CS4410/11: Operating Systems

Storage

Rachit Agarwal
Anne Bracy

Operating Systems — Where are we?

	4410	4411
Homeworks	4 + 1	-
Projects	2 + 1	3 + 1
Exams	-	-

Operating Systems — Where are we?

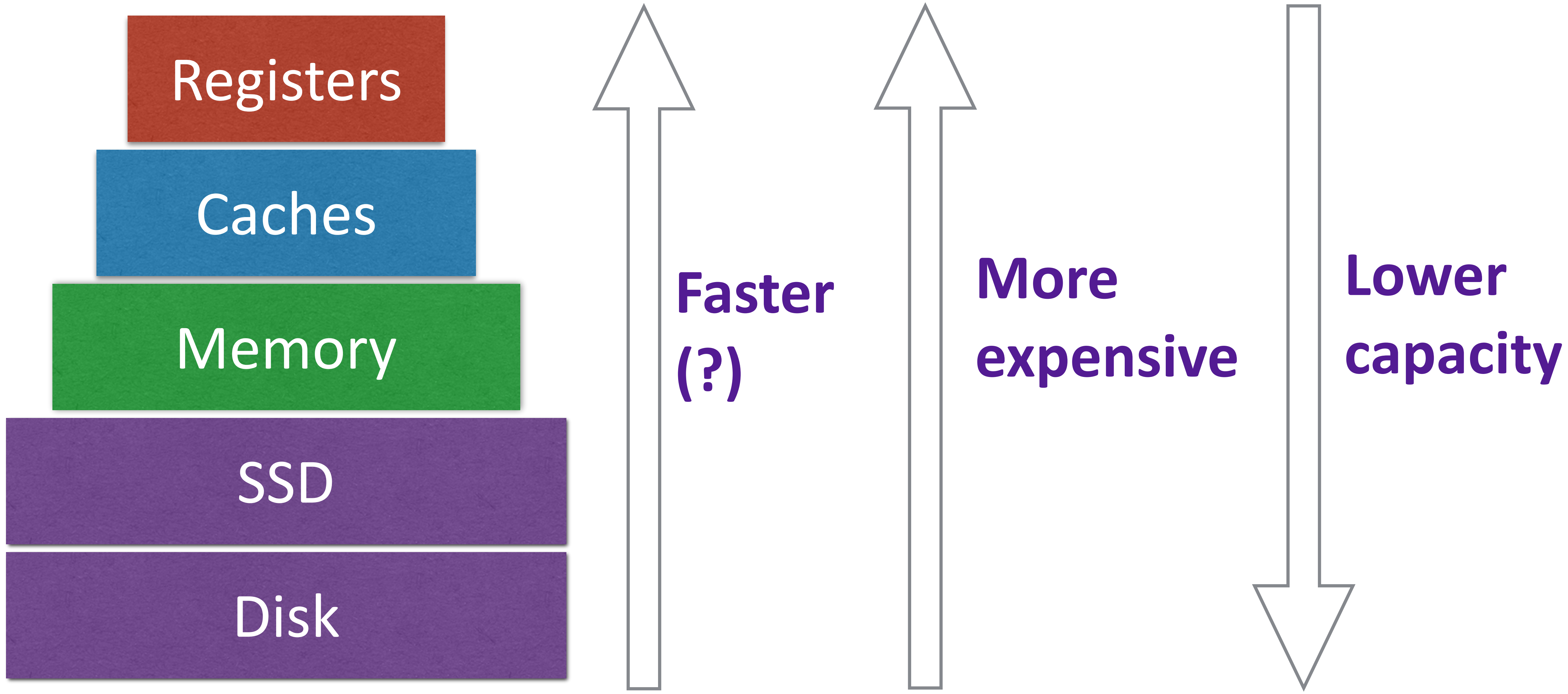
	Max	Mean	Median	Std. Dev.
HW1	20	19.2	20	1.66
HW2	35	26.71	26	5.34
HW3				
HW4				
10-P1	80	73	80	14.46
10-P2				
11-P1	100	83.2	84	8.38
11-P2	100	82.5	84	10.21
11-P3				

Operating Systems — Recap

- **Processes and Threads**
    - Abstraction of a computer (CPU, storage, network, …)
- **Synchronization, Deadlock**
    - Sharing resources “correctly”
- **CPU Scheduling**
    - Sharing CPU resources “efficiently”
- **Networking**
    - Sharing network resources “efficiently”

Operating Systems — Storage (Next 7 lectures)

Sharing Storage “efficiently” and …



Registers

Caches

Memory

SSD

Disk

Faster (?)

