PA200 - Cloud Computing

Lecture 3: Virtualization technologies

Warm-up

Let's rehearse on the previous lectures...

What's the cloud?

1. Usage model of computer resources
2. Networked computers
3. Distributed computing technology
4. A collection of heterogeneous computers

Cloud traits?

1. High availability
2. On-demand self-service
3. High performance
4. Broad network access
5. Resource pooling
6. Rapid elasticity
7. Measured service
8. Improved information security

Cloud service models?

1. Software as a Service
2. Application as a Service
3. Platform as a Service
4. Infrastructure as a Service
5. Data as a Service

Cloud deployment models?

1. Public Cloud
2. Private Cloud
3. Hybrid Cloud
4. Personal Cloud
5. Community Cloud
6. Enterprise Cloud

In this lecture

- History of concurrency and virtualization
- Virtualization technologies
- Hypervisors
- Cloud software

History of virtualization

- Early 1960: batch processing
- 1970: first commercial time-sharing system - IBM S/370
- 2005: Intel VT-x, AMD-V - new instruction set
- 2005-: VMware, VirtualBox, KVM...

Mid sixties: S/360

IBM S/360: single-program, batched jobs

- Expensive idling on I/O

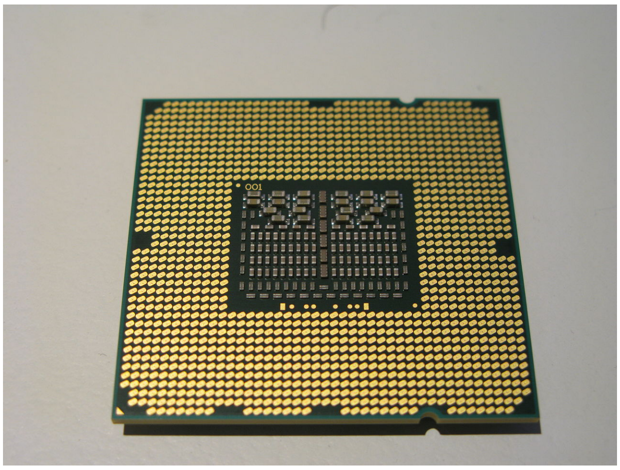
Early seventies: TSS/360

IBM TSS/360 & S/370 introduced virtual memory, privilege separation

- Eventually lost out to batch jobs
- Security concerns due to multi-tenancy

Raise of virtualization

- 1990+: software virtualization in desktop PC
- 2005+: HW assisted virtualization (Intel VT-x, AMD-V)
- Web-farms as a driving factor



