Operating Systems Research

Gaurav Somani

Areas : Systems

Operating systems and their relation with

- File and storage systems - ZFS, Google FS…
- Distributed systems – Amoeba, Plan 9, Grid, Cloud, …
- Mobile systems – Android, Symbian…
- Secure systems
- Embedded systems – TinyOS, RTOS…
- Virtualization – Xen, VirtualBox, OpenNebula…
- Networking/ Device Support – Device Drivers, Protocol Stack……

Introduction to Virtualization Technology

Virtualization - Introduction

Virtualization is the technique to create an efficient and isolated duplicate of a real machine [1].

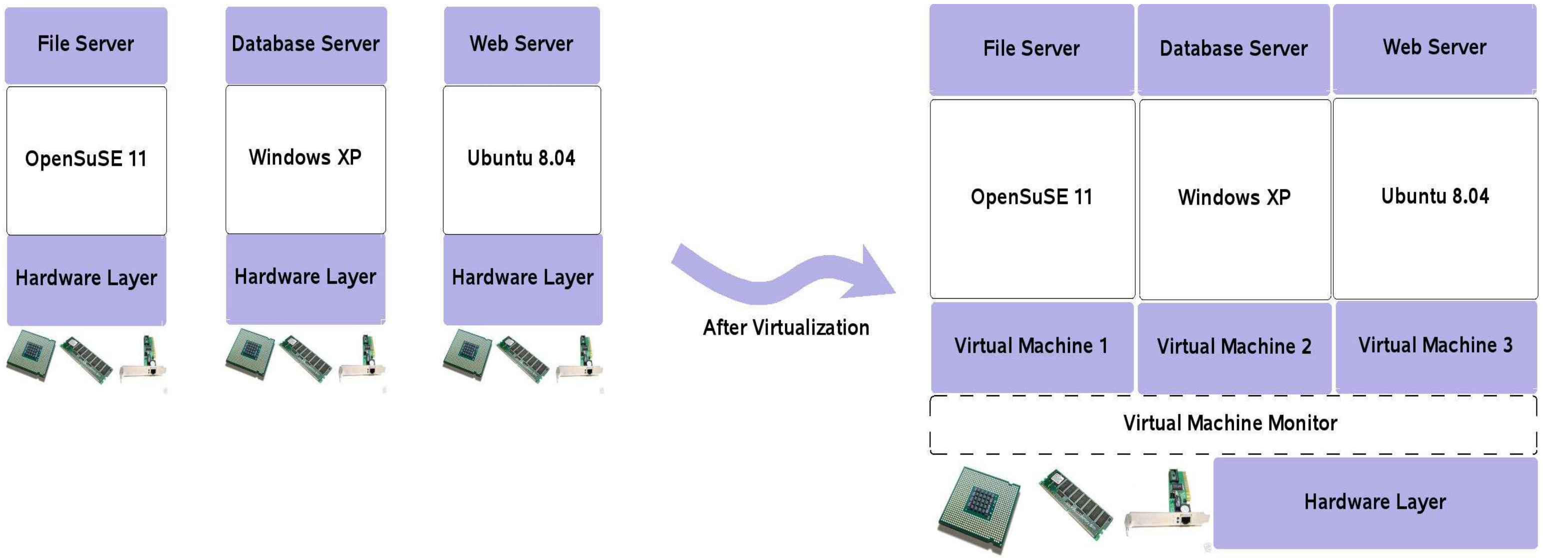


Figure 1. Implementing Virtualization

Why to virtualize?
1. Flexibility
2. Availability
3. Fault Isolation
4. Scalability
5. Hardware Utilization
6. Security
7. Cloning

Why not to virtualize?
1. Virtualization Overhead
2. Single point of failure
3. Real time applications

Virtual Machines

- Virtualize whole system environment :
  - OS, Applications
- Virtual Machine Monitor (VMM) OR Hypervisor:
  - Software layer that provides virtualization support
  - Manages multiple virtual machines
- Virtual Machine:
  - Replica of a machine environment
- Guest OS:
  - OS running inside a VM

Formal Model of Virtual Machines [1]

