CLOUD COMPUTING
Virtualization

Zeinab Zali
Isfahan University Of Technology

References: “Cloud Computing, Theory and Practice, Chapter 8 and 10
Official documents and published papers of virtualization tools

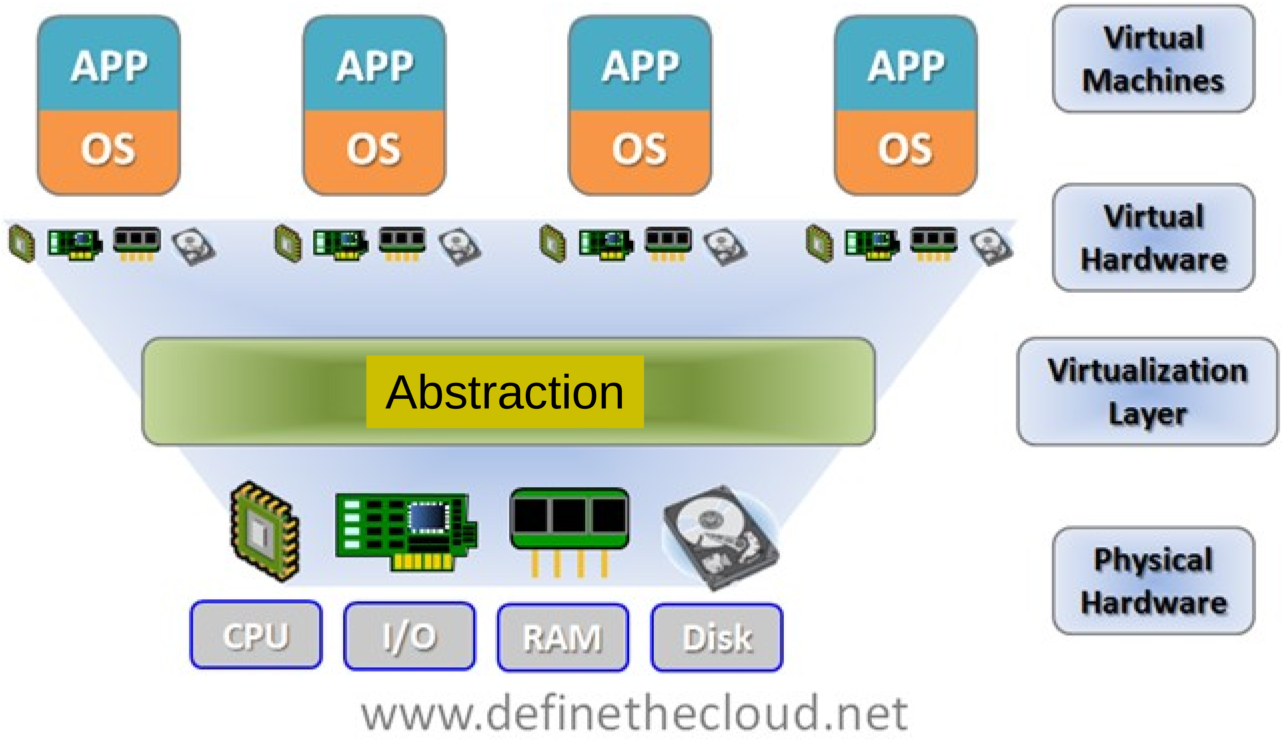
Virtualization concepts

Resource Sharing in clouds

- Economics of Clouds requires sharing resources
- How do we share a physical computer among multiple users?
  - Answer: Abstraction
  - Abstraction: what a generic computing resource should look like
  - Then providing this abstract model to many users

Resource Sharing in clouds

- Abstraction enables Virtualization



Virtualization

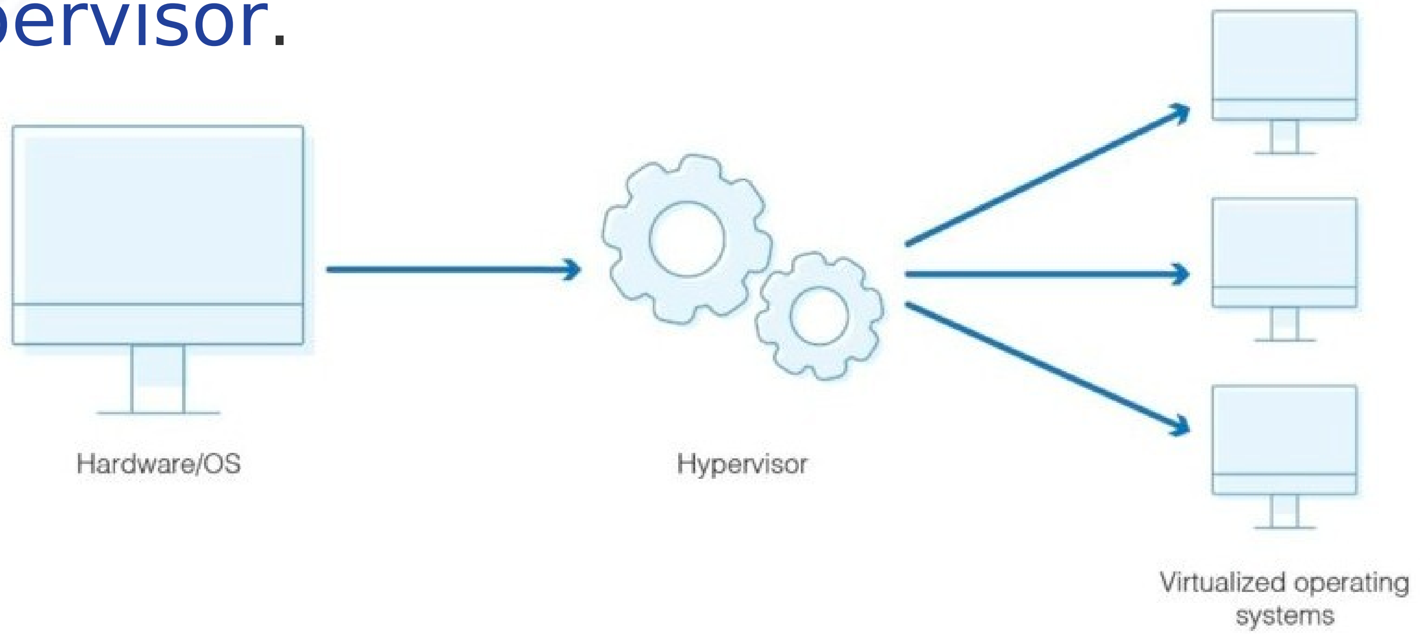
- Clouds are based on Virtualization
  - offer services based mainly on virtual machines, remote procedure calls, and client/servers
- A VM is an isolated environment with access to a subset of physical resources of the computer system
- The instantaneous demands for resources of the applications running concurrently are likely to be different and complement each other

Virtualization benefits

- supporting portability, improve efficiency, increase reliability, and shield the user from the complexity of the system
- providing more freedom for the system resource management because VMs can be easily migrated
- allowing a good isolation of applications from one another

Hypervisor

- System virtualization is implemented by a thin layer of software on top of the underlying physical machine architecture; this layer is referred to as a virtual machine monitor or hypervisor.

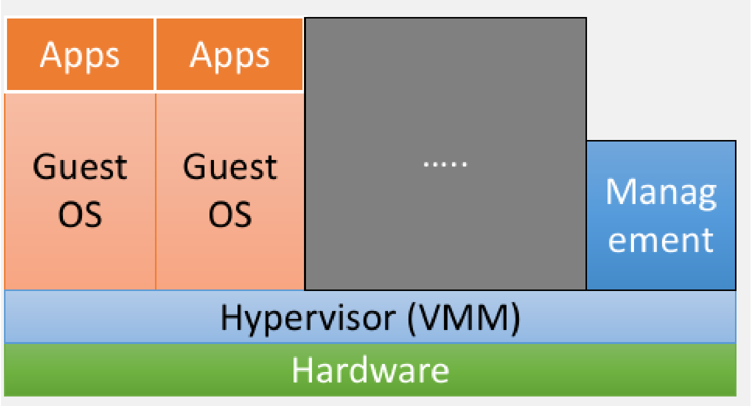


Types of Virtualization

- Native or full virtualization
- Para-virtualization
- OS level
  - Containers
  - Jails
  - Chroot
  - Zones
  - Open-VZ → Virtuozzo

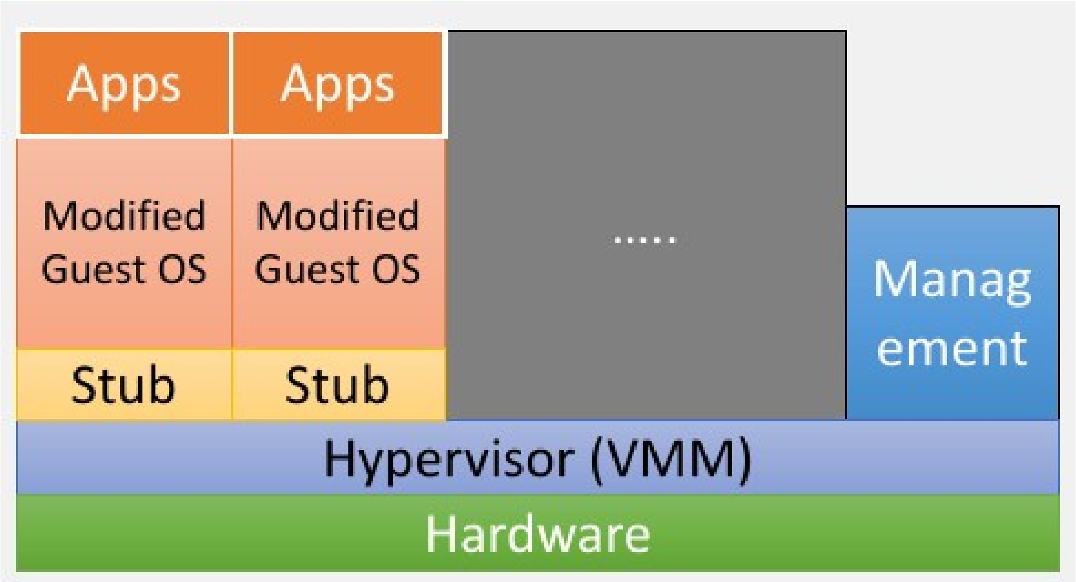
Full/native Virtualization

- the virtual machine simulates enough hardware to allow an unmodified "guest" OS (one designed for the same CPU) to be run in isolation
- the hardware abstraction layer of the guest OS must have some knowledge about the processor architecture.
  - Requires virtualizable architecture (HW assisted)
- OS sees exact hw
- Example: Vmware, Virtualbox