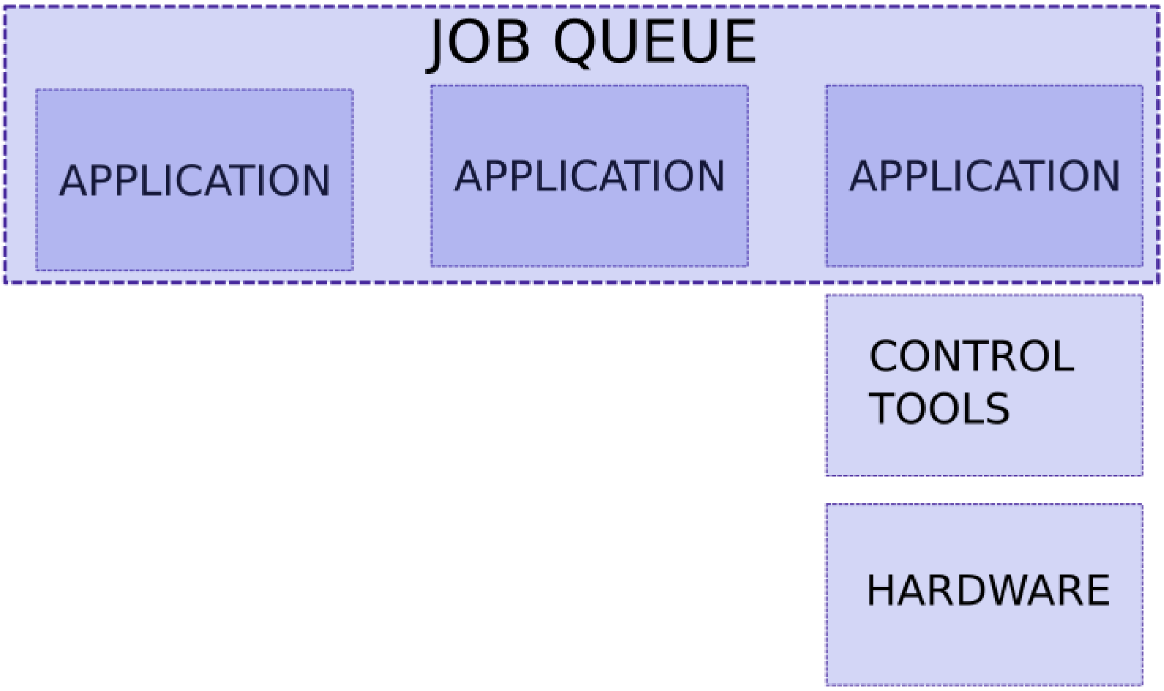
Concurrency and isolation

Many forms of concurrency and isolation:

- Multi-programming
- Multi-tasking
- Multi-threading
- Virtual machines
- Containers
- CPUs:
    - Multi-core
    - Hyper-threading

Concurrency: multi-programming

Sequential processes

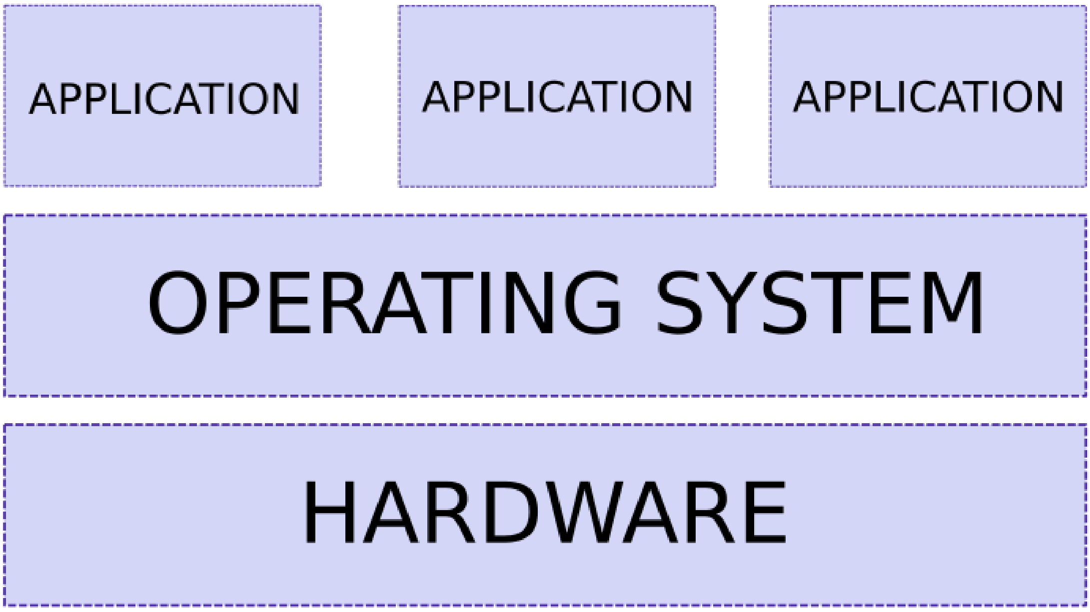


HW-assisted multitasking

- Resource access separation at CPU level
    - Memory protection, protection rings
- CPU scheduling
    - Cooperative
    - Preemptive
    - Real-time
- Intel 80286: protecting apps
- Intel 30386: protecting kernel and apps