❑ **Popek, Goldberg 1974**

❑ **VMM characteristics**

VMM should have 3 characteristics:

1. Equivalence: A program running in a VM should behave identically to running in native mode except for performance)
2. Efficiency: Programs running in a VM should show only minor decreases in speed
3. Resource control: VMM should have complete control of system resources

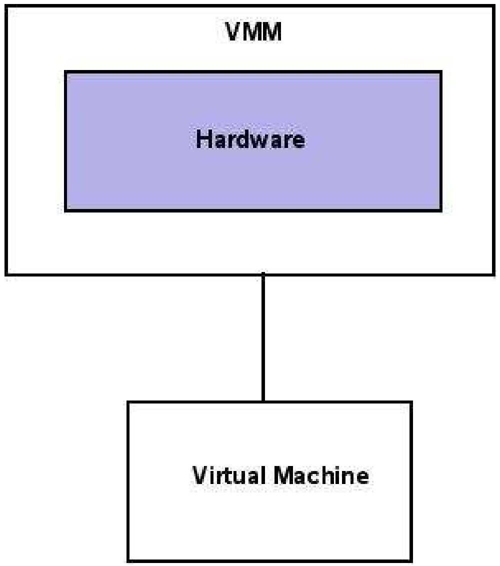


Figure 3 : Formal Model of Virtual Machine Monitor [1]

Virtualization Theory

- Describes the requirements for an architecture to be virtualizable
- Types of instructions:
  - Sensitive: Change system state (e.g.: resource allocation, protected data, etc.)
  - Innocuous: Regular instructions
  - Privileged instructions: Can be executed only in kernel mode (Trap in user mode)

Virtualization Theory (contd.)

- **Theorem:** An architecture can be virtualized if sensitive instructions are a subset of privileged instructions
- **Intuition:** VMM can capture all sensitive instructions
- **Efficiency:** All innocuous instructions should be executed natively
- Many architectures (e.g.: Intel x86) do not satisfy this theorem, E.g.: POPF instruction for setting interrupt flag

Resource Virtualization

We need to virtualize:

- CPU
- Memory
- I/O devices
- Network

Processor Virtualization

- VMM runs in privileged mode
- Guest OS and applications run in non-privileged mode
- VMM time slices between different VMs similar to process time slicing
- Whole VM state preserved on switching
- Need to trap sensitive instructions
- Dynamic binary translation: Patch sensitive instructions to trap into the VMM
- Hypervisor: Allows explicit *hyper-calls*

Interrupt Handling

- Interrupts managed by VMM
- Guest OS disables interrupts => VMM queues up subsequent interrupts
- Guest enables interrupts => deliver queued interrupts
- VMM traps special instructions, e.g., POPF, so that VM sees disabled/enabled interrupts
- Timer interrupts are handled by VMM