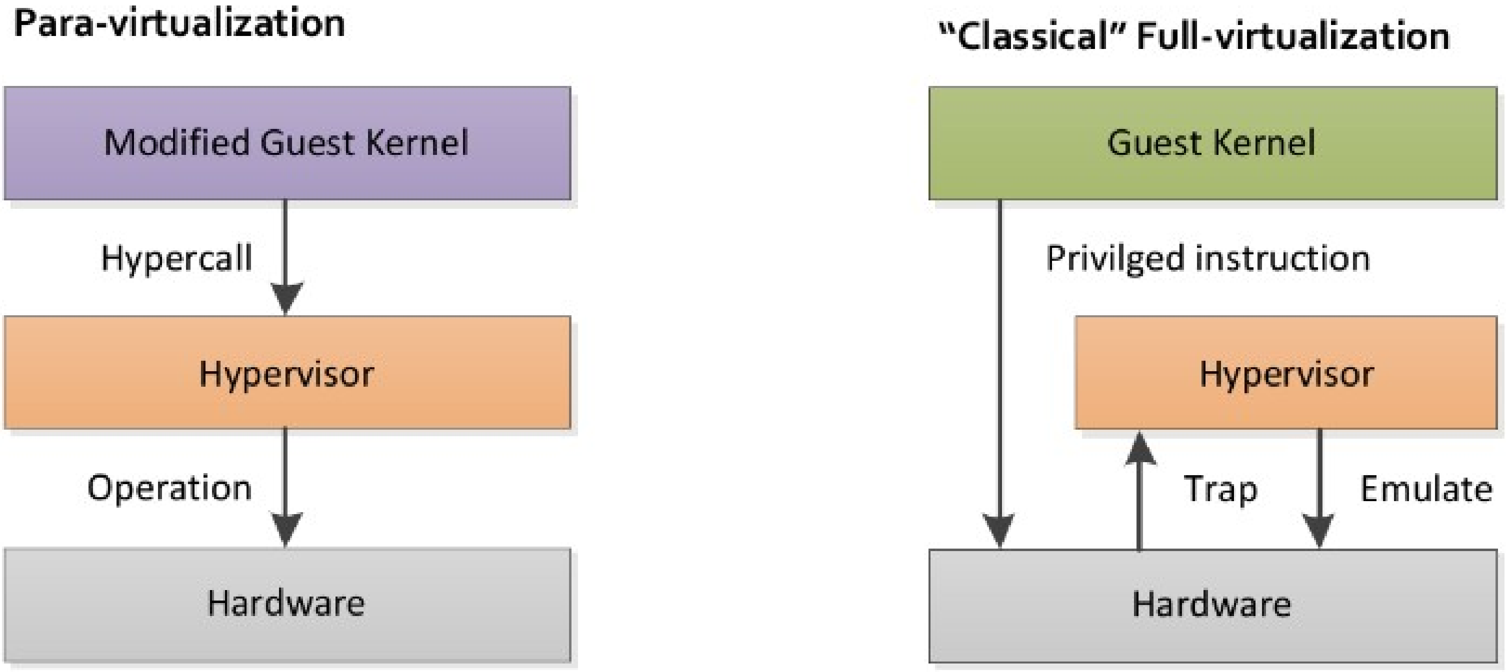
Para Virtualization

- the VM does not necessarily simulate hardware
- VM offers a special API that can only be used by modifying the "guest" OS
  - OS knows about VMM
- Improved performance with low overhead
- Example: Xen



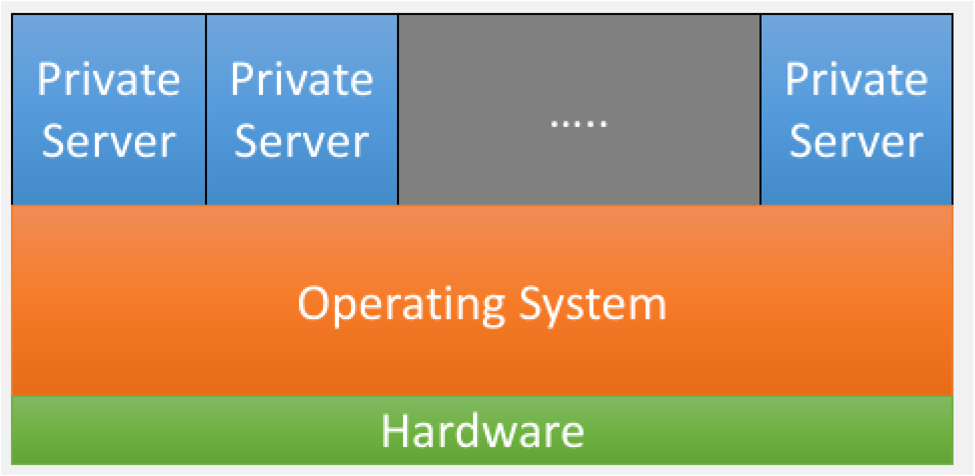
Full vs Para-Virtualization

In full virtualization, guests will issue a hardware calls but in paravirtualization, guests will directly communicate with the host (hypervisor) using the drivers



OS-level virtualization

- virtualizing a physical server at the operating system level, enabling multiple isolated and secure virtualized servers to run on a single physical server.
- Examples:
  - Linux-Vserver
  - Solaris Containers
  - FreeBSD Jails
  - Chroot
  - Cgroups
  - OpenVZ

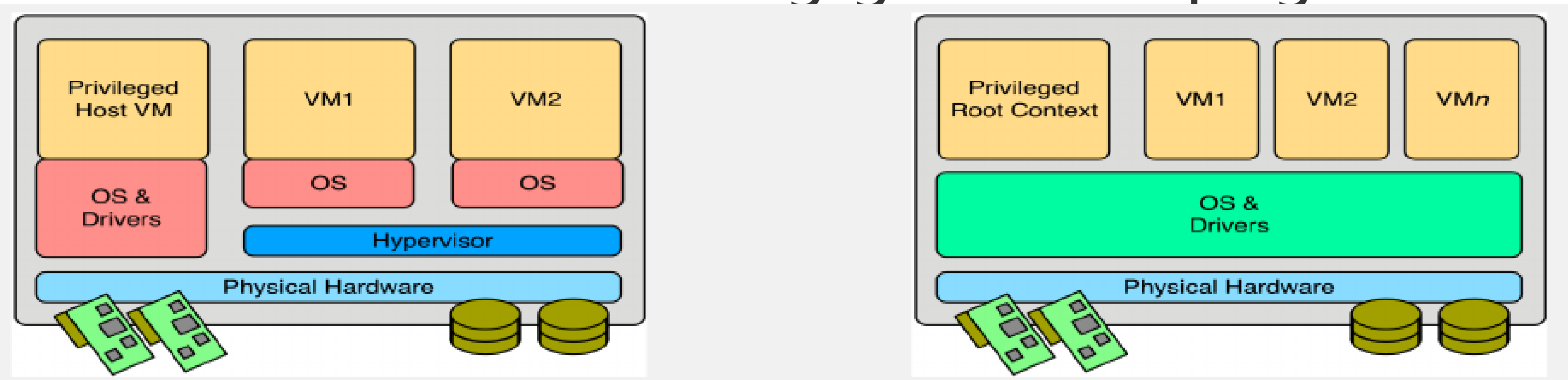


Containers

- Containers are based on operating-system-level virtualization
- An application running inside a container is isolated from another application running in a different container
  - both applications are isolated from the physical system where they run
- Containers are portable and the resources used by a container can be limited

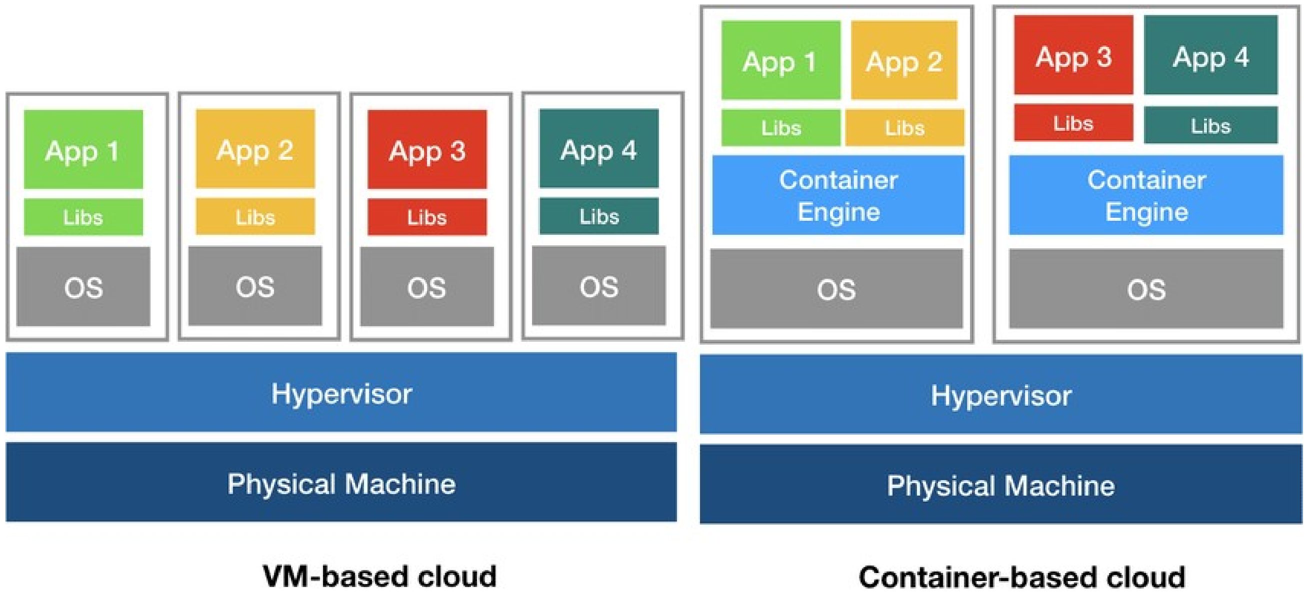
Containers vs Hypervisors

- Containers
  - Share host OS and drivers
  - Have small virtualization layer
  - Naturally share pages
- Hypervisors
  - Have separate OS plus virtual hardware
  - Have trouble sharing guest OS pages



Containers vs Hypervisors

- Containers are more elastic than hypervisors
- Container slicing of the OS is ideally suited to cloud
- Many Cloud providers use containers to support PaaS
- Hypervisors' only advantage in IaaS is support for different OS families on one server