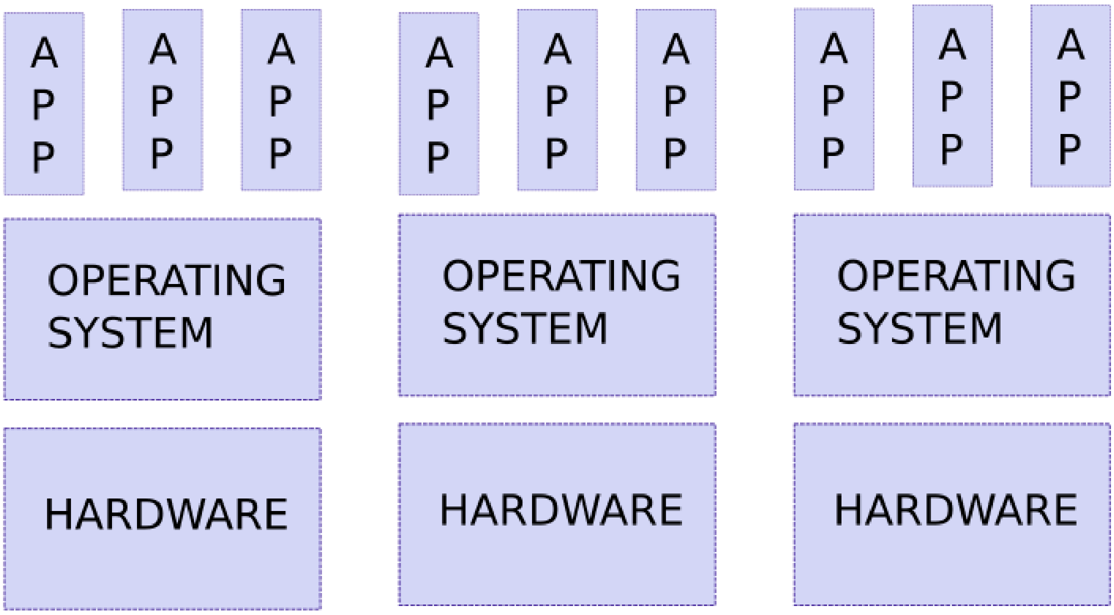
Concurrency: multi-tasking

Concurrent processes



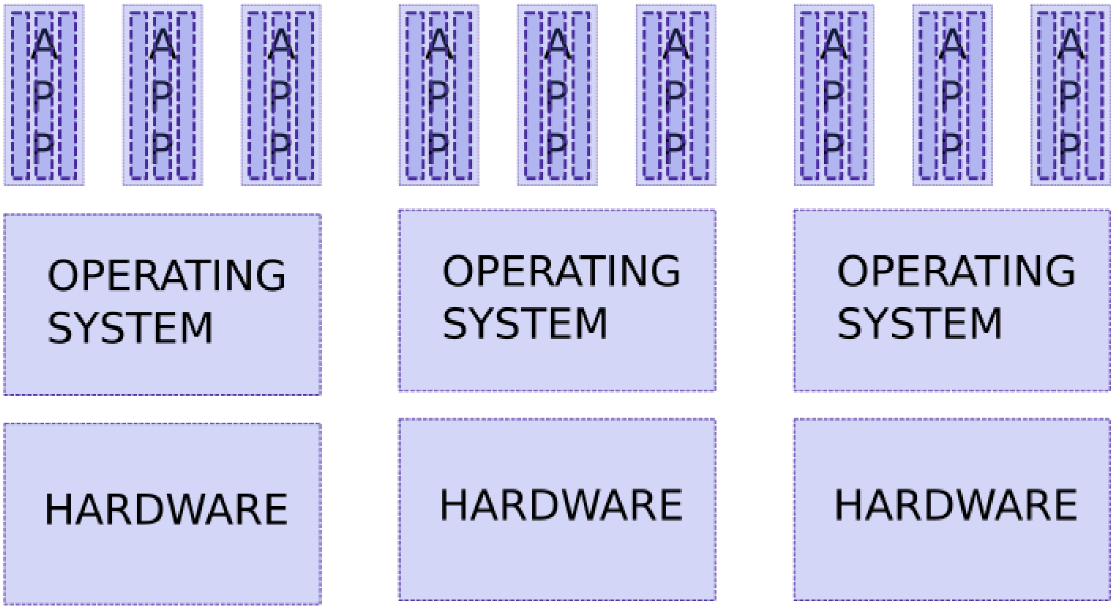
Concurrency: Multiple systems

Multiple systems, concurrent processes



Concurrency: Multiple threads

Multiple systems, concurrent processes, concurrent threads



HW-assisted virtualization