Virtual machine Scheduling

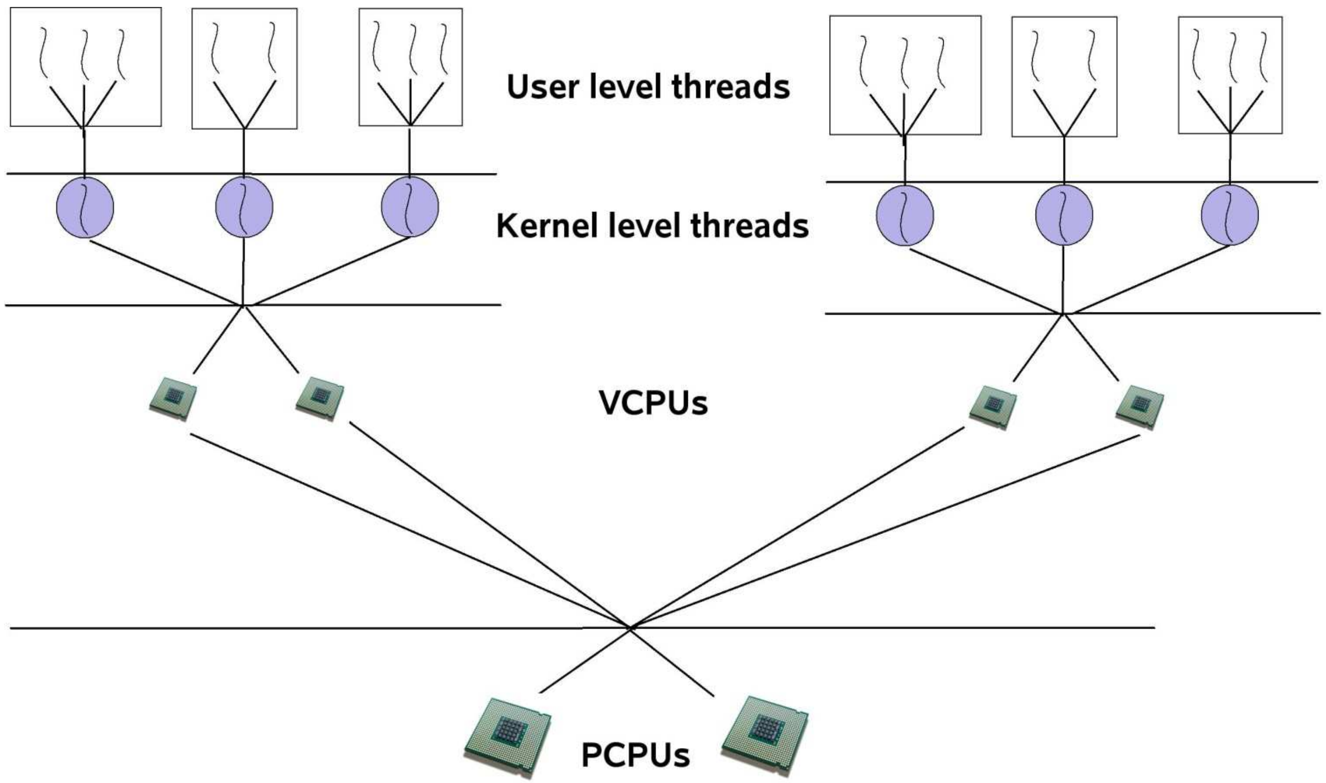


Figure 7 : Scheduling in Xen (based on [4])

Three Tiers of schedulers

- ❑ User space threads to kernel level threads in guests
- ❑ Guest kernel mapping threads to VCPUs
- ❑ VCPUs to Physical processors

Memory Virtualization

- Guest OS sees flat “physical” address space
- Page tables within guest OS:
  - Translate from virtual to physical addresses
- Second-level mapping:
  - Physical addresses to machine addresses
- VMM can swap a VM’s pages to disk transparently

Shadow page tables

- Translate from virtual addresses to machine addresses
- Used directly by MMU
- Unmapped pages transparently handled by VMM
- If guest tries to modify page table, control passed to VMM

I/O Virtualization

Problems:

- Different devices have different characteristics
- Large number of devices

Techniques:

- VMM virtualizes devices, translates into native device I/O
- VMM gives access to I/O devices, but controls access

Host-based I/O

- VMM uses device drivers in the host OS
- Guest I/O commands converted to host I/O system calls
- E.g.: virtual disk is mapped to a file on host
- Disk reads/writes become file reads/writes

Virtual Network

- Each VM is assigned a separate IP address
- Communication among VMs: No physical networking required
- How do machines across the network talk to a specific VM?
- Physical network device in promiscuous mode => Listens to all packets, picks up VM-specific packets

