Distributed Systems Design
COMP 6231

Processes

Lecture 3 – Part 1

Essam Mansour

Today…

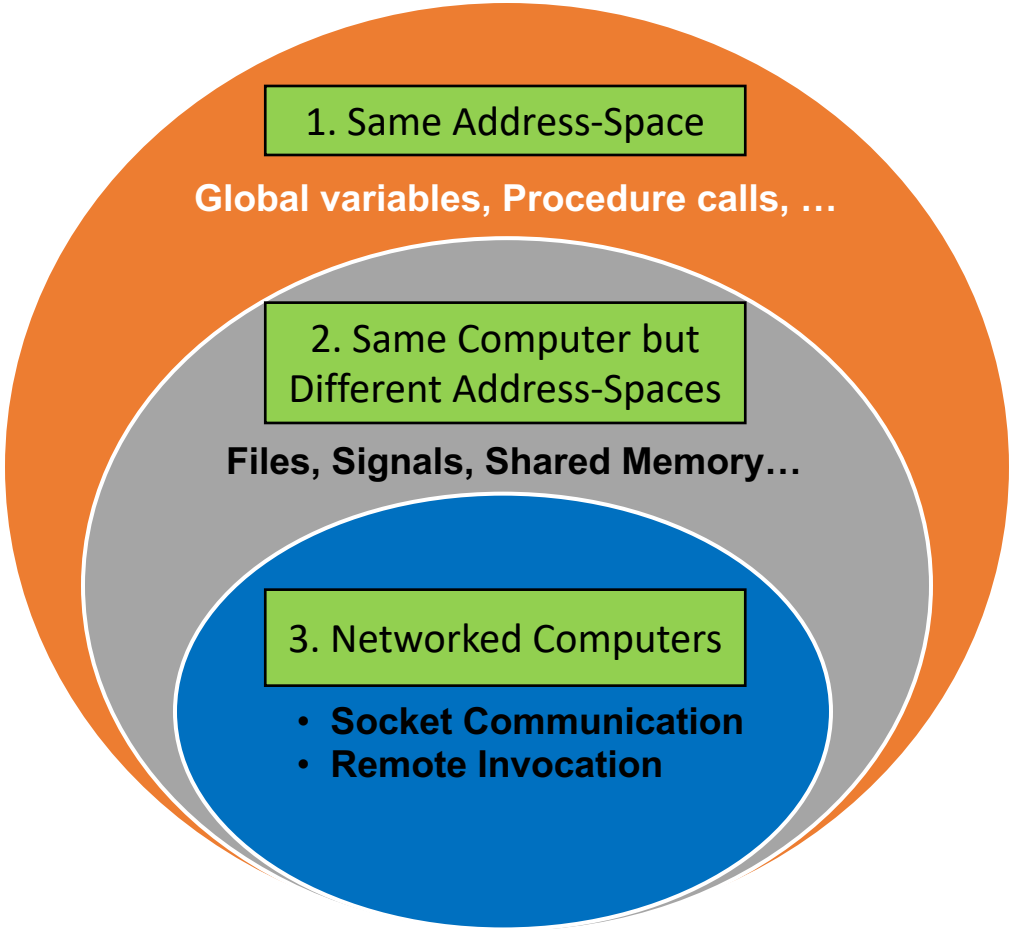
- Last Session:
  - Architectures for distributed systems
- Today's Session:
  - Processes
  - Threads
  - Virtualization

Communicating Entities in Distributed Systems

- Communicating entities in distributed systems can be classified into two types:
  - System-oriented entities
    - Processes
    - Threads
    - Nodes
  - Problem-oriented entities
    - Objects (in *object-oriented programming* based approaches)

Classification of Communication Paradigms

- Communication paradigms can be categorized into *three* types based on where the entities reside. If entities are running on:

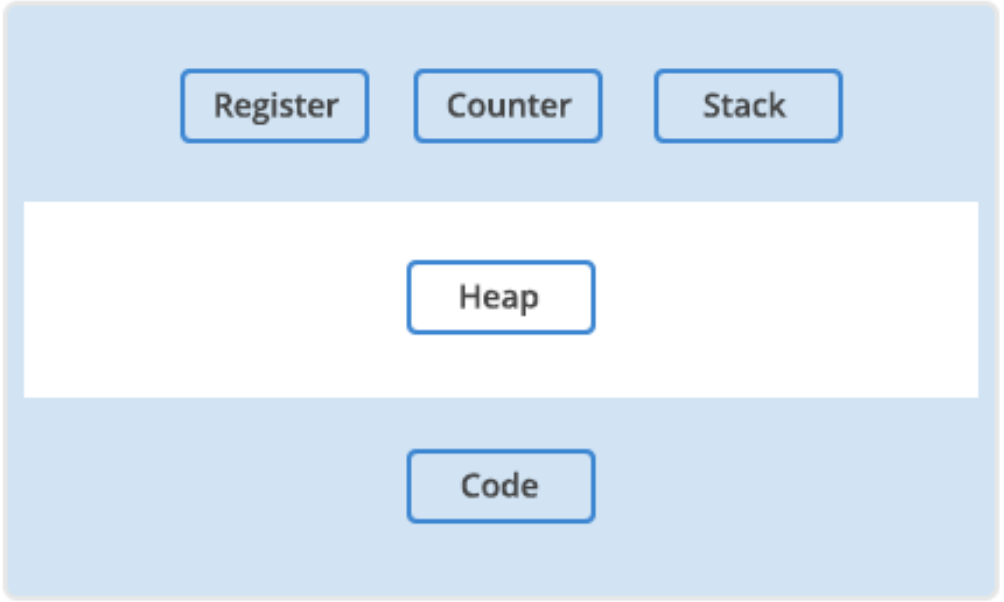


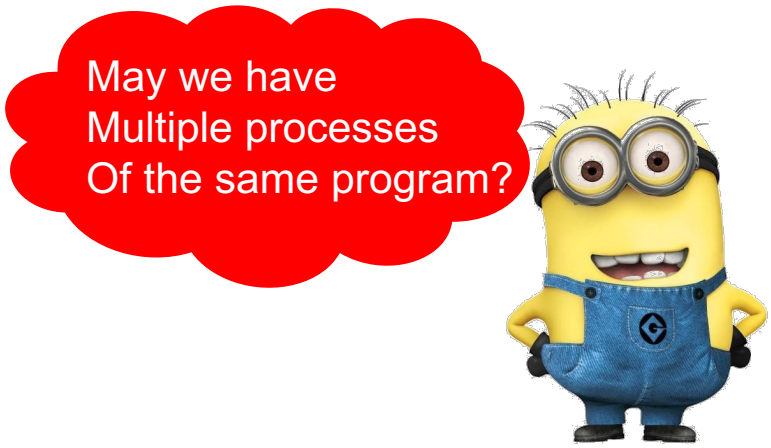
Processes vs Threads

- A program to be executed needs more than just binary code.
- It needs:
  - Memory, and
  - operating system resources
    - a register may hold an instruction, or storage address,
    - program counter keeps track of where a computer is in its program sequence,
    - “stack” is a data structure that stores information about the active subroutines of a computer program

