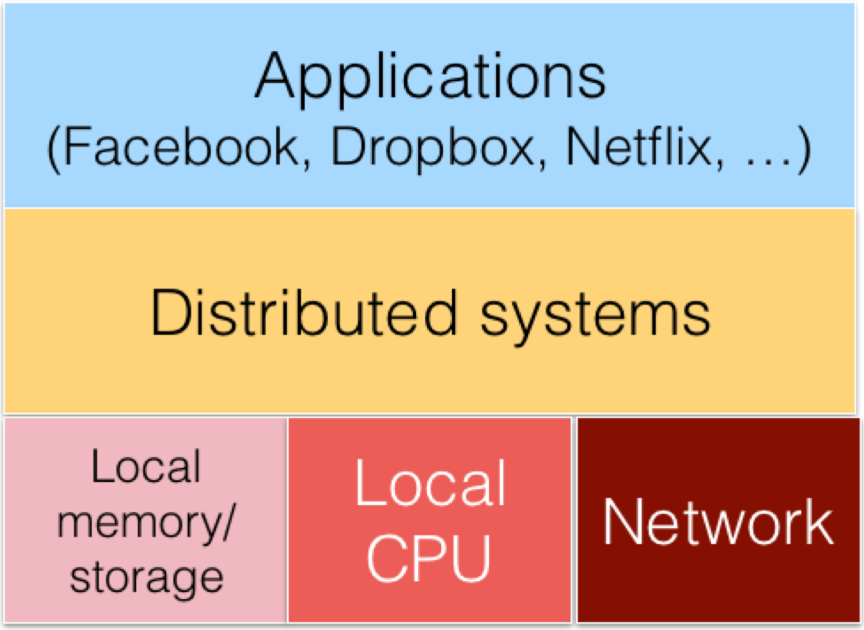
- Memory management
- Virtual memory

- Hard disk drives
- RAID
- Flash SSDs
- File and I/O systems

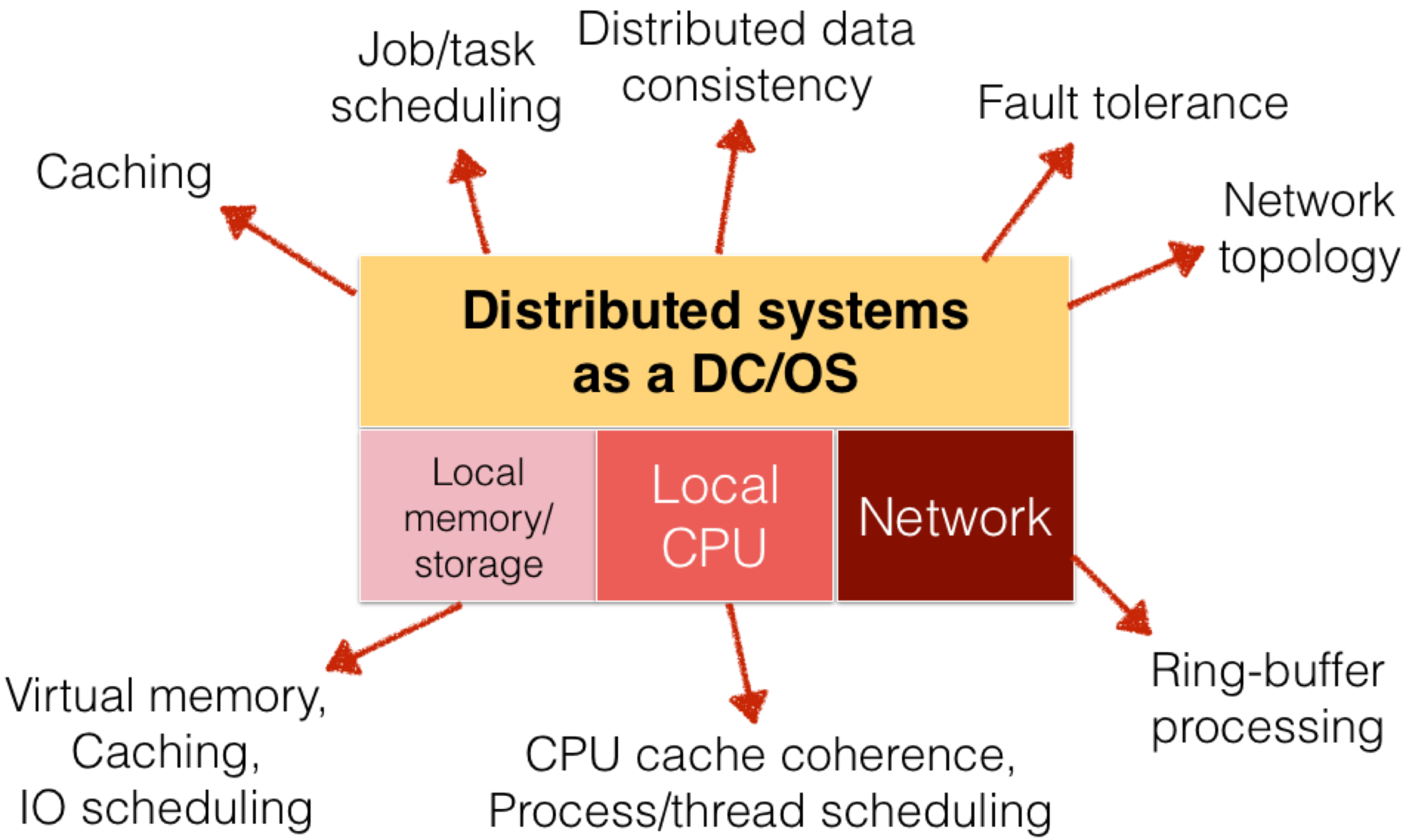
- **Distributed systems**

Distributed Systems as a DC/OS

Distributed Systems as a DC/OS



Distributed Systems as a DC/OS



Why do you take this course?

General Learning Goals

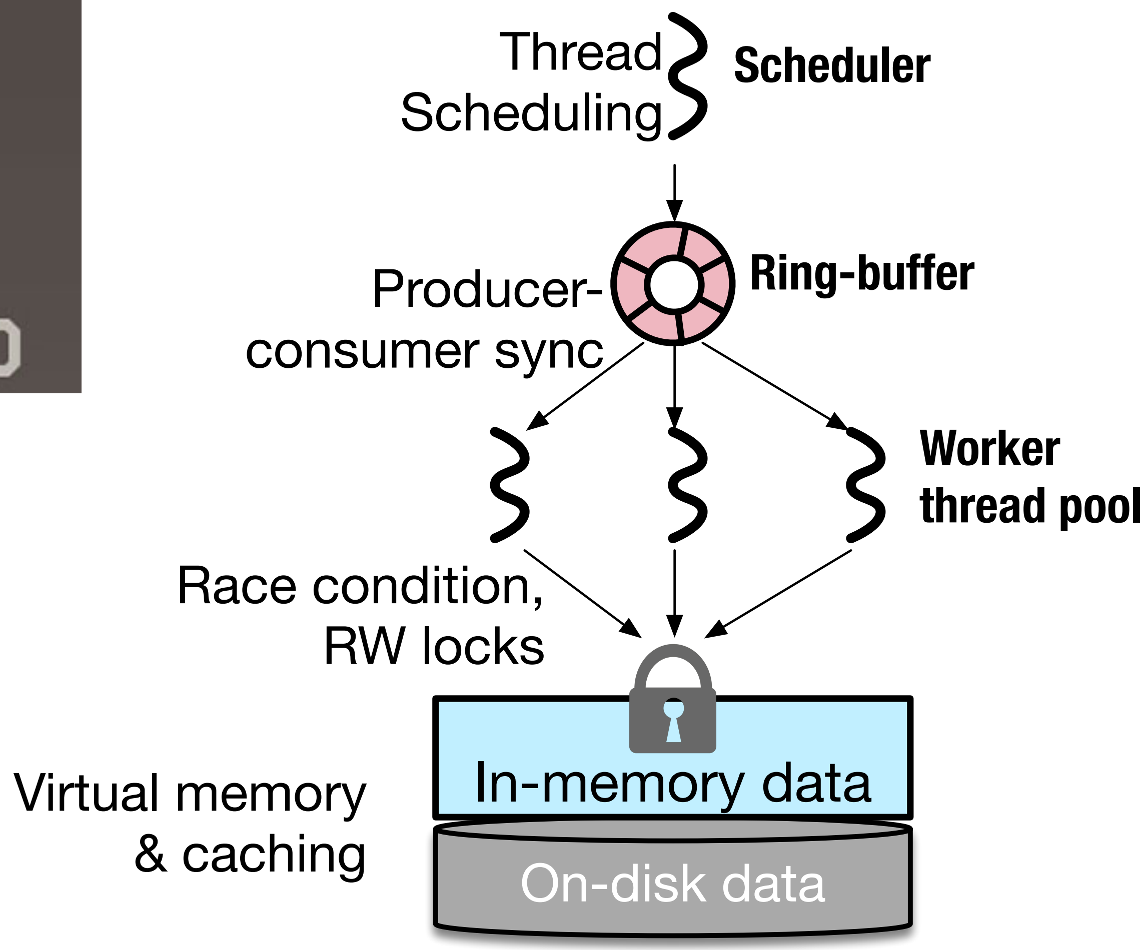
1. Grasp basic knowledge about **Operating Systems** and **Computer Systems** software
2. Learn important systems concepts in general
   - Multi-processing/threading, synchronization
   - Scheduling
   - Caching, memory, storage
   - And more…
3. Gain hands-on experience in ***writing/hacking/designing*** large systems software

Why do you take this course?

- The OS concepts are everywhere
  - Fundamental OS techniques broadly generalize to widely-used systems technique
    - Scheduling
    - Concurrency
    - Memory management
    - Caching
    - …

One example: Memcached

- Memcached is a distributed in-memory object cache system
  - Written in C
  - In-memory hash table
  - Multi-threading



Memcached can be treated as a user-space mini-OS

Next class…

- Process abstraction