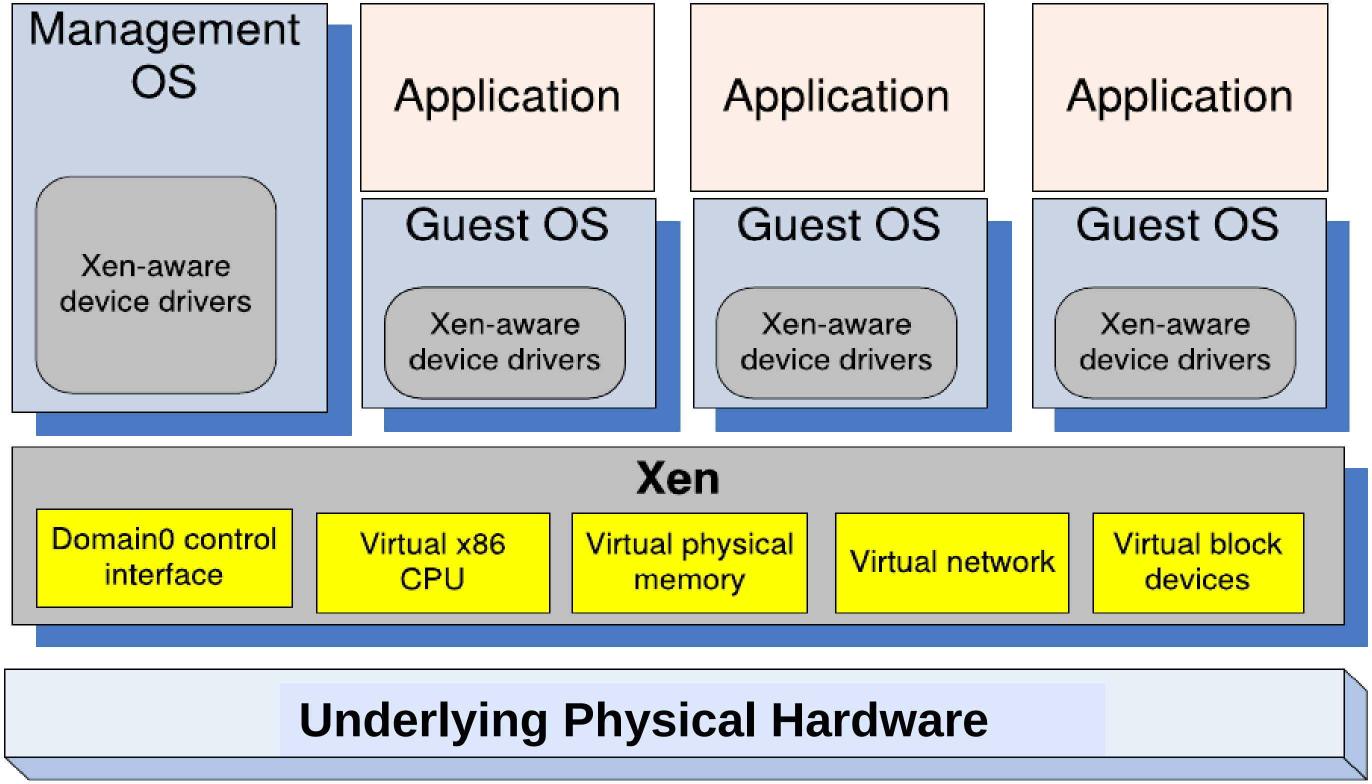
Virtualization tools

Xen

- Xen is an x86 virtual machine monitor
- The design is targeted at hosting up to 100 VMs simultaneously on a modern server
- Xen allows operating systems such as Linux and Windows XP to be hosted simultaneously for a negligible performance overhead
  - at most a few percent compared with the unvirtualized case
- Xen supports Paravirtualization (as well as full virtualization)
  - it does require modifications to the guest operating system

Xen Architecture

- Dom0: The management OS dedicated to the execution of Xen control functions and privileged instructions
- DomU: Guest operating systems and applications
  - A guest OS could be XenoLinix, XenoBSD, or XenoXP



Xen domains

- domain0 is created at boot time which is permitted to use the control interface
  - Responsible for hosting the application-level management software.
- The control interface provides
  - The ability to create and terminate guest domains
  - Control guest domains associated scheduling parameters
  - Control guest domains physical memory allocations
  - Control the guest domains access given to the machine’s physical disks and network devices

Xen control interactions

- synchronous calls from a domain to Xen may be made using a hypercall
  - A software trap into the hypervisor to perform a privileged operation
- notifications are delivered to domains from Xen using an asynchronous event mechanism
  - similar to traditional Unix signals, there are only a small number of events, each acting to flag a particular type of occurrence.
  - Examples: events are used to indicate that new data has been received over the network, or that a virtual disk request has completed.

Xen: paravirtualized x86 interface

Memory Management	Segmentation	Cannot install fully-privileged segment descriptors and cannot overlap with the top end of the linear address space.
	Paging	Guest OS has direct read access to hardware page tables, but updates are batched and validated by the hypervisor. A domain may be allocated discontiguous machine pages.
CPU	Protection	Guest OS must run at a lower privilege level than Xen.
	Exceptions	Guest OS must register a descriptor table for exception handlers with Xen. Aside from page faults, the handlers remain the same.
	System Calls	Guest OS may install a 'fast' handler for system calls, allowing direct calls from an application into its guest OS and avoiding indirecting through Xen on every call.
	Interrupts	Hardware interrupts are replaced with a lightweight event system.
	Time	Each guest OS has a timer interface and is aware of both 'real' and 'virtual' time.
Device I/O	Network, Disk, etc.	Virtual devices are elegant and simple to access. Data is transferred using asynchronous I/O rings. An event mechanism replaces hardware interrupts for notifications.

Xen: Memory Management

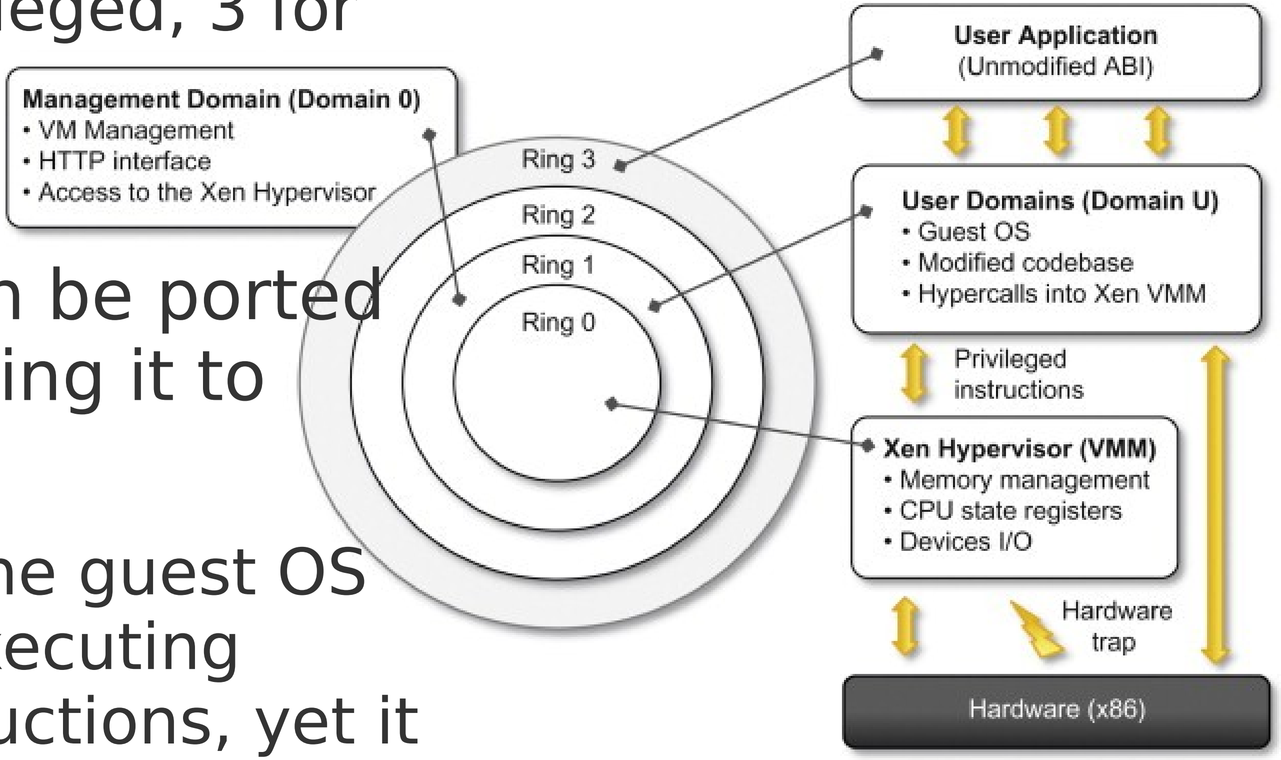
- Each time a guest OS requires a new page table it allocates and initializes a page from its own memory reservation and registers it with Xen
- At this point the OS must relinquish direct write privileges to the page-table memory
  - all subsequent updates must be validated by Xen.