Processes vs Threads

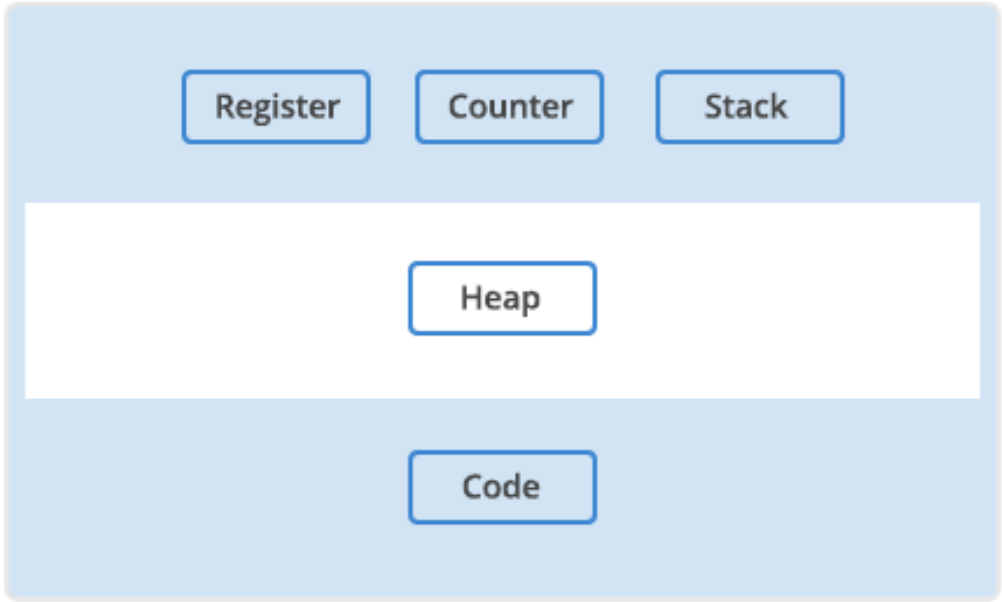
- A computer process





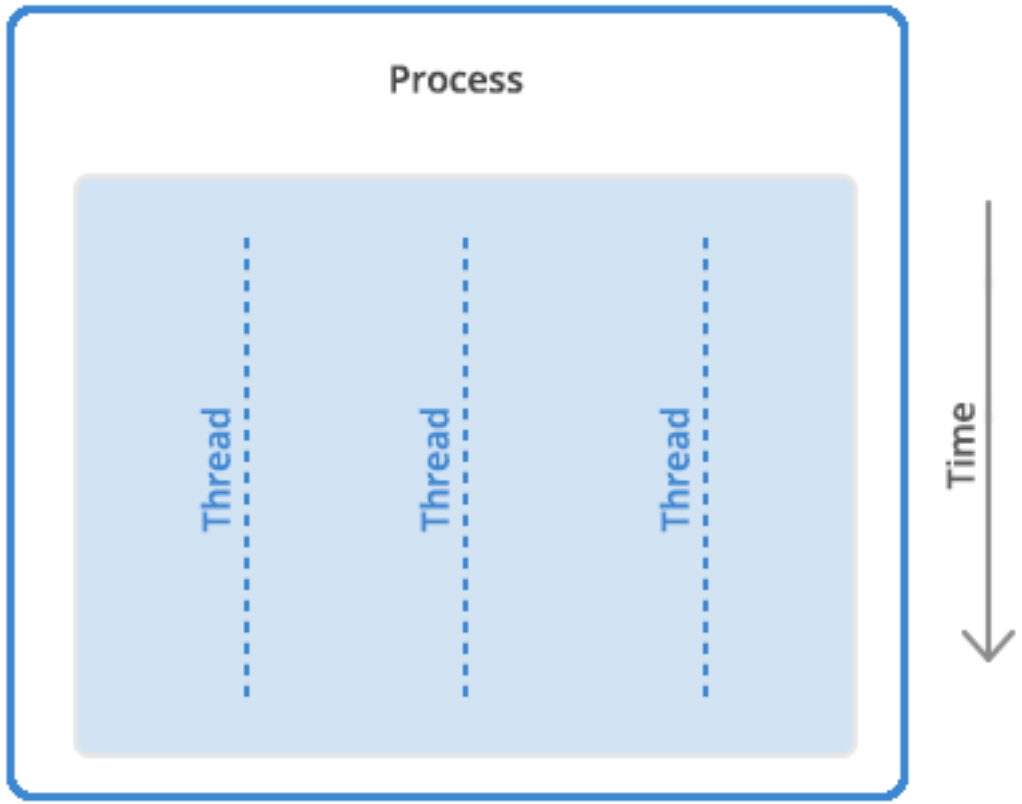
Processes vs Threads

- A computer process
- Yes, but each process has a separate memory address space, i.e., a process runs independently and is isolated from other processes.