Kinds of Virtualization

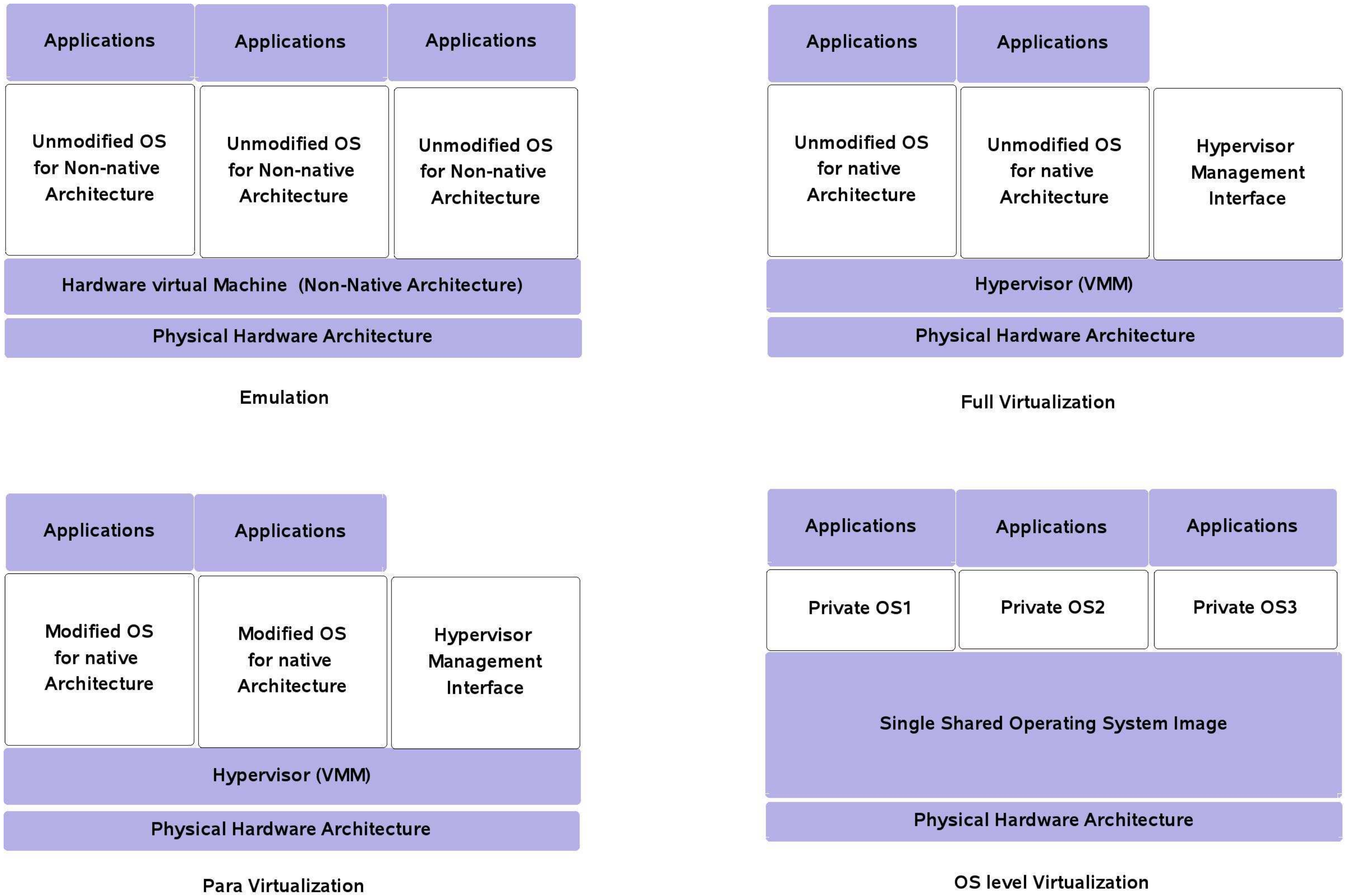
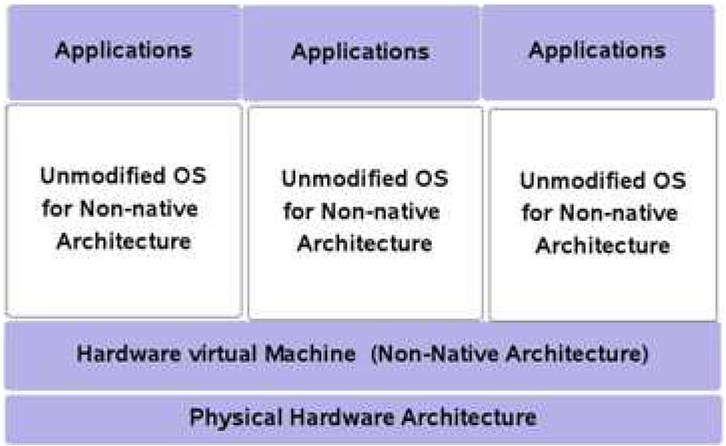