Xen: CPU management

- the insertion of a hypervisor below the operating system violates the usual assumption that the OS is the most privileged entity in the system.
- In order to protect the hypervisor from OS misbehavior (and domains from one another) guest OSes must be modified to run at a lower privilege level.
- Efficient virtualizion of privilege levels is possible on x86 because it supports four distinct privilege levels in hardware

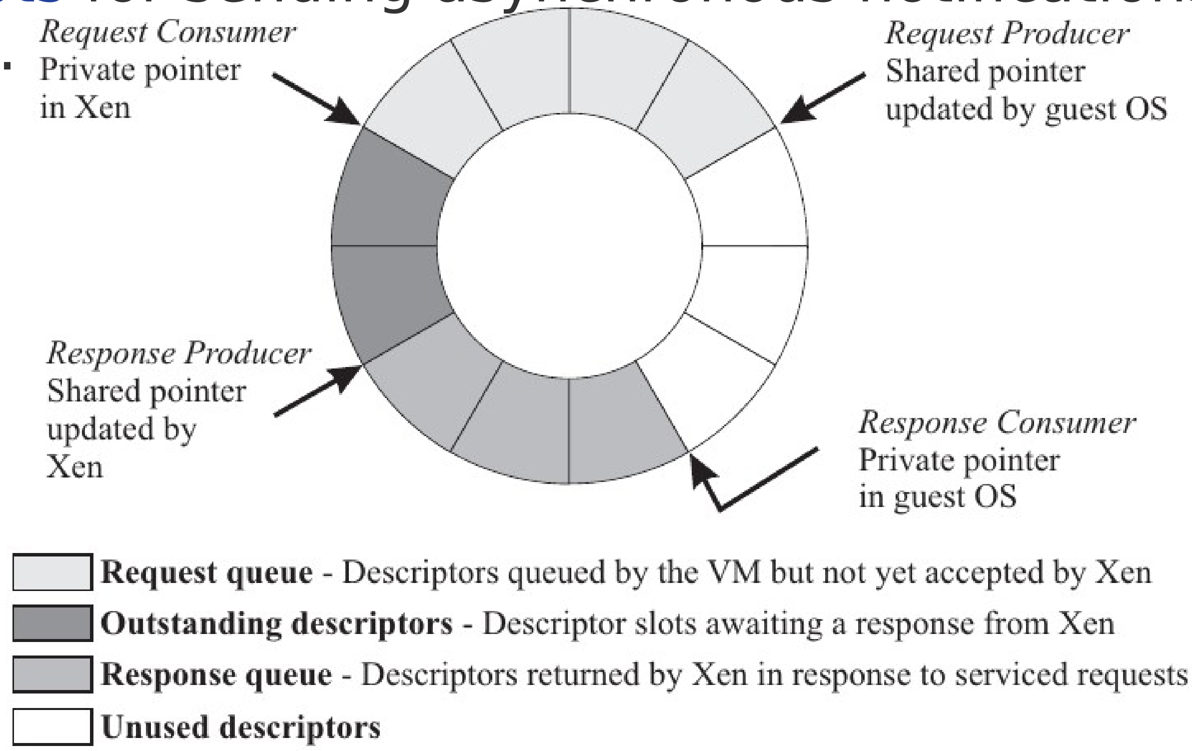
Xen: CPU management

- 4 distinct privilege levels
  - 0 for most privileged, 3 for least privileged
- Any guest OS can be ported to Xen by modifying it to execute in ring 1
  - This prevents the guest OS from directly executing privileged instructions, yet it remains safely isolated from applications running in ring 3



Xen: I/O management

- I/O data is transferred to and from each domain via Xen, using shared-memory and asynchronous buffer descriptor rings.
  - an event-delivery mechanism instead of hardware interrupts for sending asynchronous notifications to a domain.

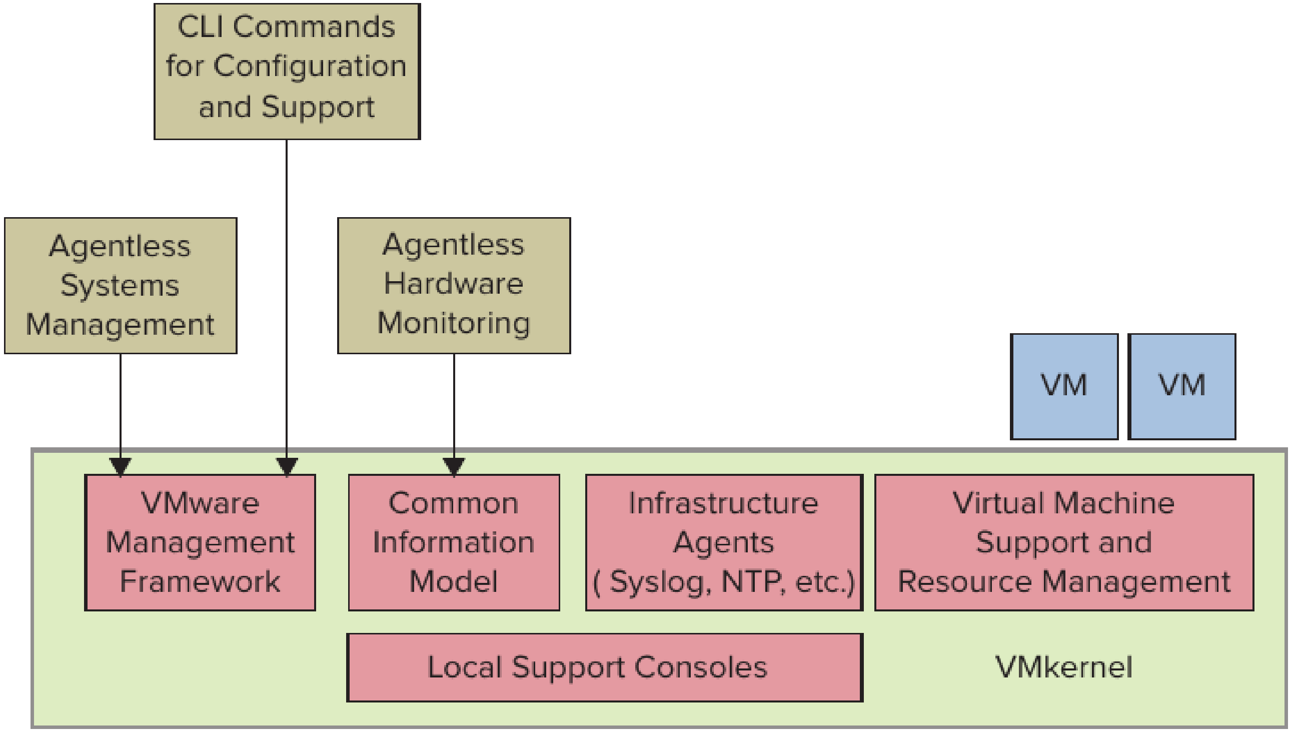


Xen properties

- Xen separates the hypervisor execution from management OS, management stack, device drivers, and guests
- Components are interchangeable – choose the best OS for domain0 to support your needs
- Strong isolation between all components assisted with modern hardware and domains can restart without taking out full system
- The Xen hypervisor is the most used virtualization platform in the cloud computing space, with leading vendors such as Amazon, Cloud.com, GoGrid, and Rackspace

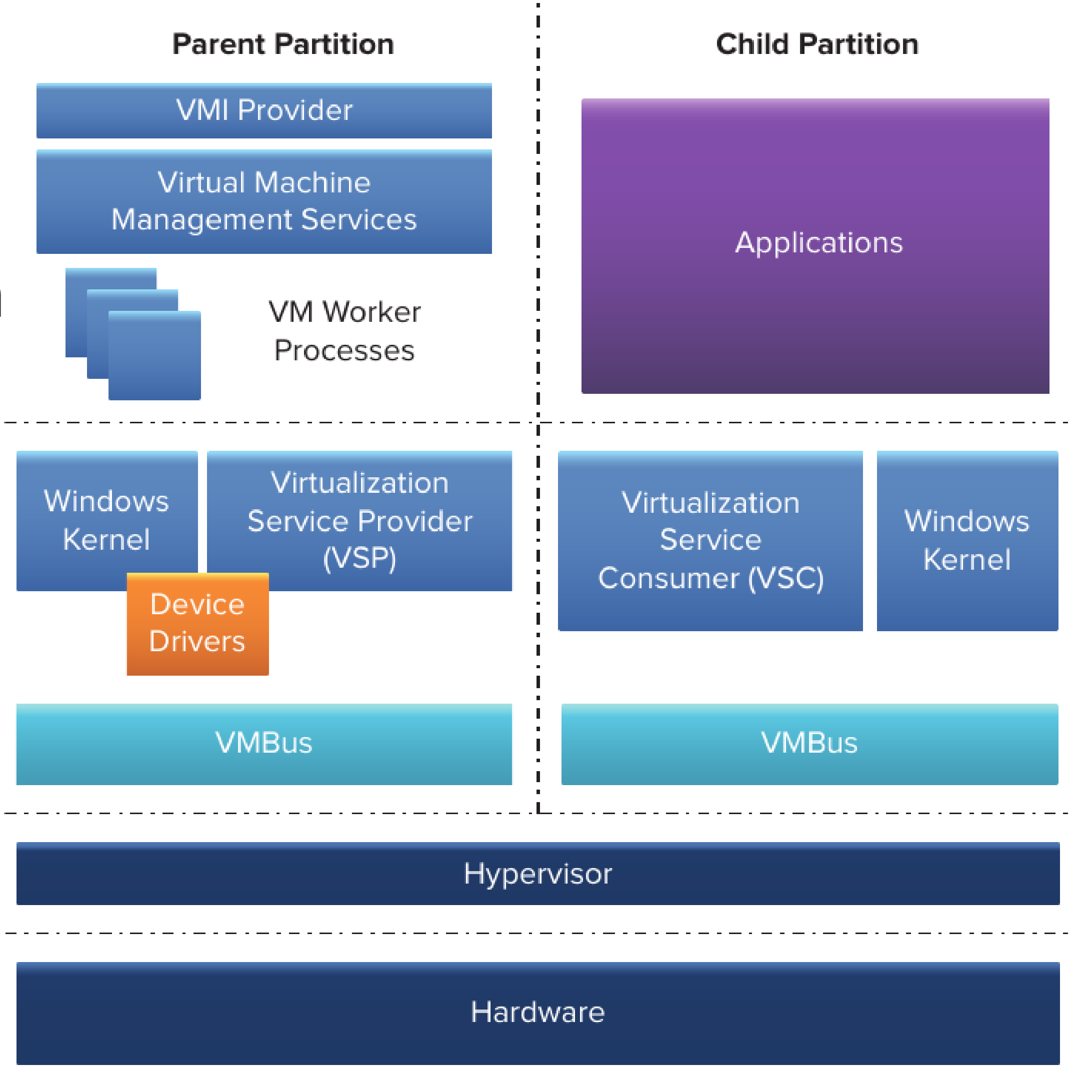
VMware's ESXi Server

- Elastic Sky X Integrated: A type 1 hypervisor including an OS kernel
- Maximum number of virtual machines per host: 1024
- Both para and full virtualization support



HyperV

- a Microsoft virtualization technology for certain x64 versions of Windows.
- Similar to the Xen model, it requires a special parent partition that has direct access to the hardware resources



Jail chroot

- A chroot operation changes the apparent root directory for a running process and its children
  - This artificial root directory is called a chroot jail
- To make chroot useful for virtualization, FreeBSD expanded the concept and introduced the jail command.
  - With jail it is possible to create various virtual machines, each having its own set of utilities installed and its own configuration

Linux Containers

- Better isolation as compared to a chroot
- Linux containers are open source.
- Unlike XEN or OpenVZ , no patch is required to the kernel.
  - apt-get install lxc-utils
  - lxc-create -f /etc/lxc/lxc-centos.conf

OpenVZ (I)

- OpenVZ, a system based on OS-level virtualization, uses a single patched Linux kernel
- The guest operating systems in different containers may be different software distributions, but must use the same Linux kernel version that the host uses
- An OpenVZ container emulates a separate physical server, it has its own files, users, process tree, IP address, shared memory, semaphores, and messages.
  - Each container can have its own disk quotas.