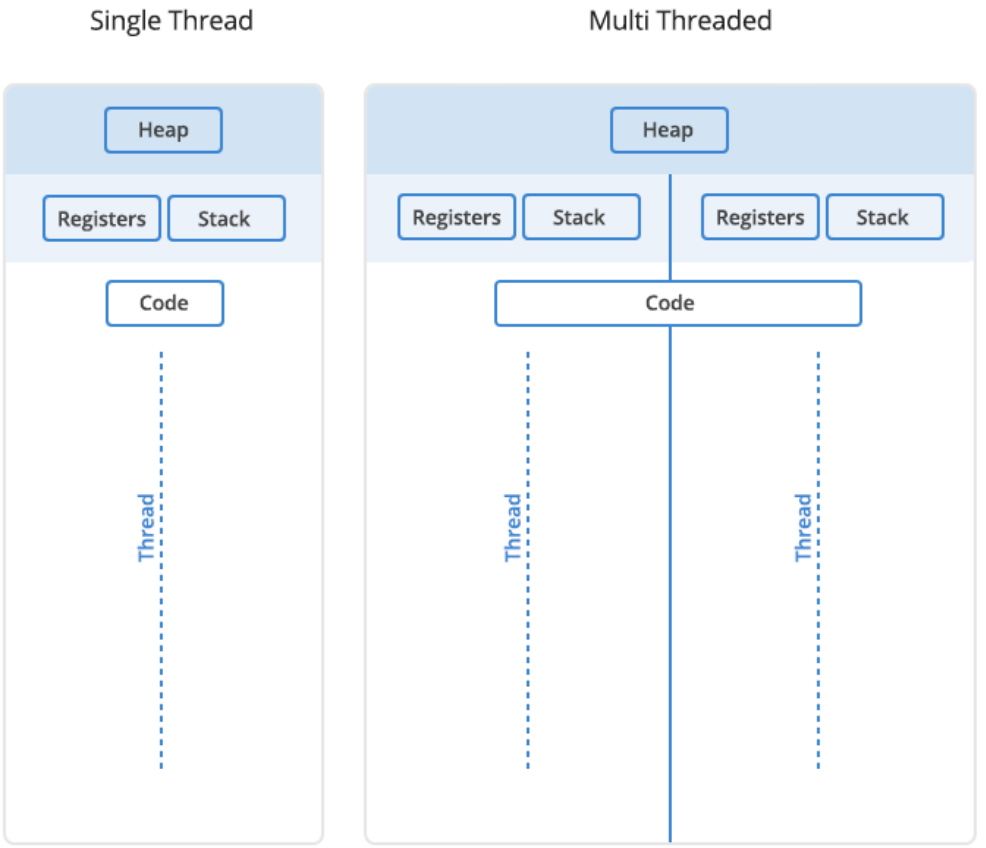
Processes vs Threads

- A thread could be defined as the unit of execution within a process.



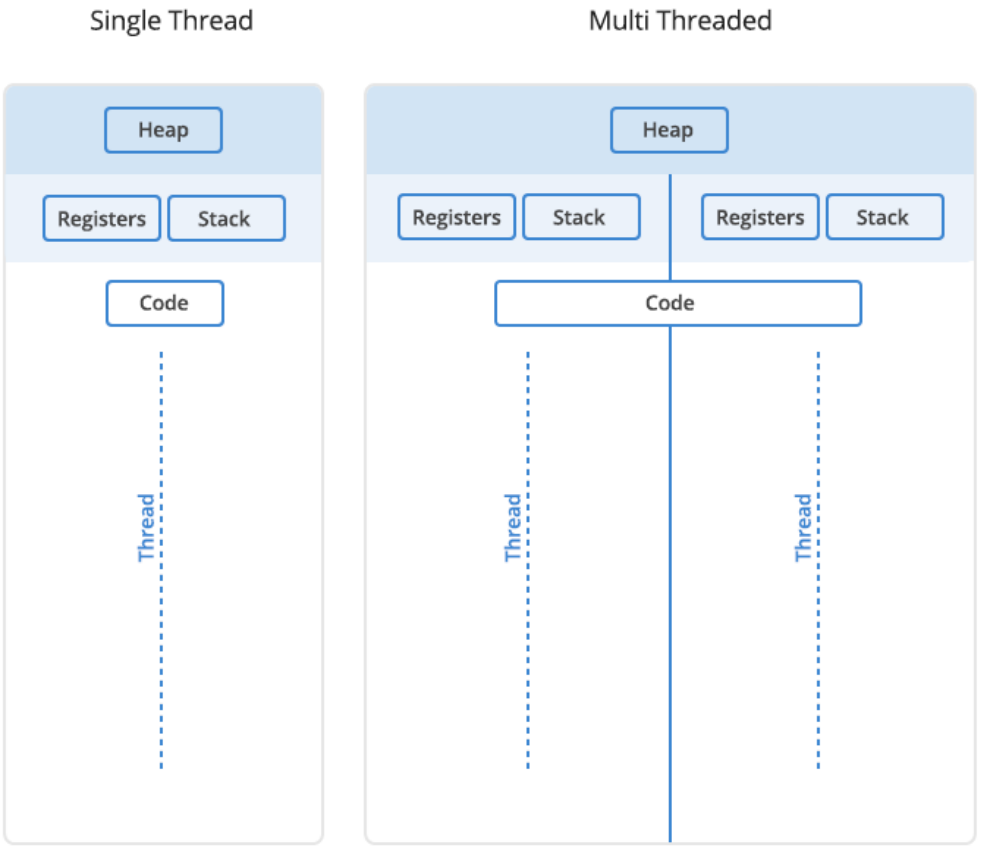
Processes vs Threads

- A thread could be defined as the unit of execution within a process.



Processes vs Threads

- A thread could be defined as the unit of execution within a process.



Threads could be seen as lightweight processes as they have their own stack but can access shared data.