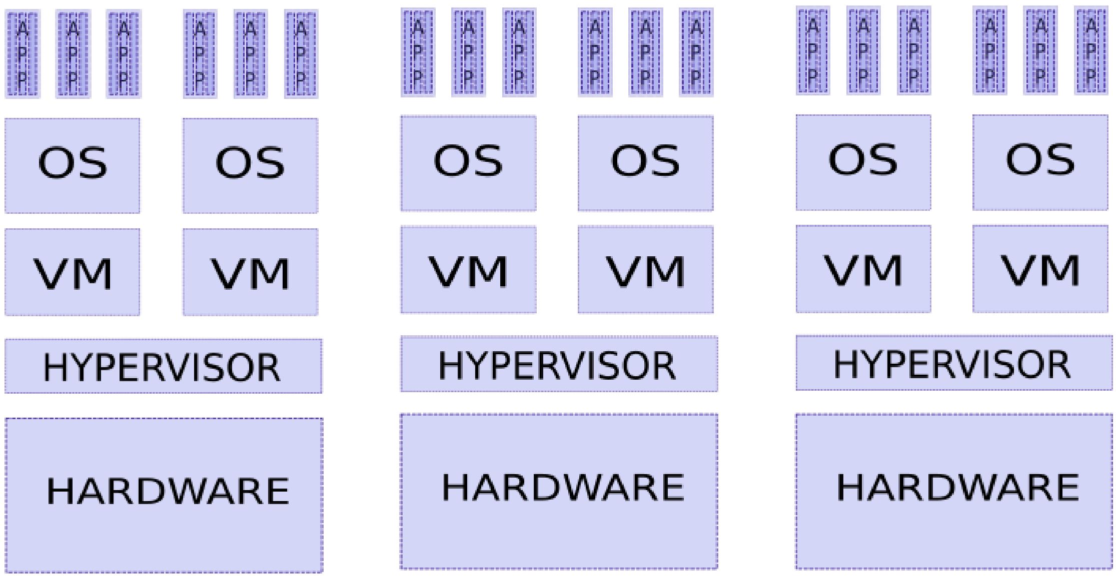
- The concept of VM at the HW level
- Resource protection and isolation
- S/360-67, Intel VT-x, AMD-V

Concurrency: Virtual machines

- Virtual machine emulates a physical computer
- OS executes within a VM
- Tenant OSes are isolated from each other
- VMs are heavy and expensive