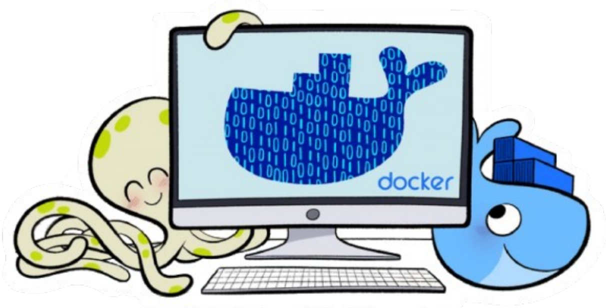
OpenVZ (II)

- OpenVZ has a two level scheduler:
  - at the first level, a fair-share scheduler allocates CPU time slices to containers based on cpuunits values.
  - The second level scheduler is a standard Linux scheduler deciding what process to run in that container.
- The I/O scheduler is also two-level;
  - each container has an I/O priority
  - the scheduler distributes the available I/O bandwidth according to priorities

OpenVZ (III)

- OpenVZ memory allocation is more flexible than in hypervisors based on paravirtualization
- The memory not used in one virtual environment can be used by other virtual environments.
- The system uses a common file system;
- Vserver is another tool provides virtualization for GNU/Linux systems
  - Pre-patched kernel included with Debian

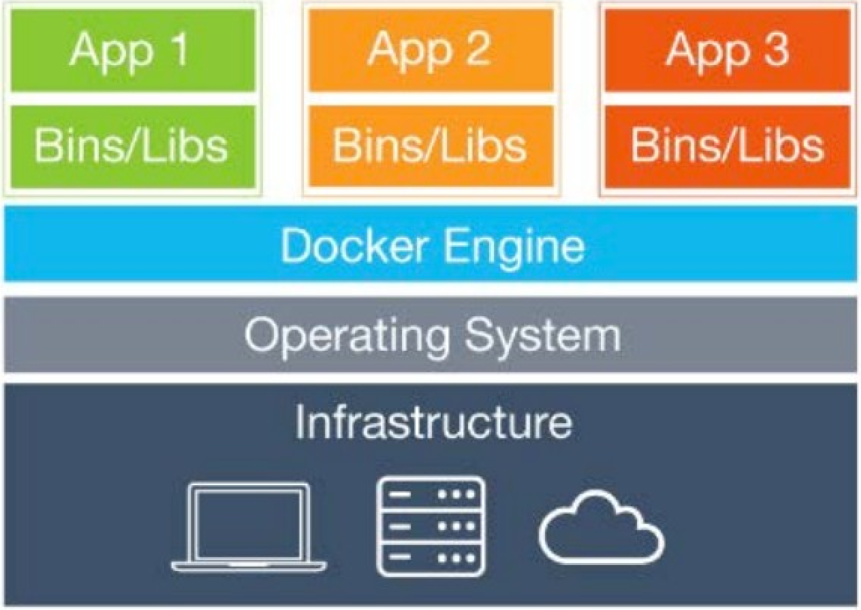
Docker

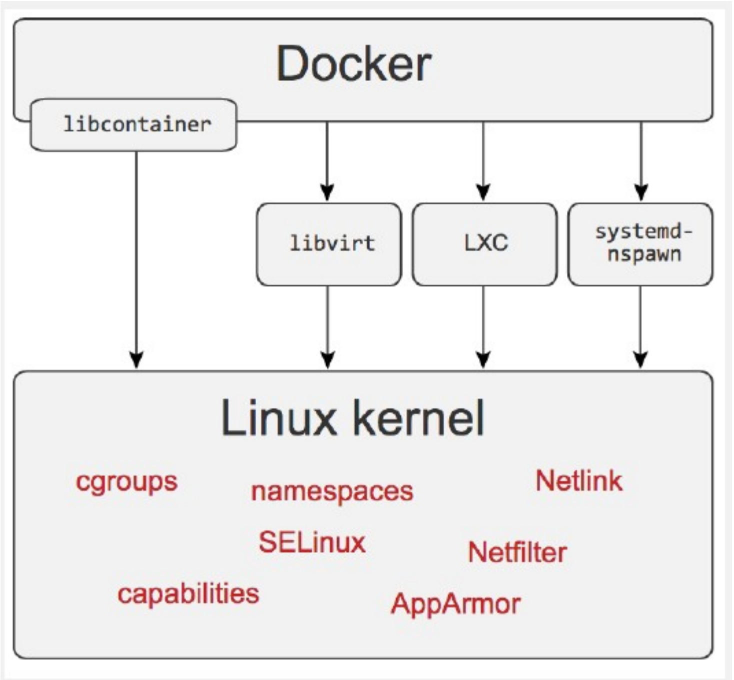


- A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings
  - A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another
- Docker team: Red Hat, IBM, Google, Cisco Systems and Amadeus IT Group

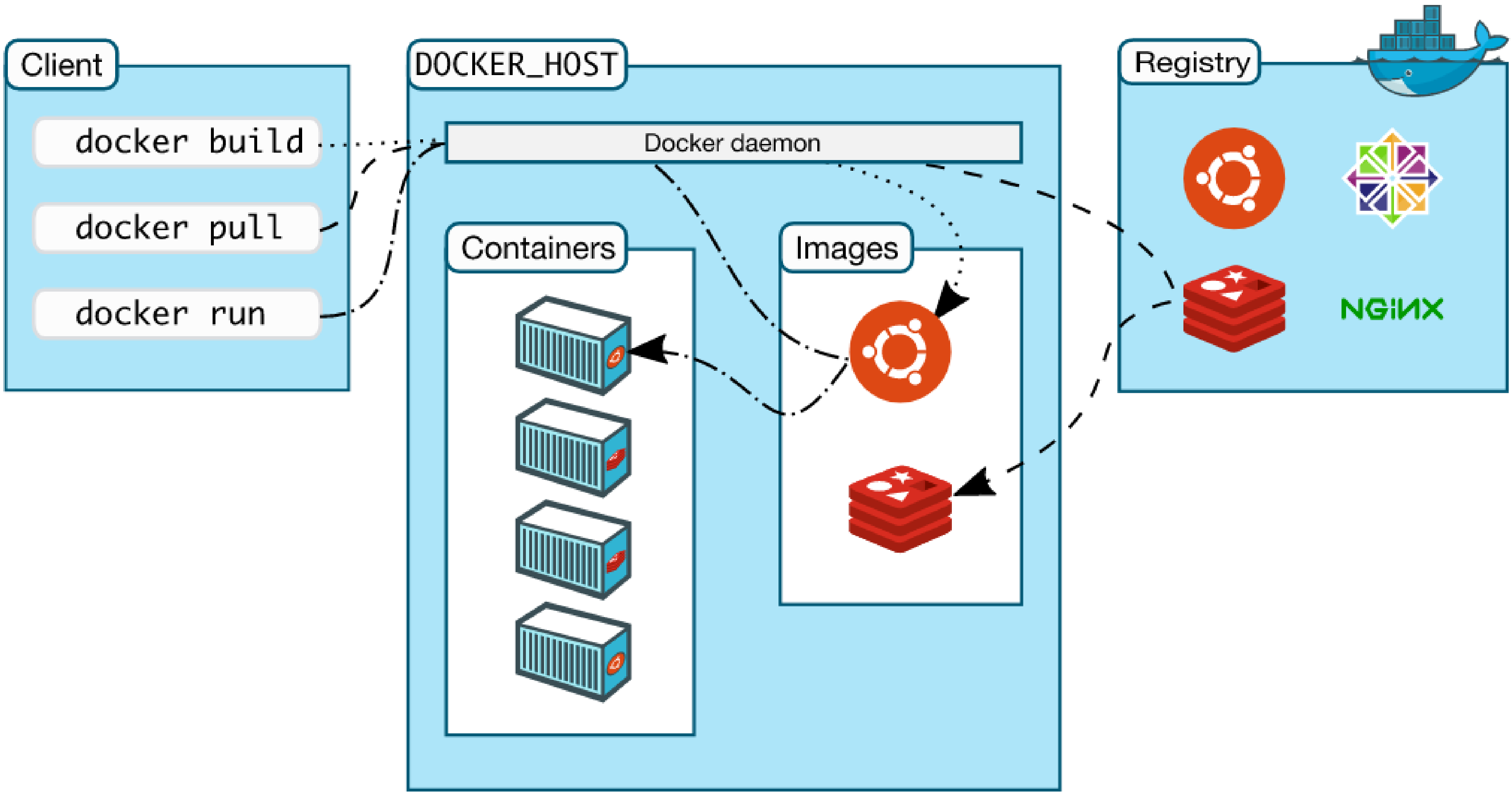
Docker

- The software that hosts the containers is called Docker Engine
  - Docker is installed on linux, windows and Mac