Processes vs Threads

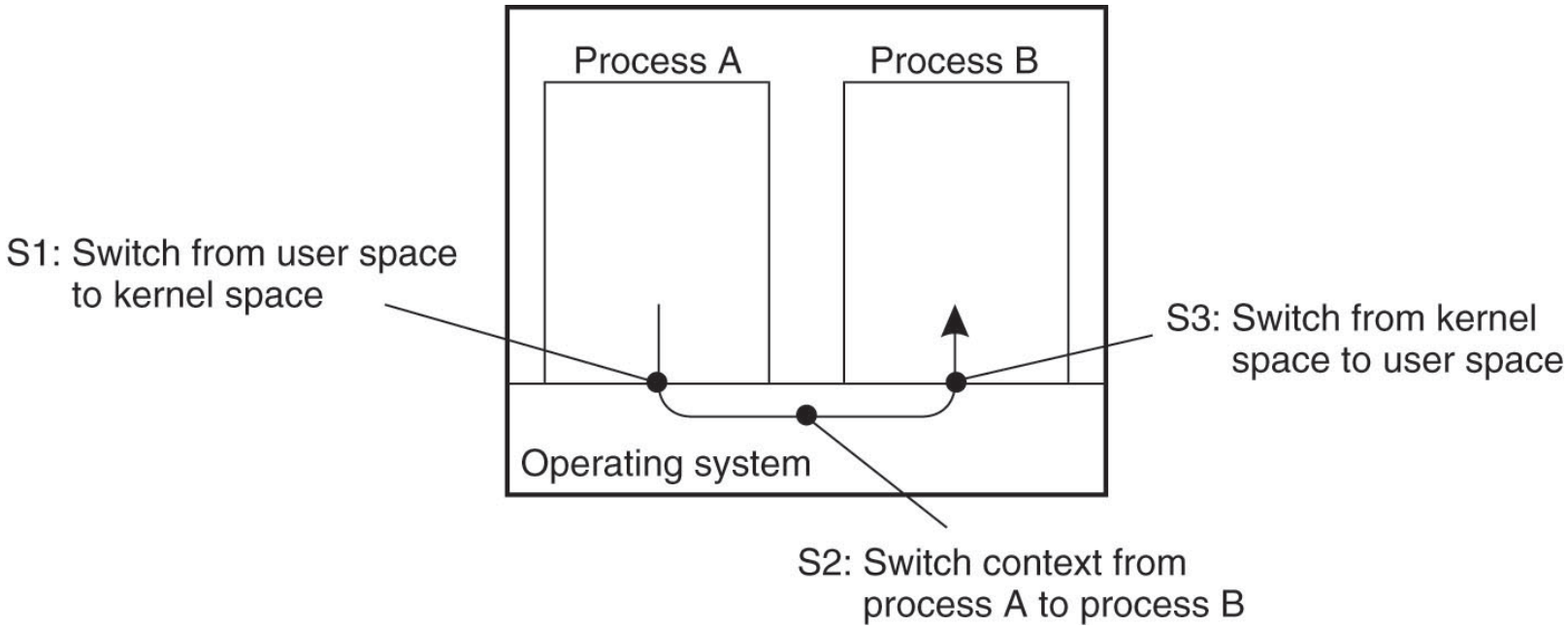


Figure 3-1. **Context switching** as the result of interprocess communication

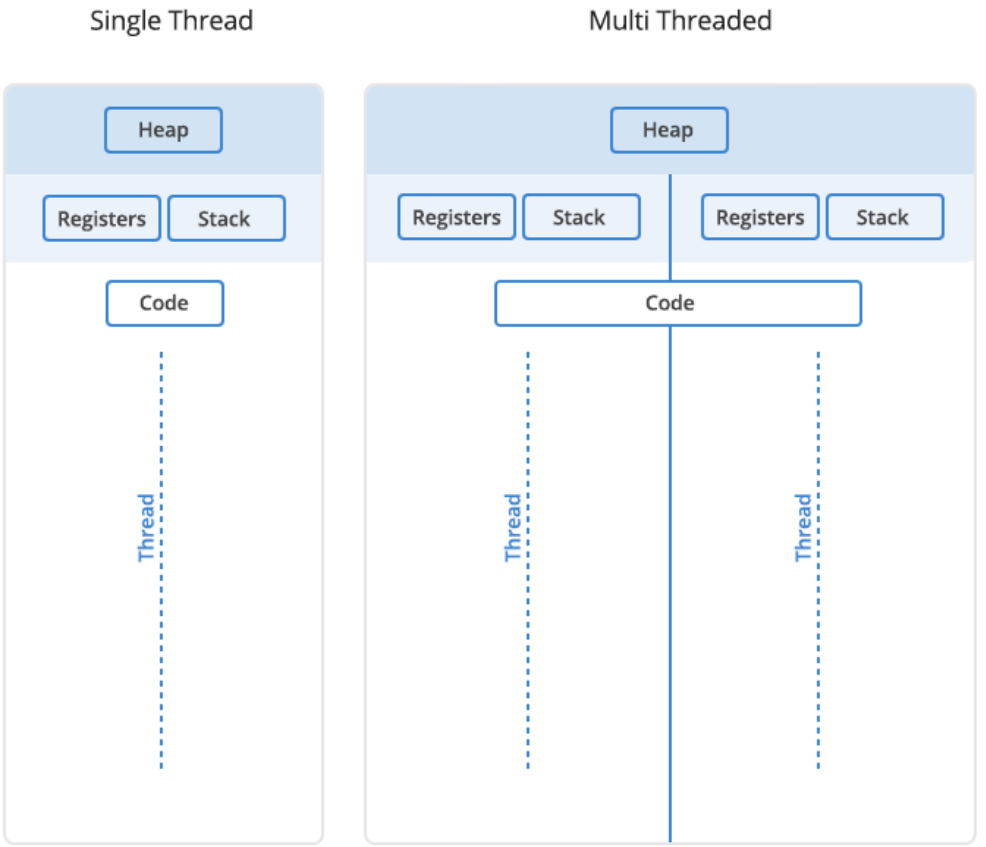
Processes vs Threads

Why use threads?

- Avoid needless blocking:
  - a single-threaded process will block when doing I/O; in a multi-threaded process, the operating system can switch the CPU to another thread in that process.
- Exploit parallelism:
  - the threads in a multi-threaded process can be scheduled to run in parallel on a multiprocessor or multicore processor.
- Avoid process switching:
  - structure large applications not as a collection of processes, but through multiple threads.

Processes vs Threads

- A thread could be defined as the unit of execution within a process.



Threads could be seen as lightweight processes as they have their own stack but can access shared data.