More expensive

Lower capacity

Operating Systems — Memory

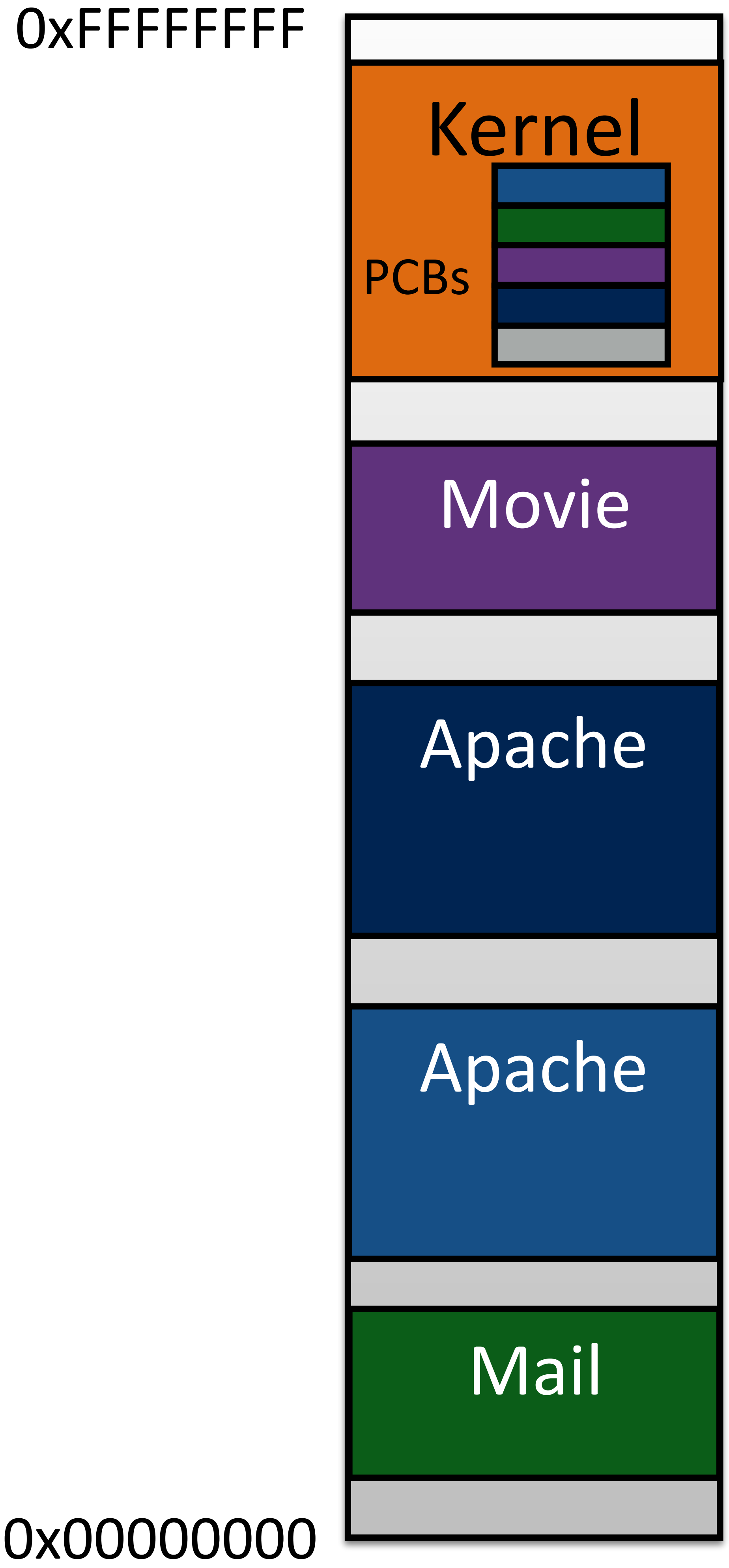
Goal of Memory Management

- **Sharing of memory across processes**
    - Why share memory?
    - Why processes? Why not threads?
- **Time-sharing**
    - Load one program onto machine
    - Execute to completion
    - Problem: Long I/O leads to inefficiencies
- **Space-sharing**
    - Simultaneously running multiple processes

Memory — Sharing

Challenges of space-sharing

- **Protection**
  - Across processes
- **Naming and addressing**
  - Identify physical addresses?
- **Efficiency**
  - Utilization? Using faster memory?