Docker architecture

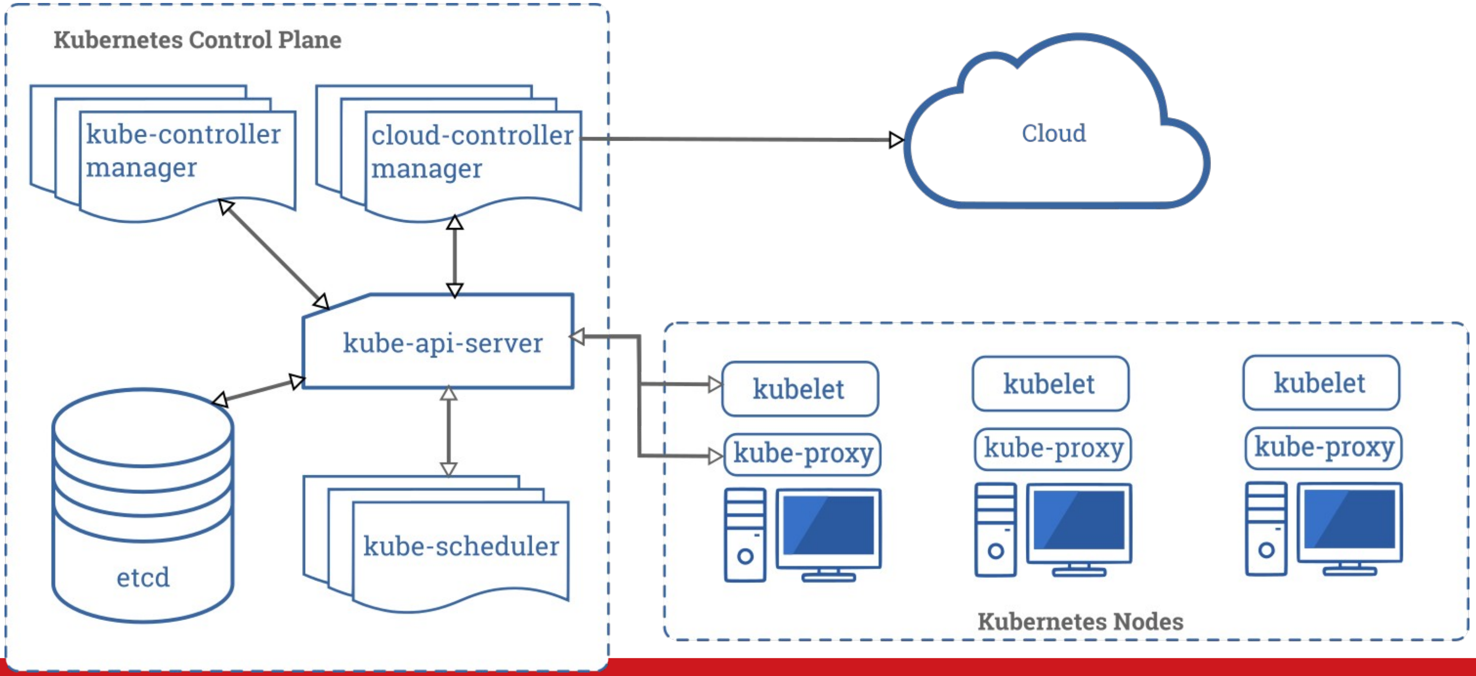


Kubernetes

- Kubernetes is an open source software system developed and used at Google for managing containerized applications in a clustered environment
  - It is a cluster manager for containers
  - It provides deployment, scaling, load balancing, logging, monitoring, etc., services common to PaaS Kubernetes

Kubernetes components (I)

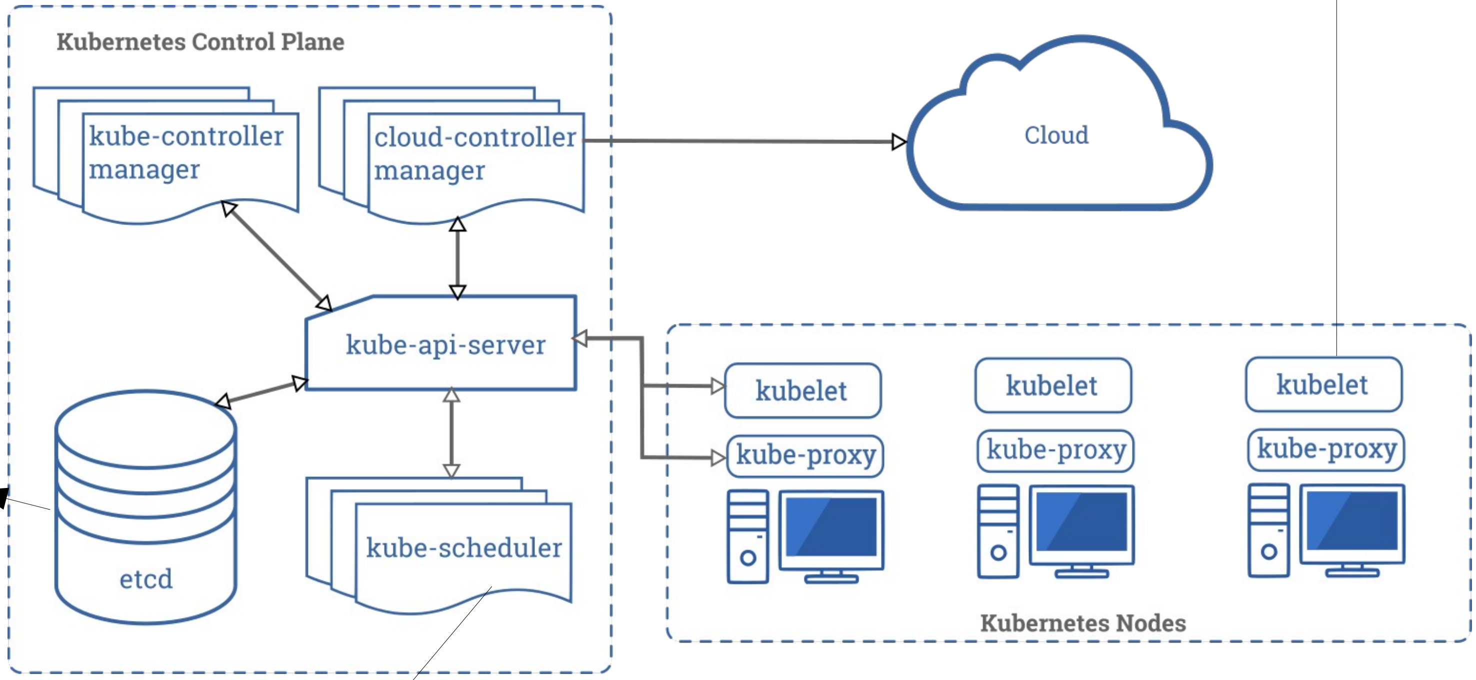
- A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications
- The worker node(s) host the Pods that are the components of the application workload
- The control plane manages the worker nodes and the Pods in the cluster



Kubernetes components (II)

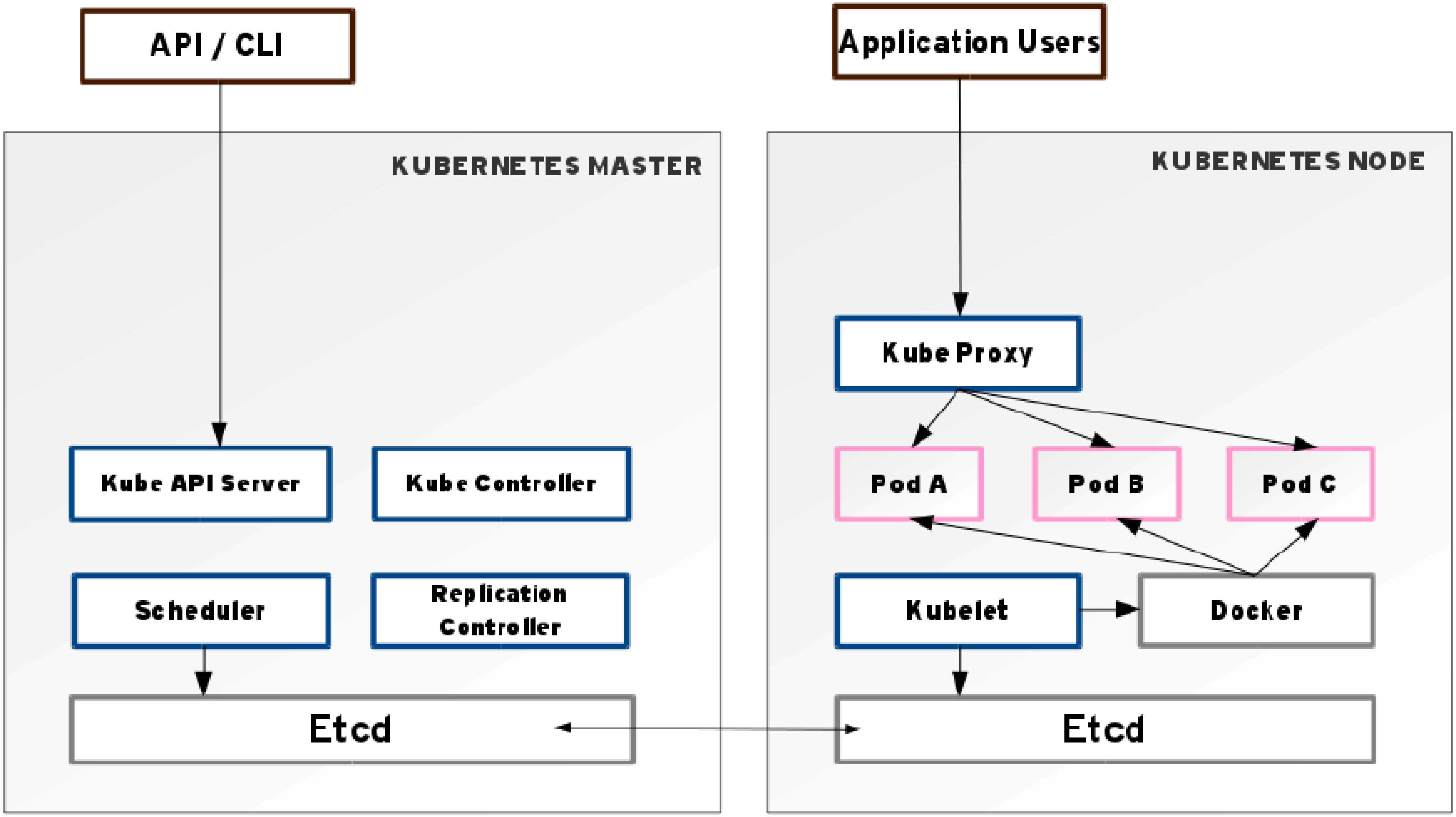
An agent that runs on each node in the cluster

Consistent and highly-available key value store used as Kubernetes’ backing store for all cluster data



watches for newly created Pods with no assigned node, and selects a node for them to run on

Kubernetes components (III)