Processes vs Threads

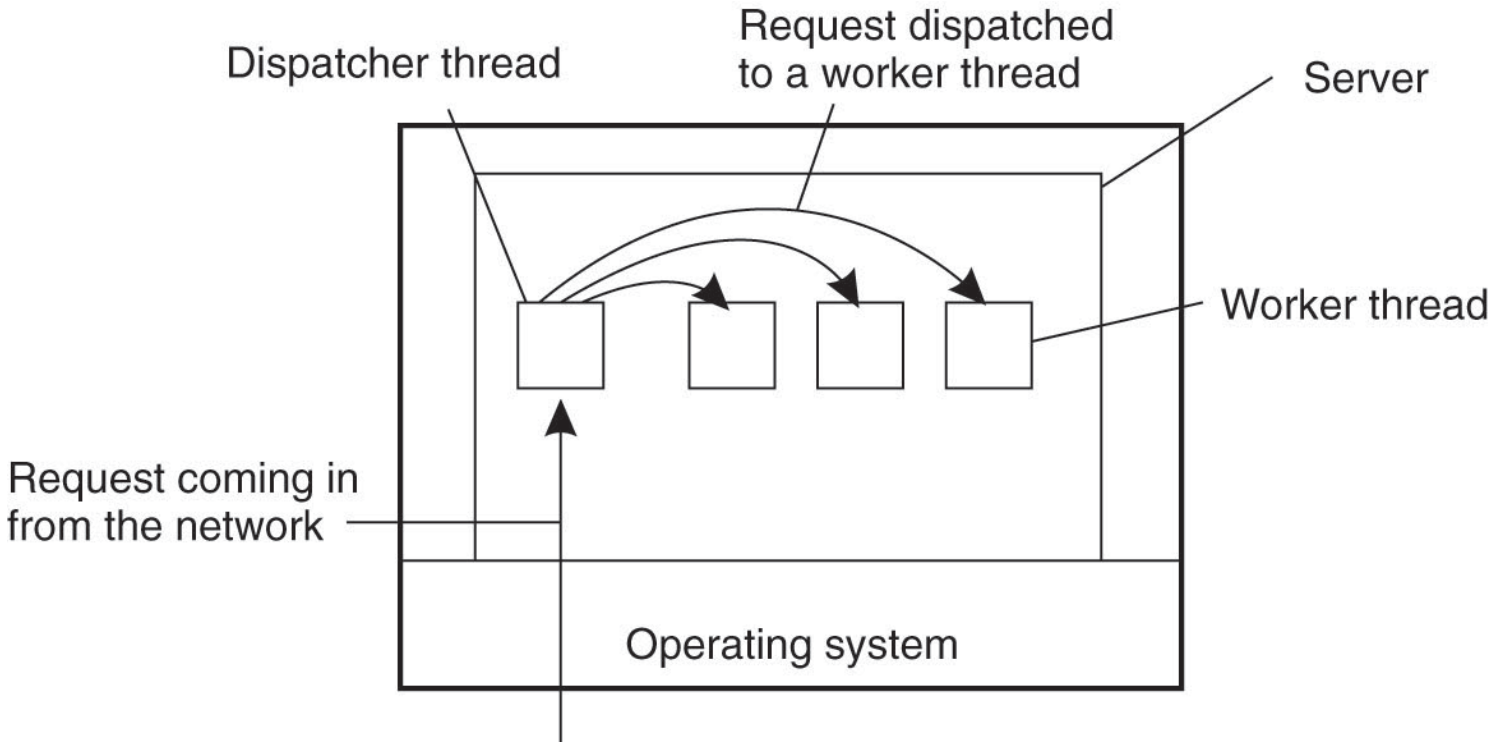


Figure 3-3. A multithreaded server organized in a dispatcher/worker model.

Processes vs Threads

Design Choose: Process over Thread or Thread over Process?

Use-cases:

- The Chrome browser
- Spreadsheeted
- A web client

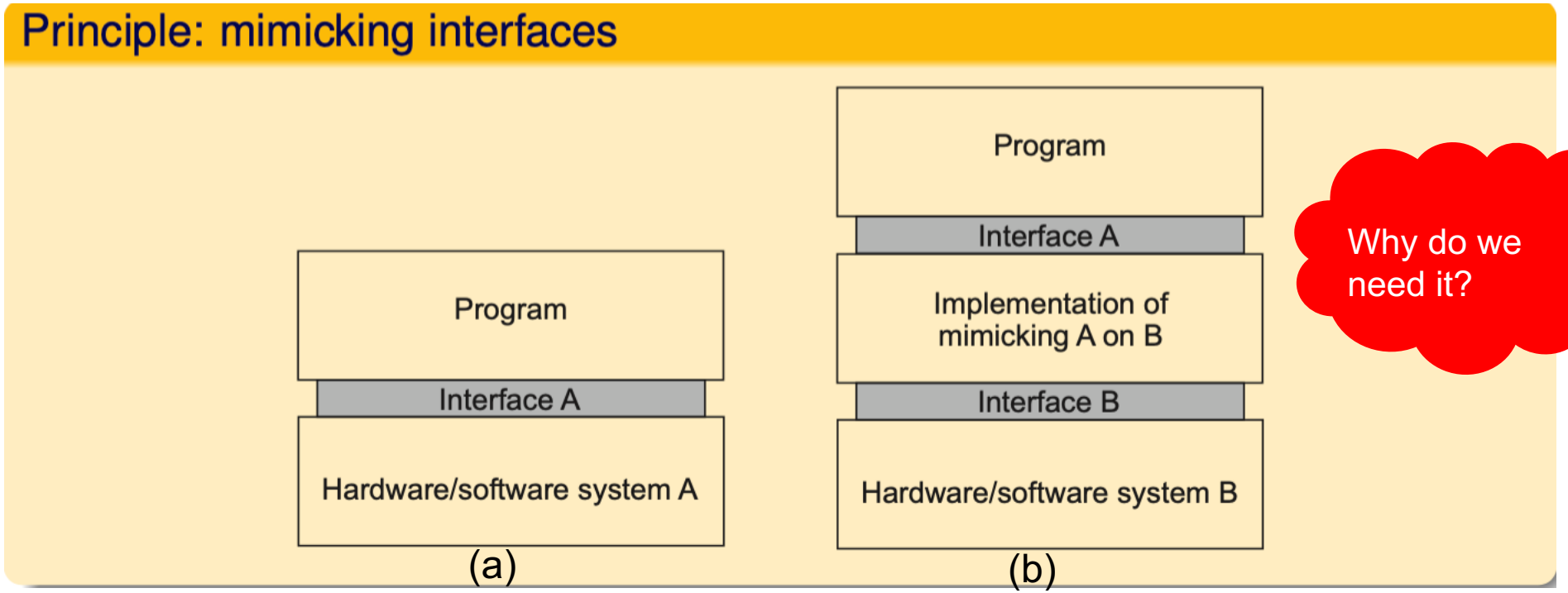
Processes vs Threads

Design Choose: Process over Thread or Thread over Process?

Use-cases:

- The Chrome browser (multiple processes)
- Spreadsheeted (multiple threads)
- A web client (multiple threads)