Memory — Sharing

Option 1: Load all

- Load all processes into memory
- Switch between them under OS control
- Must relocate program when load it
- Big Problem: Protection
    - A bug in one processes can kill others
- Guess who used it?
    - MS-DOS, MS-Windows

Memory — Sharing

Option 2: Copy on load

- Copy entire process memory to disk during I/O
- Copy back when it restarts
- No need to relocate
- Big Problem: Performance
  - Why?
- Guess who used it?
- Early versions of Unix

Memory — Sharing

Option 3: Access Check

- Give each program a piece of memory
- Upon each memory reference
    - check that it stays within its address space
- How to implement this?
    - Address translation
    - Base and bound registers
- Cray-1

Memory Sharing — Access Check

Address Translation (more later)

- Program generates virtual addresses
- “Virtual addresses” translated into physical addresses

Memory Sharing — Access Check

Base and Bound registers

- Base: Physical address corresponding to virtual address 0
- Bound: higher allowable virtual address

Memory Sharing — Model

Overall model

- Each process has a virtual address space
- Internally mapped to physical address space
    - Virtual to Physical allocation?

Memory Sharing — Model

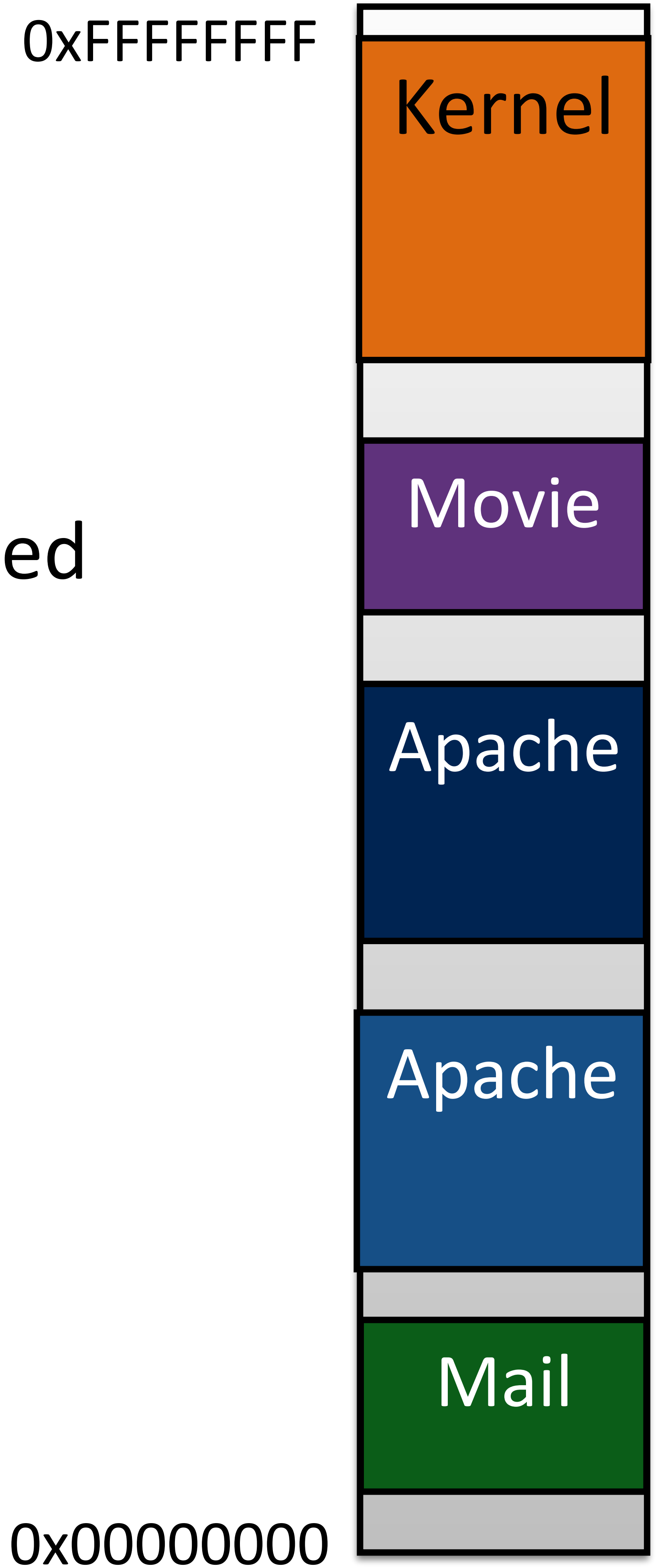
Virtual to Physical allocation

- **First-Fit**
  - Allocate first “hole” that is big enough
- **Best-Fit**
  - Allocate smallest “hole” that is big enough
- **Worst-Fit**
  - Allocate largest “hole” that is big enough

Memory Sharing — Model

Virtual to Physical allocation: Problem?

- **External Fragmentation**
- Available physical memory, but fragmented
- **Various options**
  - Wait for space (problem?)
  - Make space (how?)



Memory Sharing — Model

Virtual to Physical allocation: Solution

- Allocations at “finer granularity”
    - Pages
- Break physical address space into fixed size pages
- Map Virtual address space to multiple pages
    - Non-contiguous
- Dynamic address translation