Concurrency: Virtual machines

Multiple systems, VMs, processes, threads

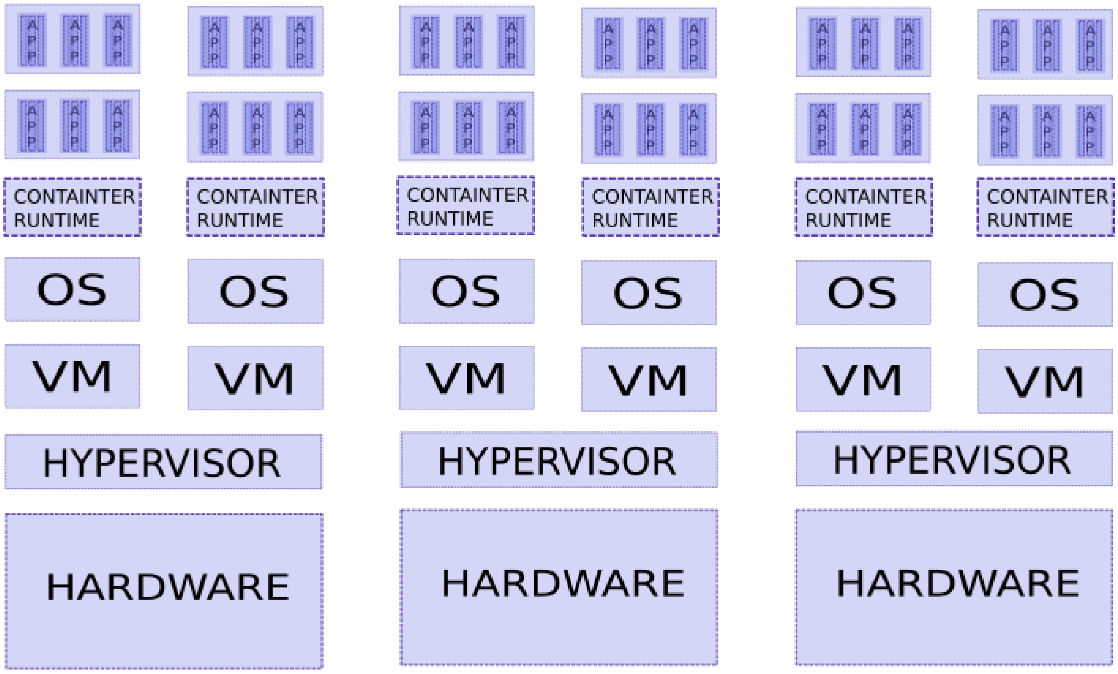


OS-level virtualization: containers

- Processes share the same kernel
- Processes have isolated memory, file system, network and PID spaces
- Many processes can be contained at once
- Containers are cheap and lightweight

Concurrency: Containers

Multiple systems, VMs, containers, processes, threads

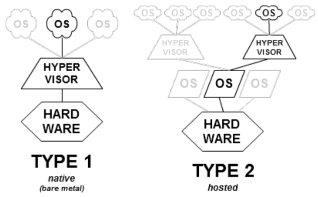


What makes up a cloud

- Isolated execution environment
    - Virtual machines and/or
    - Containers
- Guest life cycle management
    - Hypervisors
- Higher order infrastructure
    - Instance management
    - Access control
    - Networking
    - Storage

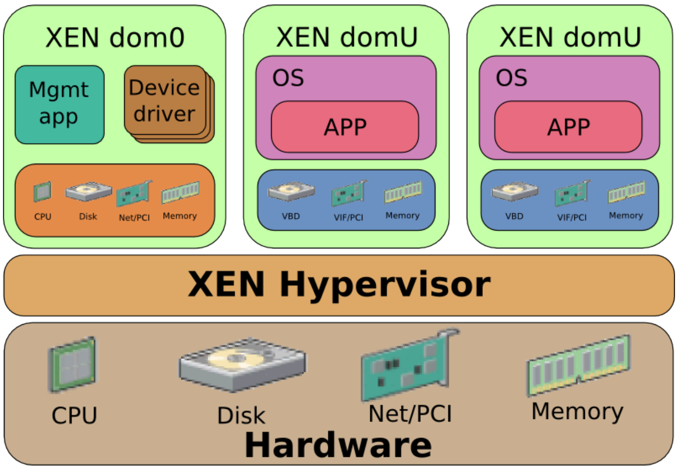
Hypervisors

- Type 1: Native
    - Runs directly on host's hardware
- Type 2: Hosted
    - The hypervisor and VMs are processes of host's operating system