Figure 4: Kinds of Virtualization [2]

Emulation

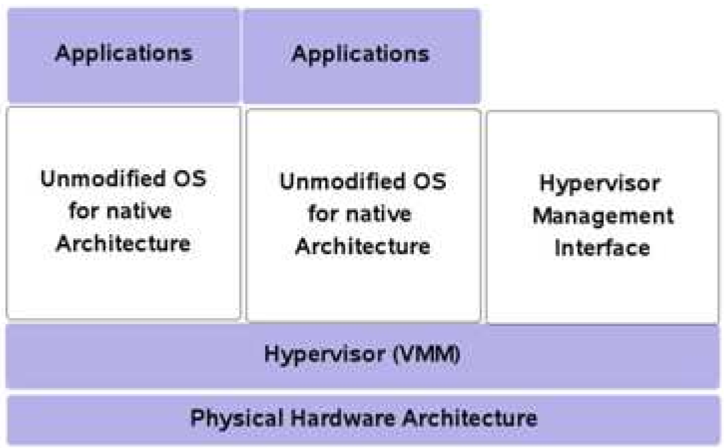
- An **emulator** duplicates the functions of one system using a different system, so that the second system behaves like the first system
- Processor emulator example – QEMU
- Works on dynamic binary translation.
- http://www.qemu.org



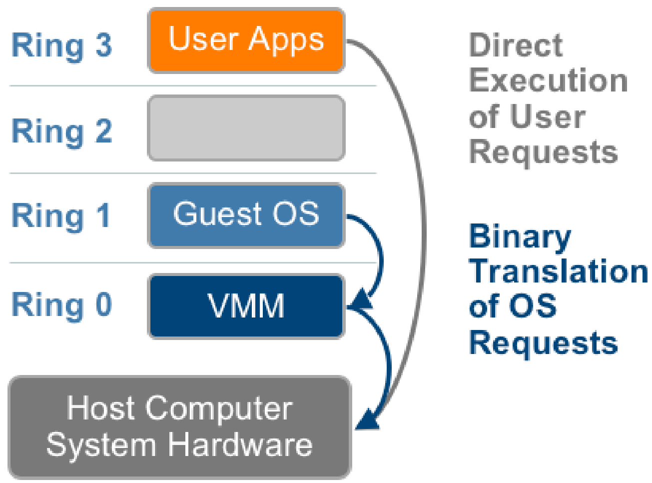
Emulation

Full Virtualization

- Xen and KVM can use full virtualization.
- Full virtualization uses hardware features of the processor to provide total abstraction of the underlying physical system (Bare-metal) and create a new virtual system in which the guest operating systems can run.
- No modifications are needed in the guest operating system.
- The guest operating system and any applications on the guest are not aware of the virtualized environment and run normally.