Virtualization

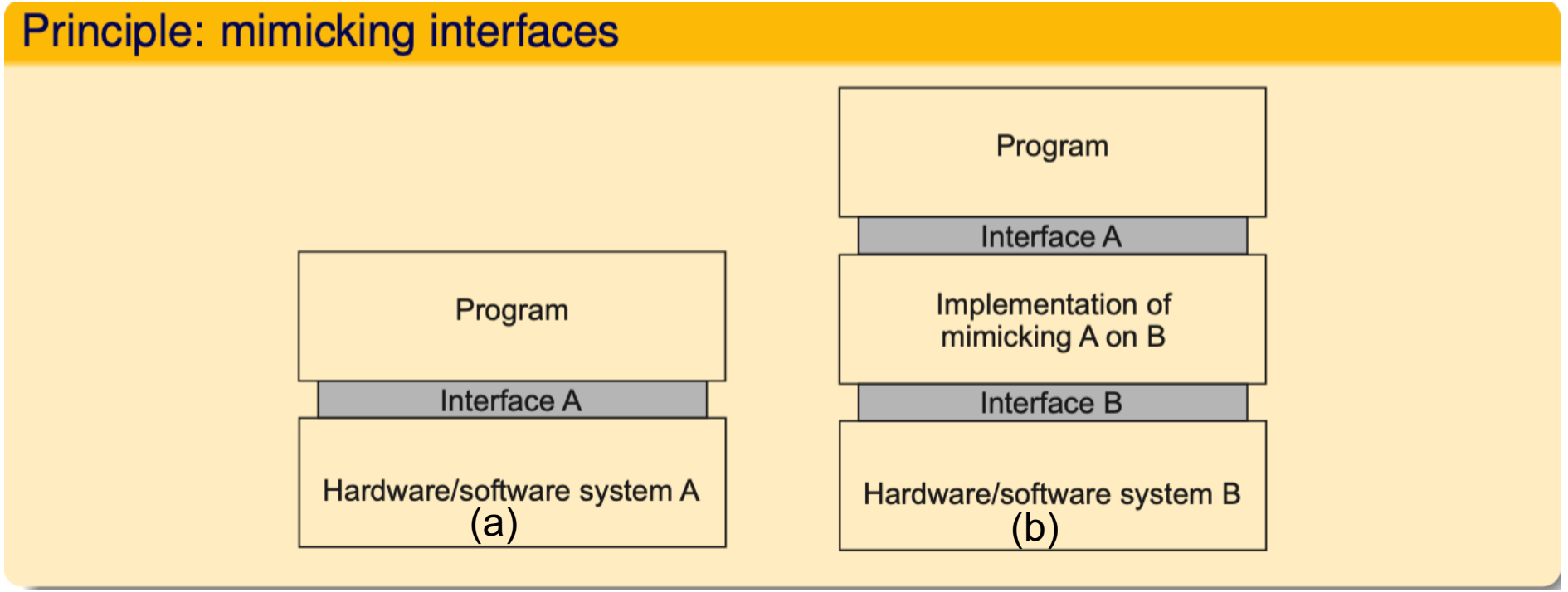


(a) General organization between a program, interface, and system.

(b) General organization of virtualizing system A on top of system B.

Virtualization

- Virtualization is important:
  - Hardware changes faster than software
  - Ease of portability and code migration,
  - Isolation of failing or attacked components



(a) General organization between a program, interface, and system.

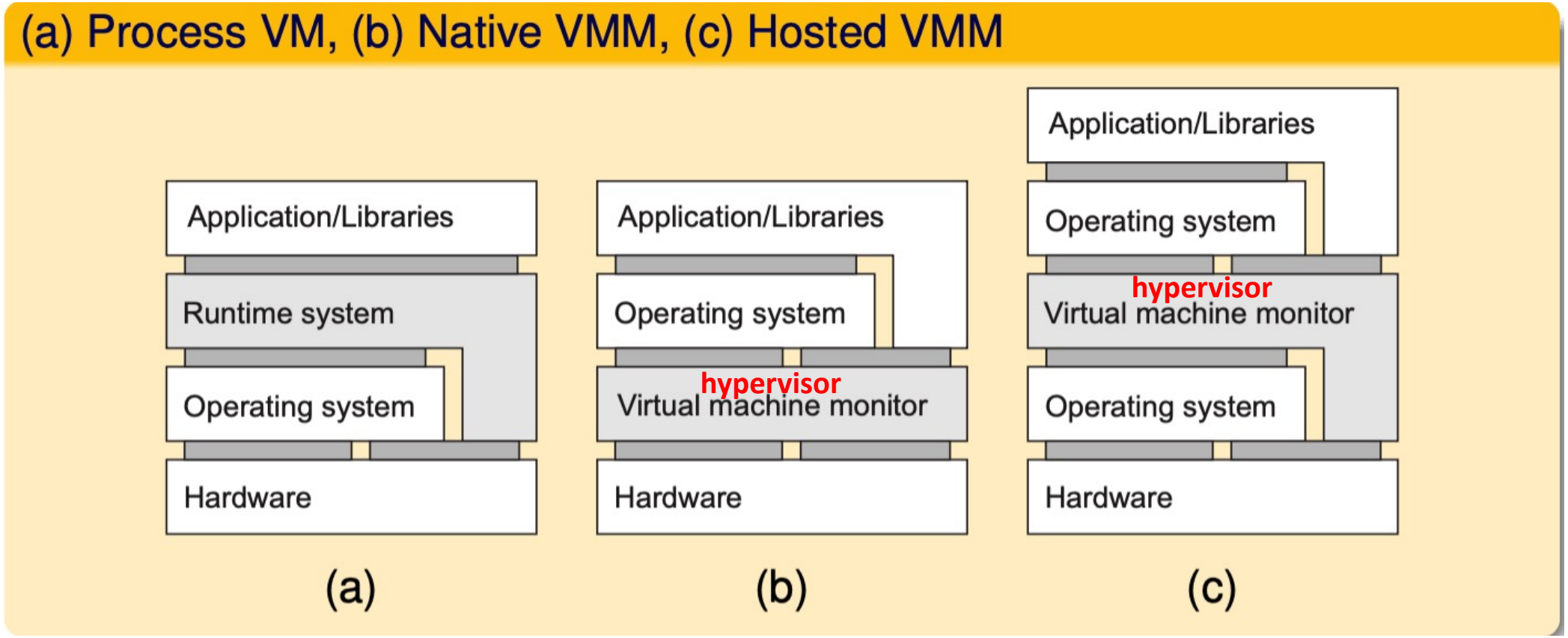
(b) General organization of virtualizing system A on top of system B.

Virtualization

Mimicking interfaces

- Four types of interfaces at three different levels
  - **Instruction set architecture**: the set of *machine instructions*, with two subsets:
    - Privileged instructions: allowed to be executed only by the operating system.
    - General instructions: can be executed by any program.
- System calls as offered by an operating system.
- Library calls, known as an application programming interface (API)