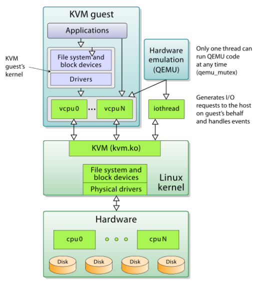
Example Type 1 hypervisors

Xen, Oracle VM Server, Microsoft Hyper-V, VMware ESX/ESXi



Example Type 2 hypervisors

VMware, Oracle VirtualBox, Parallels Desktop, Linux KVM (+QEMU), FreeBSD Bhyve



Native-hosted hypervisors

- Type 1/2:
  - Linux KVM
  - BSD bhyve

Full or para-virtualization

- Full virtualization
  - Unmodified OS on top of hypervisor
- Para-virtualization
  - Modified OS calls hypervisor API

Bare metal machine hypervisor

- Traditional hypervisors
  - Manage VMs running on bare metal machines
- Baremetal machine hypervisors
  - Manager bare metal machines
  - In the same way as VMs

Full virtualization infrastructure

- Basic cloud features
  - Hypervisor abstraction layer
  - User authentication and accounting
  - Instance life cycle management (scheduling)
  - Automated OS deployment and configuration
  - Virtualized network (SDN)
  - Storage services
- More features
  - High-availability services
  - Instance monitoring and scaling
  - Instance backup/migration
  - Virtualized databases
  - User interfaces

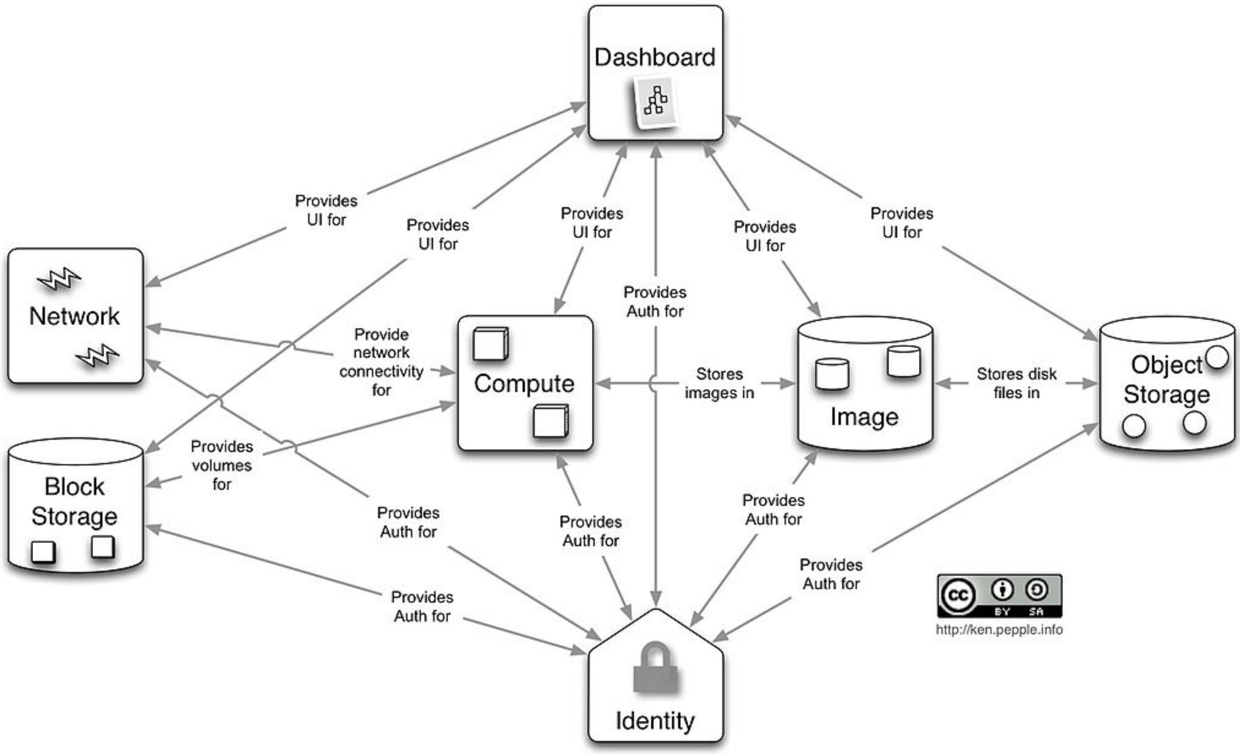
Example: oVirt

- Lightweight, all-in-one cloud (e.g. desktop)
- KVM as a hypervisor
- Reliable VMs (pets)
- Vertical scalability