Performance Evaluation of VM managers

Performance Metrics

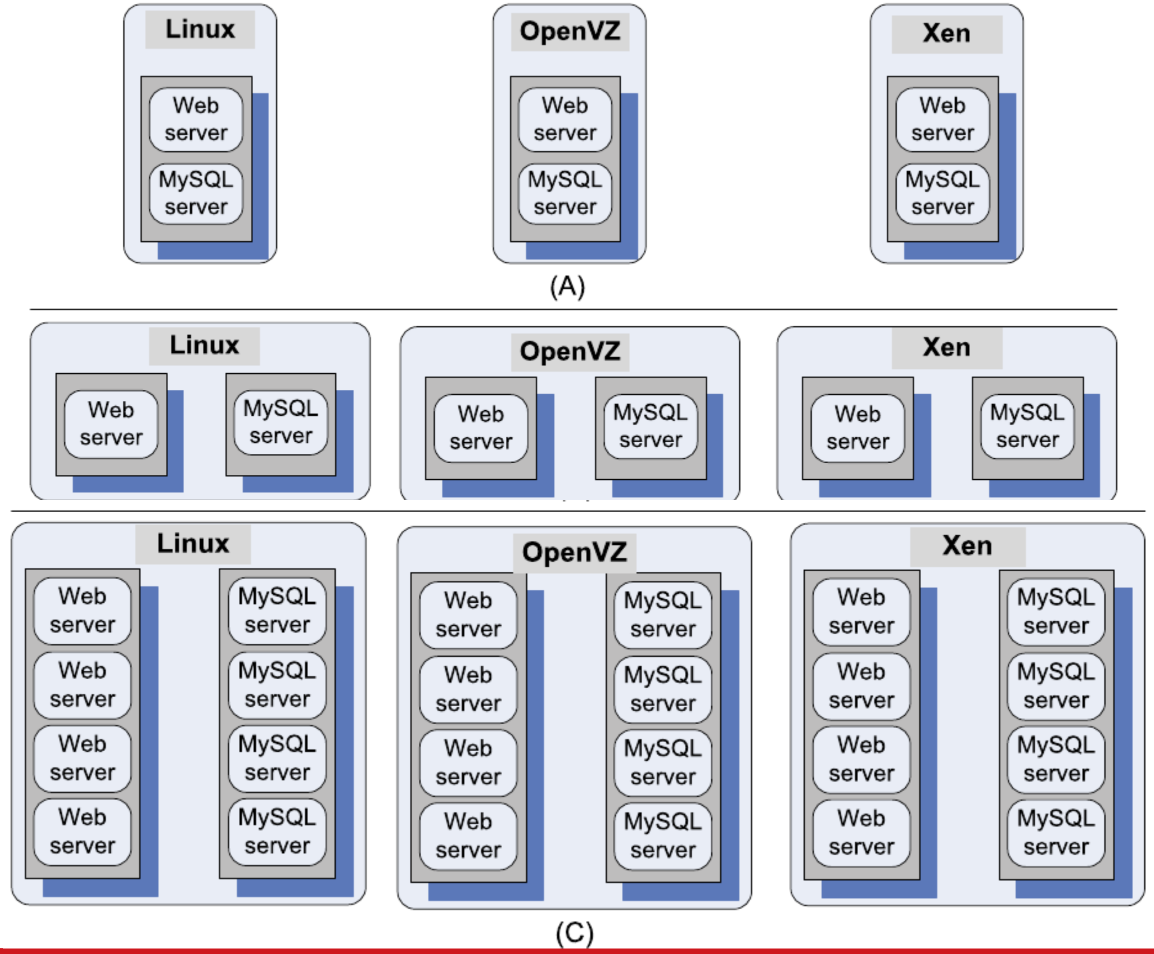
- the performance metrics analyzed are throughput and response time
- The specific questions examined are:
  - How does performance scale up with the load?
  - What is the impact on performance of a mixture of applications?
  - What are the implications of the load assignment on individual servers?

Motivation for multiplexing

- the load placed on system resources by a single application varies significantly in time
- A time series displaying CPU consumption of a single application clearly illustrates this fact and justifies CPU multiplexing among threads and/or processes
- The concept of application and server consolidation is an extension of the idea of creating an aggregate load consisting of several applications and aggregating a set of servers to accommodate this load

Performance comparision

- setup for the performance comparison of a native Linux system with OpenVZ and Xen

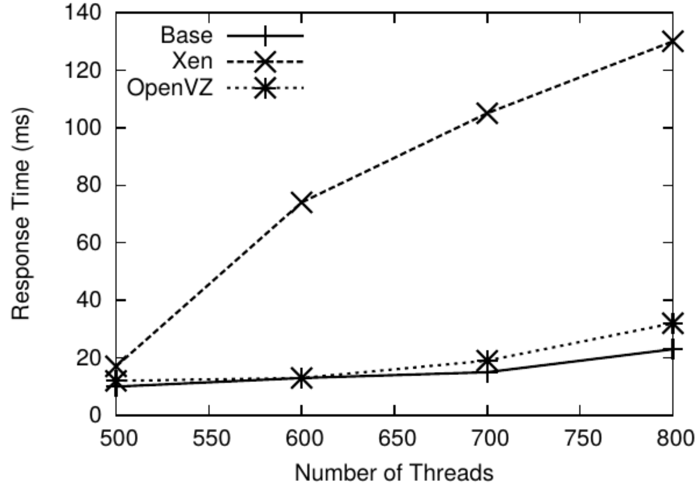


the web and the DB, share a single server

the web and the DB run on two different servers

the web and the DB run on two different servers and each has four instances

Average Response Time single node



(b) Average response time

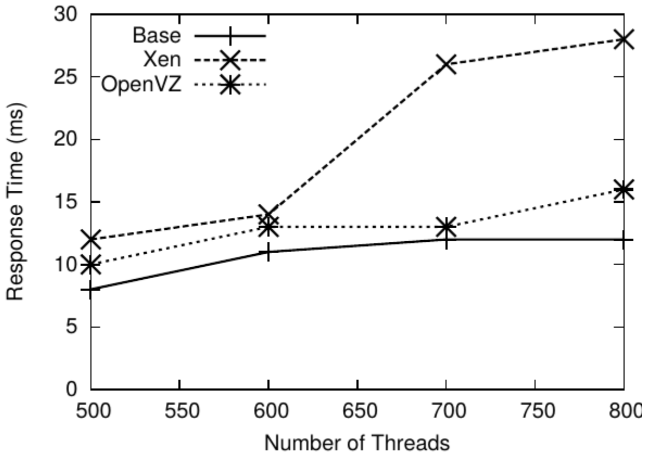
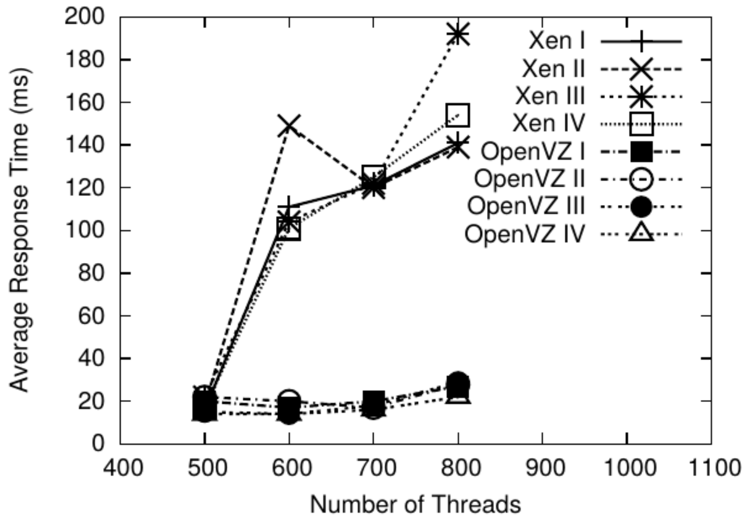


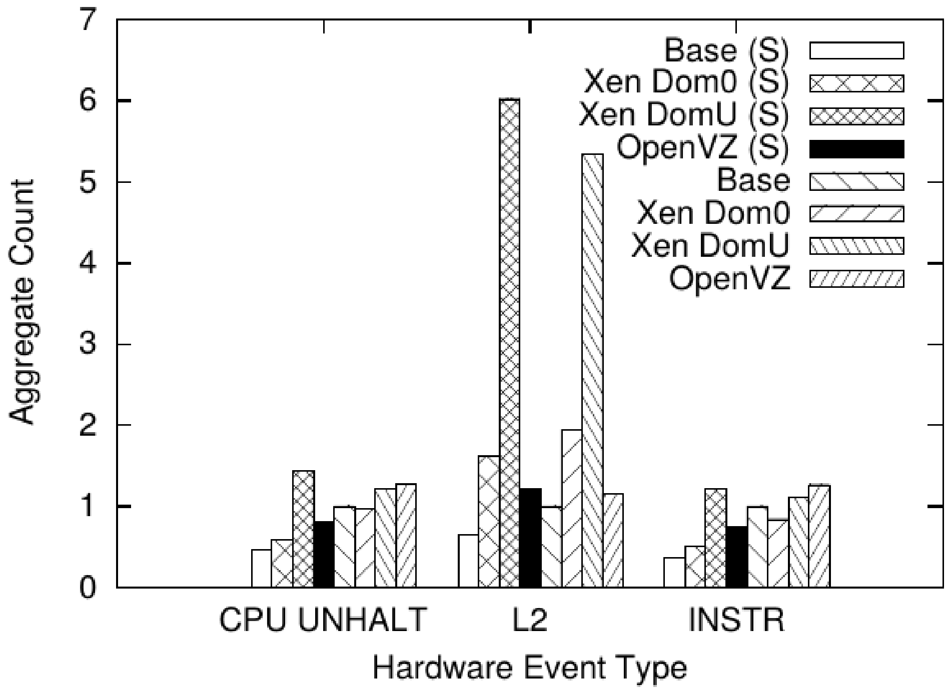
Figure 7: Two-node - average response time



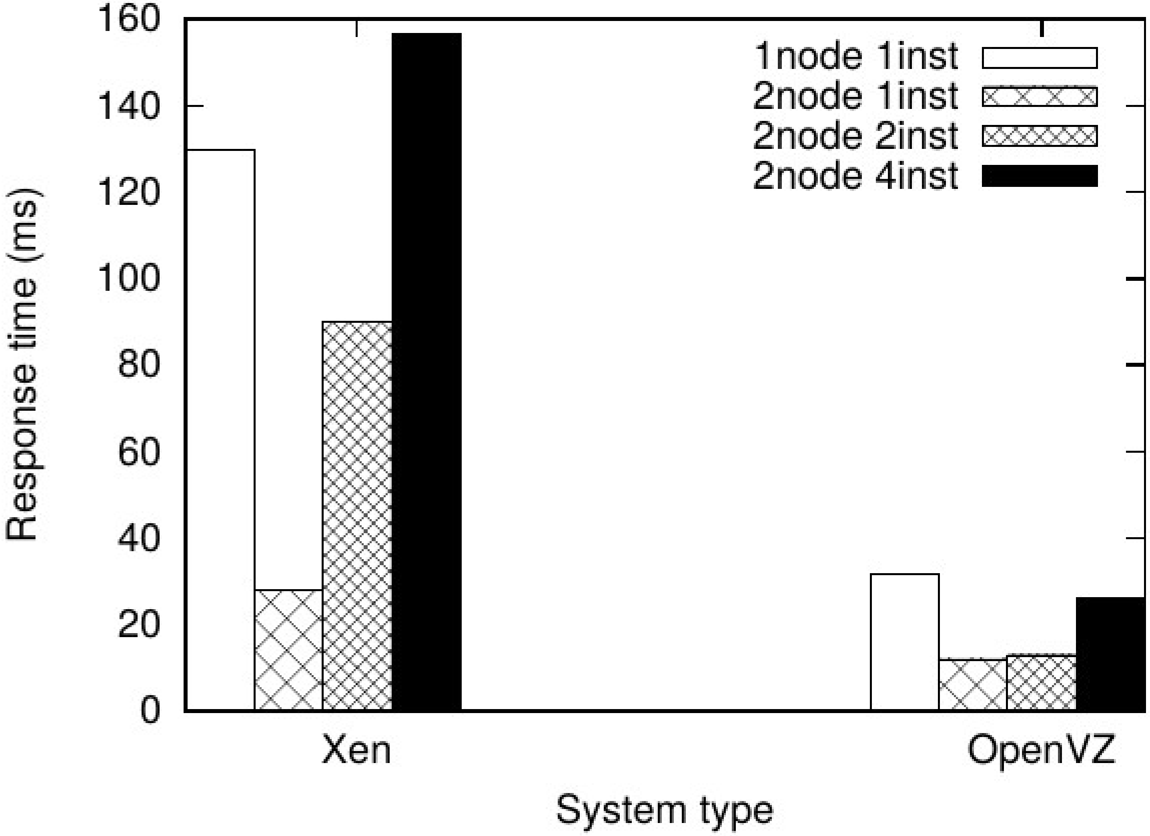
(b) Four instances

Hardware counters

- CPU UNHALT: the CPU time used by a particular binary
- L2: the number of times the memory references in an instruction miss the L2 cache
- INSTR: the number of instructions executed by a binary