Ways of virtualization

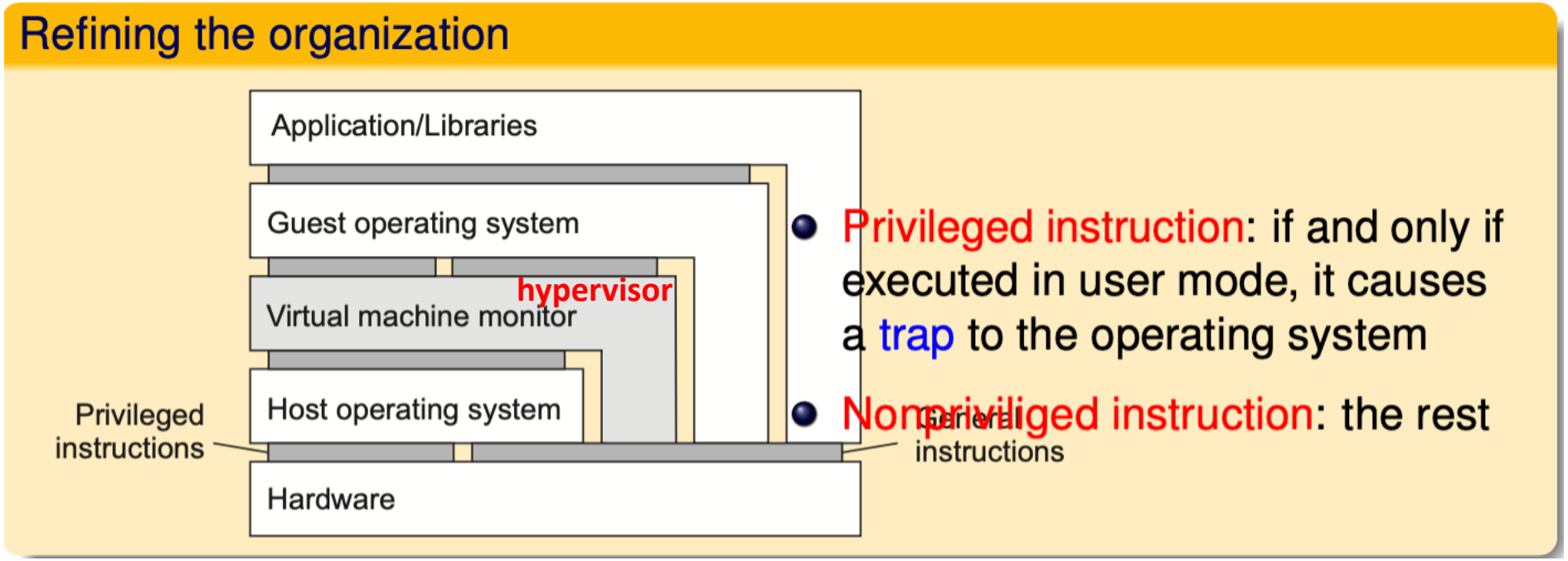


(a) Separate set of instructions, an interpreter/emulator, running atop an OS.

(b) Low-level instructions, along with bare-bones minimal operating system

(c) Low-level instructions, but delegating most work to a full-fledged OS.

Zooming into VMs: performance

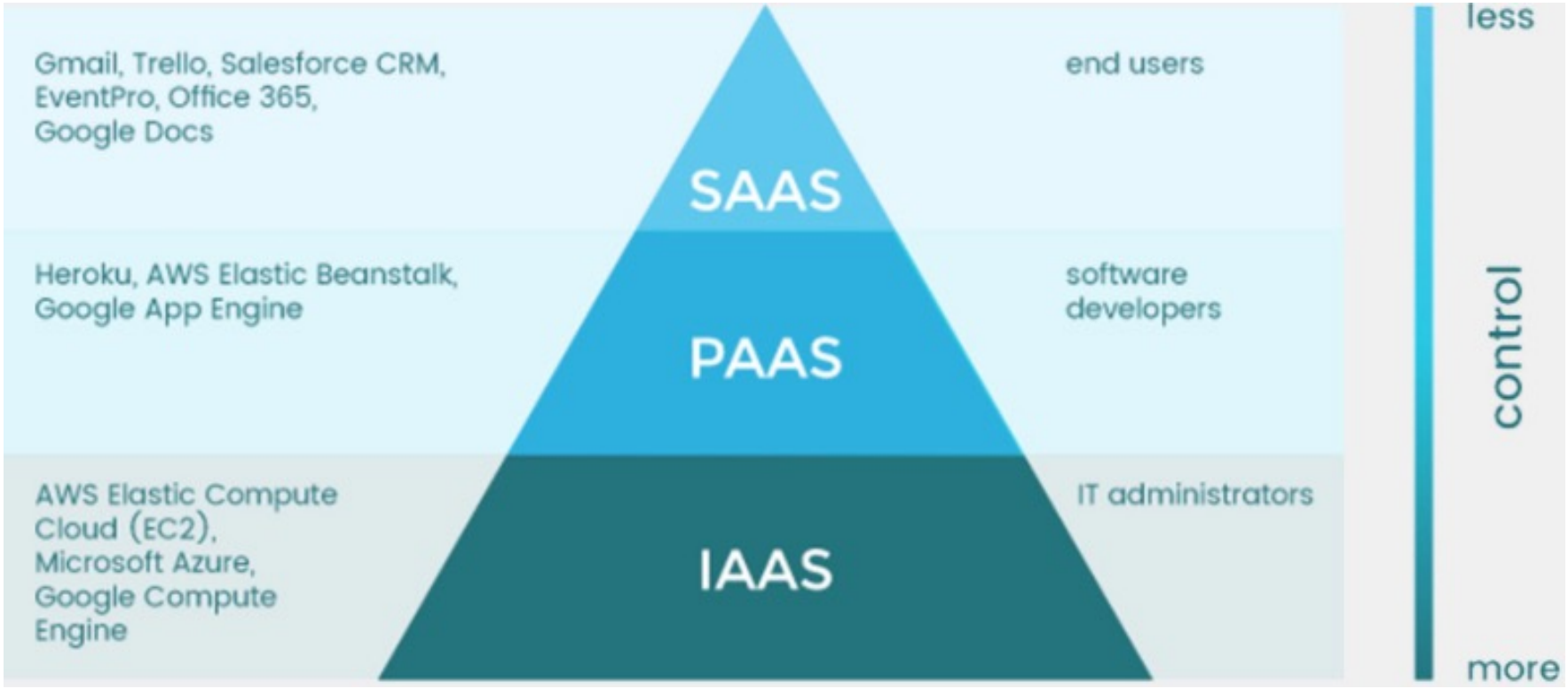


- Control-sensitive instruction: may affect configuration of a machine (e.g., one affecting relocation register or interrupt table).
- Behavior-sensitive instruction: effect is partially determined by context (e.g., POPF sets an interrupt-enabled flag, but only in system mode).