Example: OpenStack

- Heavyweight, large cloud
- Large collection of loosely-coupled projects
- Unreliable, replaceable VMs (cattle)
- Horizontal scalability

OpenStack components



Container orchestration

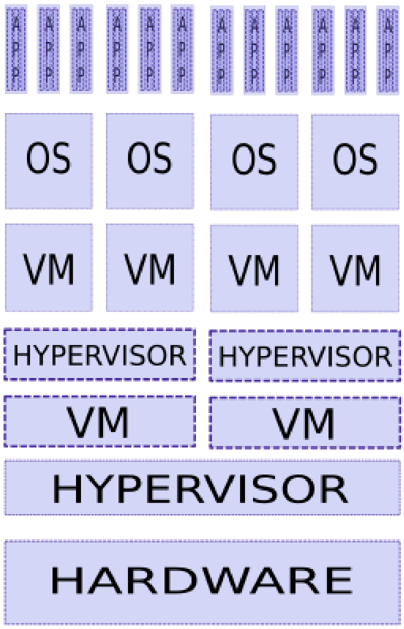
- Basic features
    - Container runtime abstraction layer
    - Container life cycle management (scheduling)
    - Resource management: memory, CPU, file system, storage volumes, network addresses etc.
    - Clustering
- More features
    - Load balancing and scaling
    - Container images management
    - User interfaces

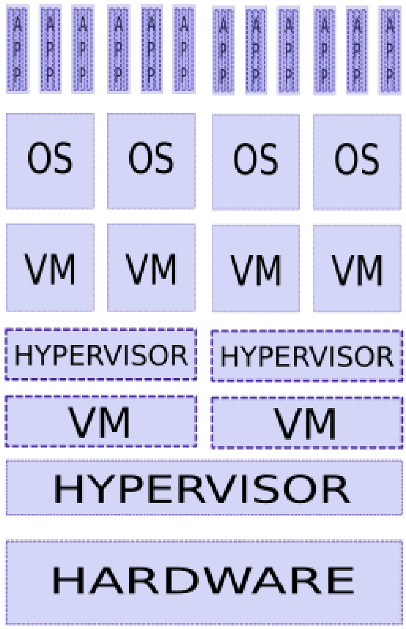
Example: container orchestration

- Docker Swarm
- Kubernetes / OpenShift
- Amazon EC2 Container Service
- Nomad

Nested virtualization

Multiple systems, VMs, nested VMs, processes, threads





Recap: the age of virtualization?

1. IBM 700/7000, since 1952
2. CP-40 research project, early sixties
3. IBM S/370, 1970
4. Gameframes, since 2007
5. Intel VT-x, AMD-V, since 2005

Recap: virtualization technologies?

1. Multi-tasking
2. Multi-threading processes
3. Containers
4. Hyper-threading CPU
5. Multi-core CPU
6. Intel VT-x, AMD-V
7. Multi-programming

Recap: hypervisor types?

1. Hybryd
2. Bare-metal
3. Native
4. Hosted
5. Para-hypervisor

Recap: what makes up a cloud?

1. Baremetal machines w/ CPU-level virtualizaiton
2. Hypervisors
3. Cloud control plane / runtime
4. cloud services (storage, networking, etc)

Recap: virtualization vs containers?

1. We can run OS in a container
2. We can run different OS'es in containers
3. Containers are more secure than VM
4. Containers consume less resources than VM
5. We can run Windows app in Linux container

Q&A

?