Average Response Time Multiple nodes

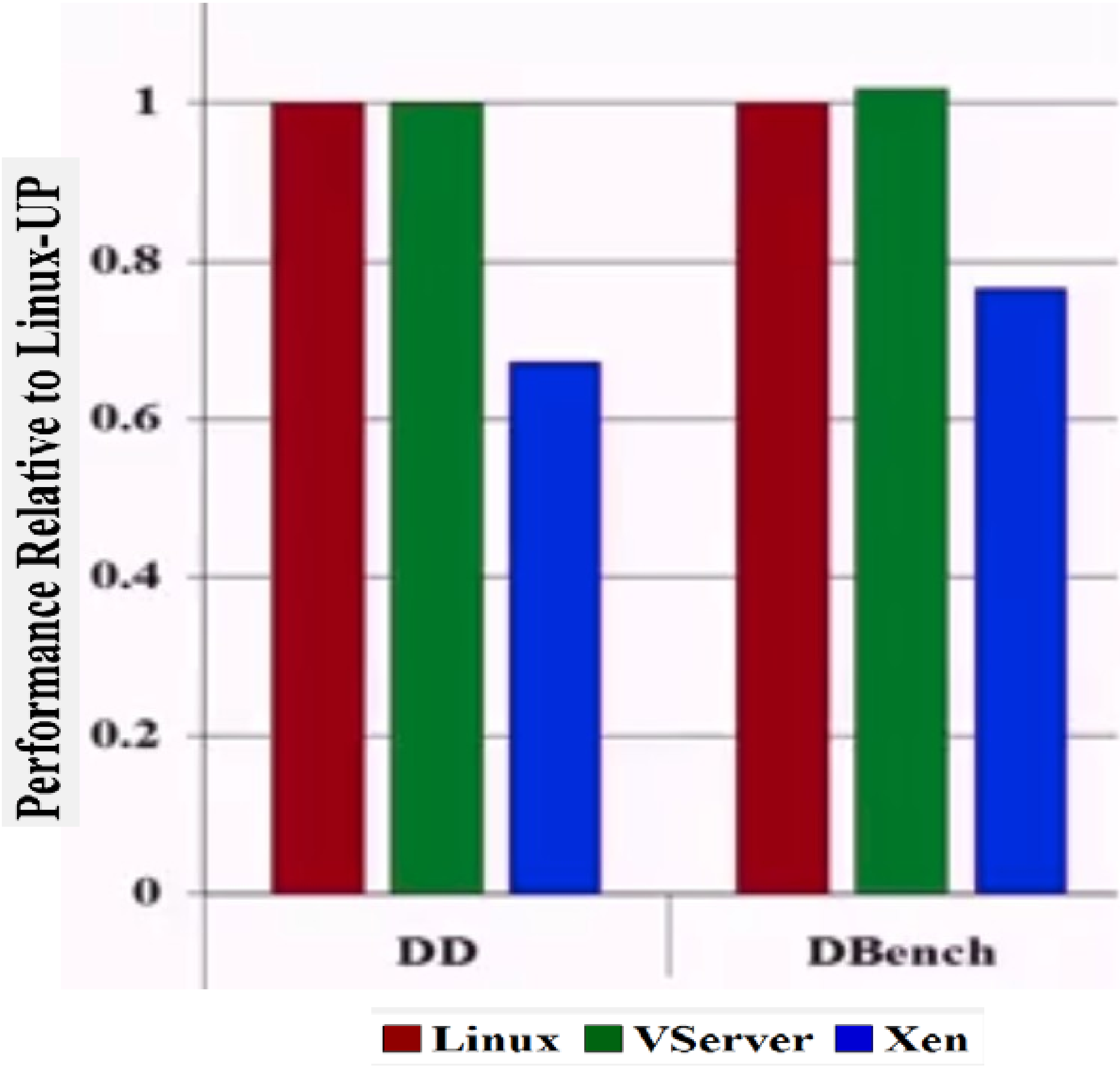


(a) Average response time

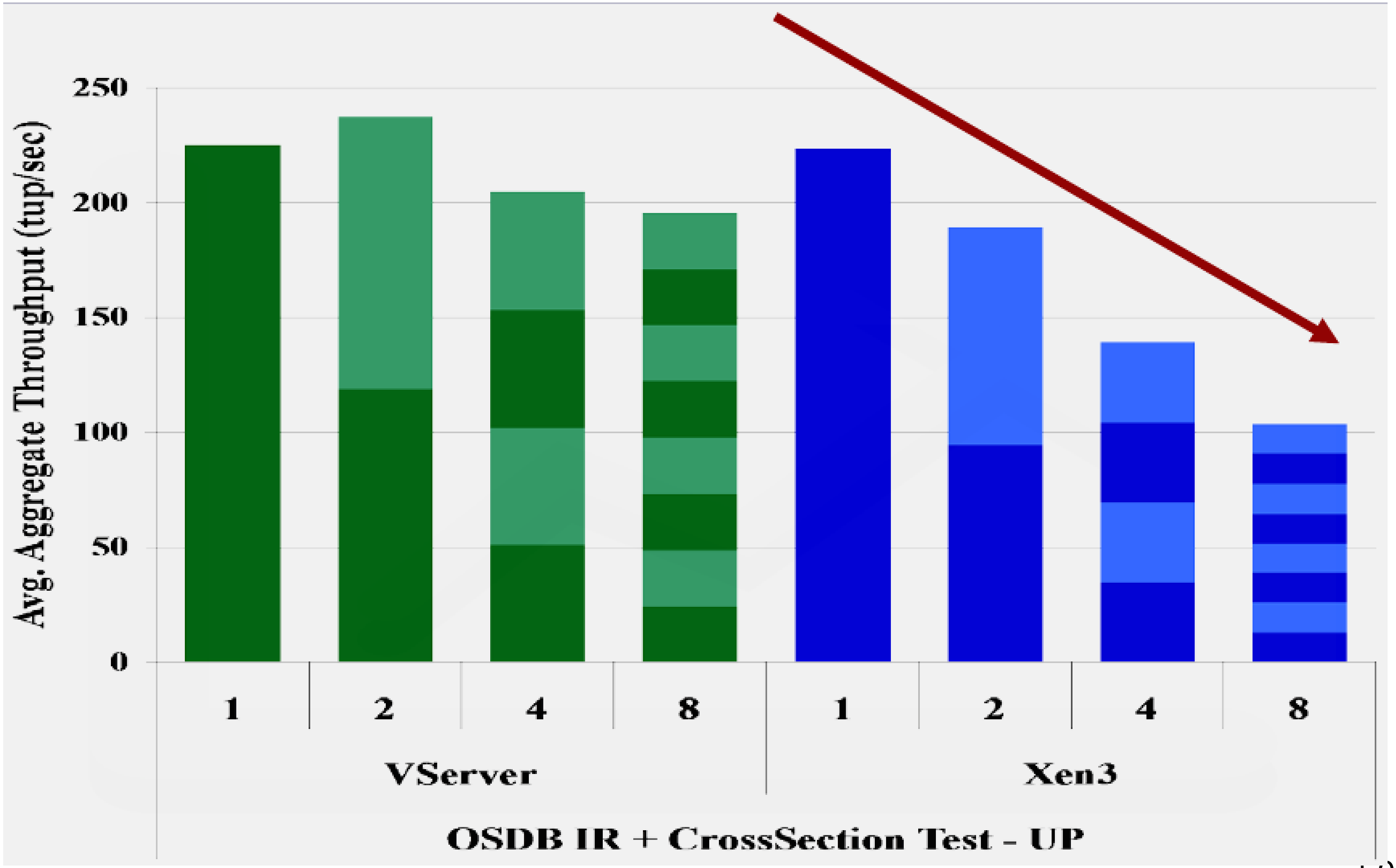
Evaluation results

- The main conclusion drawn from these experiments is that the virtualization overhead of Xen is considerably higher than that of OpenVZ
  - this is due primarily to L2-cache misses.
- Xen performance degradation is noticeable when the workload increases.

I/O performance Comparison



Performance at scale-up



OPEN-SOURCE SOFTWARE PLATFORMS FOR PRIVATE CLOUDS

Private clouds

- Private clouds provide a cost effective alternative for very large organizations
- Schematically, a cloud infrastructure carries out the following steps to run an application:
  - Retrieves the user input from the front-end.
  - Retrieves the disk image of a VM (Virtual Machine) from a repository.
  - Locates a system and requests the hypervisor running on that system to set up a VM.
  - Invokes the DHCP and the IP bridging software to set up a MAC and IP address for the VM.

Opensource tools

- Eucalyptus , OpenNebula, Nimbus, Openstack

Table 10.4 A side-by-side comparison of Eucalyptus, OpenNebula, and Nimbus.

	Eucalyptus	OpenNebula	Nimbus
Design	Emulate EC2	Customizable	Based on Globus
Cloud type	Private	Private	Public/Private
User population	Large	Small	Large
Applications	All	All	Scientific
Customizability	Administrators limited users	Administrators and users	All but image storage and credentials
Internal security	Strict	Loose	Strict
User access	User credentials	User credentials	x509 credentials
Network access	To cluster controller	–	To each compute node

Openstack

- OpenStack is a collection of open source projects that enterprises or service providers can use to set up and run their cloud compute and storage infrastructure