VMs and cloud computing

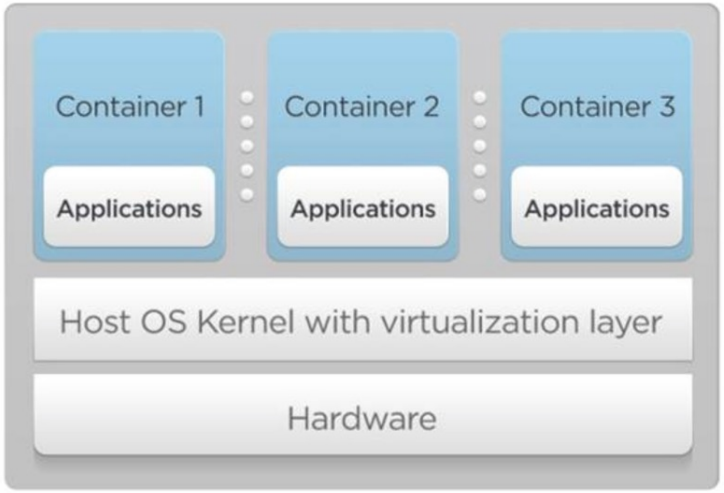
Three types of cloud services

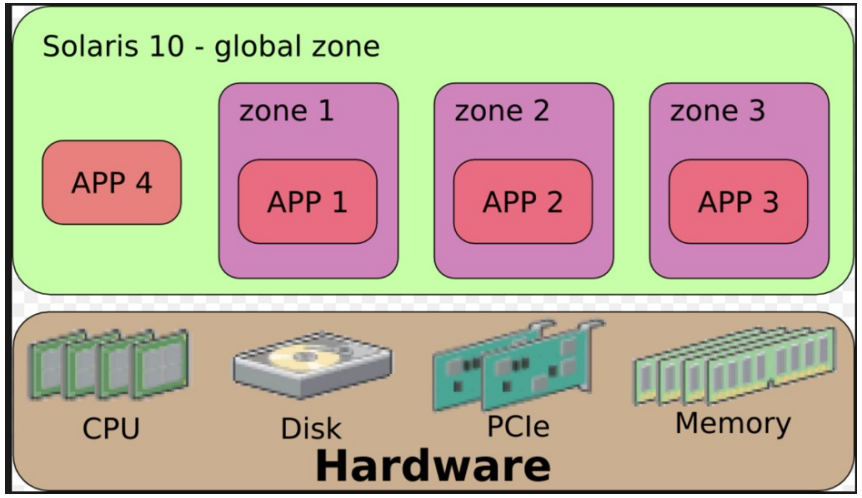
- Infrastructure-as-a-Service covering the basic infrastructure
- Platform-as-a-Service covering system-level services
- Software-as-a-Service containing actual applications



containers

- Emulate OS-level interface with native interface
- “Lightweight” virtual machines
  - No hypervisor, OS provides necessary support



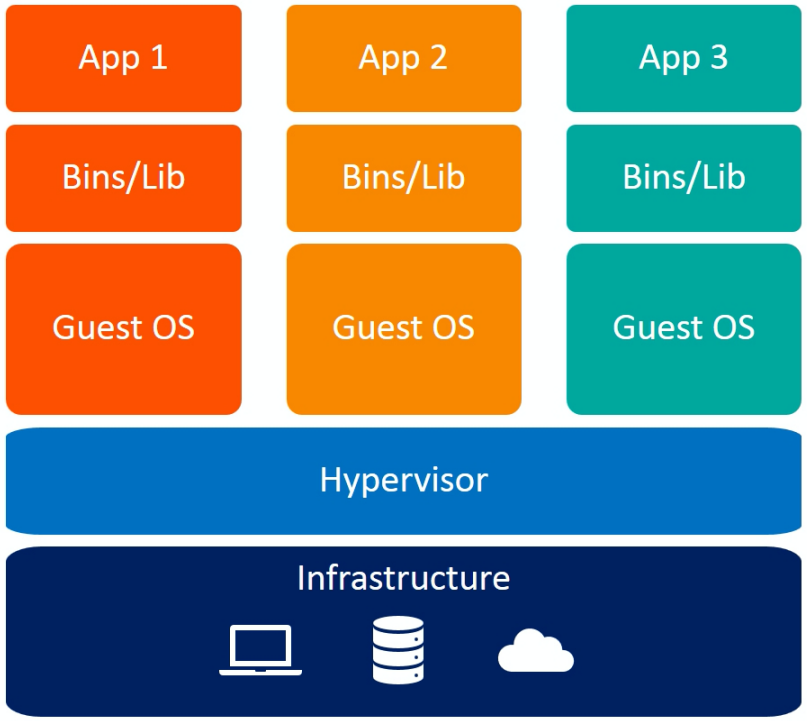


- Referred to as *containers*
  - Solaris containers, BSD jails, Linux containers

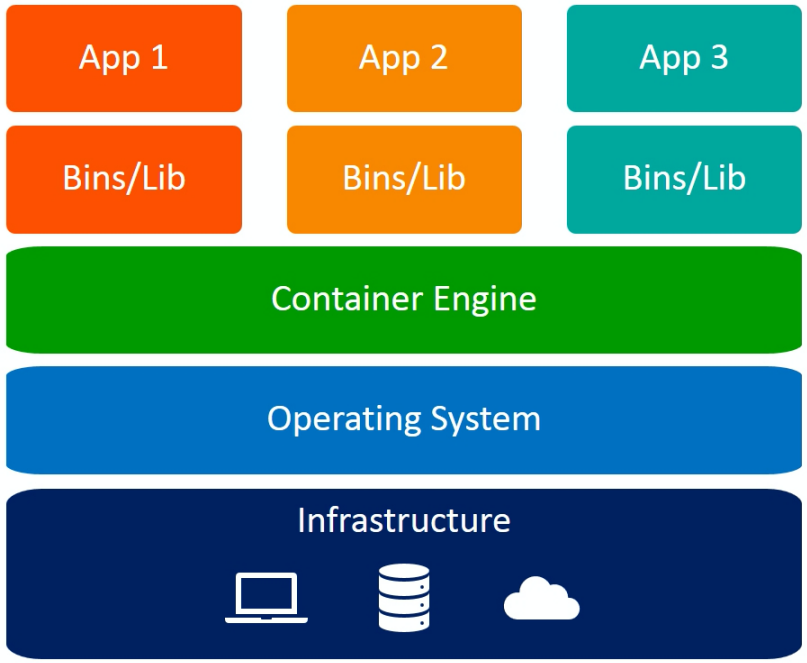
Docker and Linux Containers

- Linux containers are a set of kernel features
  - Need user space tools to manage containers
  - Virtuozo, OpenVZm, VServer,Lxc-tools, Docker
- What does Docker add to Linux containers?
  - Portable container deployment across machines
  - Application-centric: geared for app deployment
  - Automatic builds: create containers from build files
  - Component re-use
- Docker containers are self-contained: no dependencies

VMs vs Containers



Virtual Machines



Containers

- Containers share OS kernel of the host
  - OS provides resource isolation
- Benefits
  - Fast provisioning, bare-metal like performance, lightweight

VMs vs Containers

VMs	Containers
Heavyweight	Lightweight
Limited performance	Native performance
Each VM runs in its own OS	All containers share the host OS
Hardware-level virtualization	OS virtualization
Startup time in minutes	Startup time in milliseconds
Allocates required memory	Requires less memory space
Fully isolated and hence more secure	Process-level isolation, possibly less secure

A To-Do List

- Read Chapters 3, Sec 1, 2 and part of 3
- Building your team