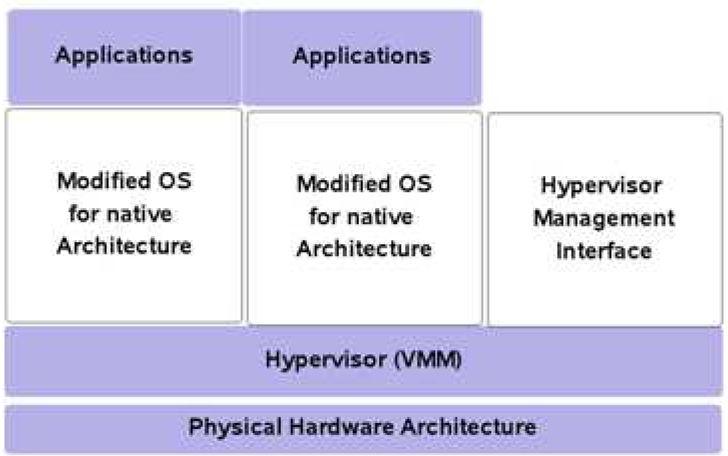


Full Virtualization

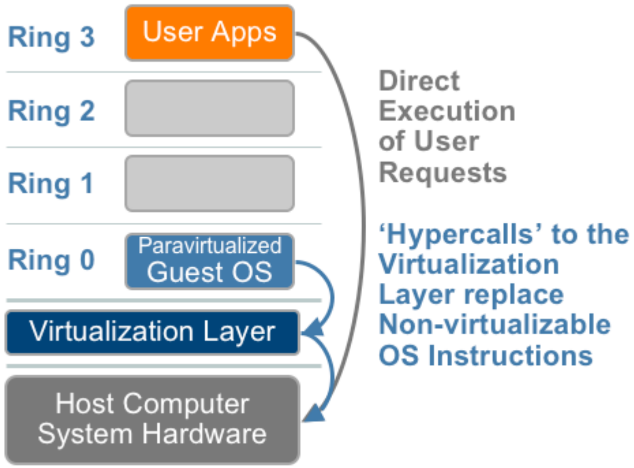


Para Virtualization

- It presents a software interface to virtual machines that is similar but not identical to that of the underlying hardware.
- The intent of the modified interface is to reduce the portion of the guest's execution time spent performing operations which are substantially more difficult to run in a virtual environment compared to a non-virtualized environment.
- The paravirtualization provides specially defined 'hooks' (Hypercalls) to allow the guest(s) and host to request and acknowledge these tasks
- Paravirtualization requires the guest operating system to be explicitly ported. (Modified)
- Example - Xen



Para Virtualization



Xen Virtual machine monitor

- Developed at the University of Cambridge Computer Laboratory.
- GNU General Public License (GPL)
- The x86 architecture support with guest OS modification
- Live migration of running virtual machines between physical hosts.
- Intel and AMD Virtualization Technology for unmodified guest operating systems
- Xen host kernel code runs in ring 0, while the hosted domains run in ring1 or Ring 3.

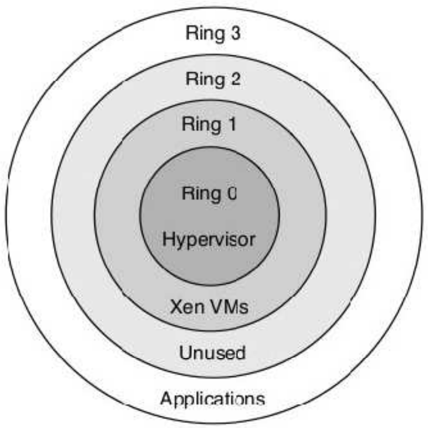


Figure 5 : Privilege Rings in x86 [3][7]

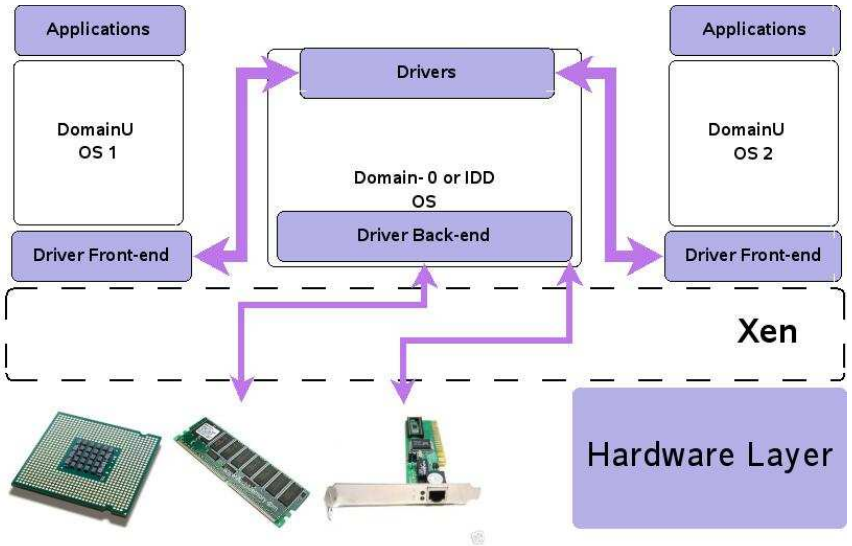
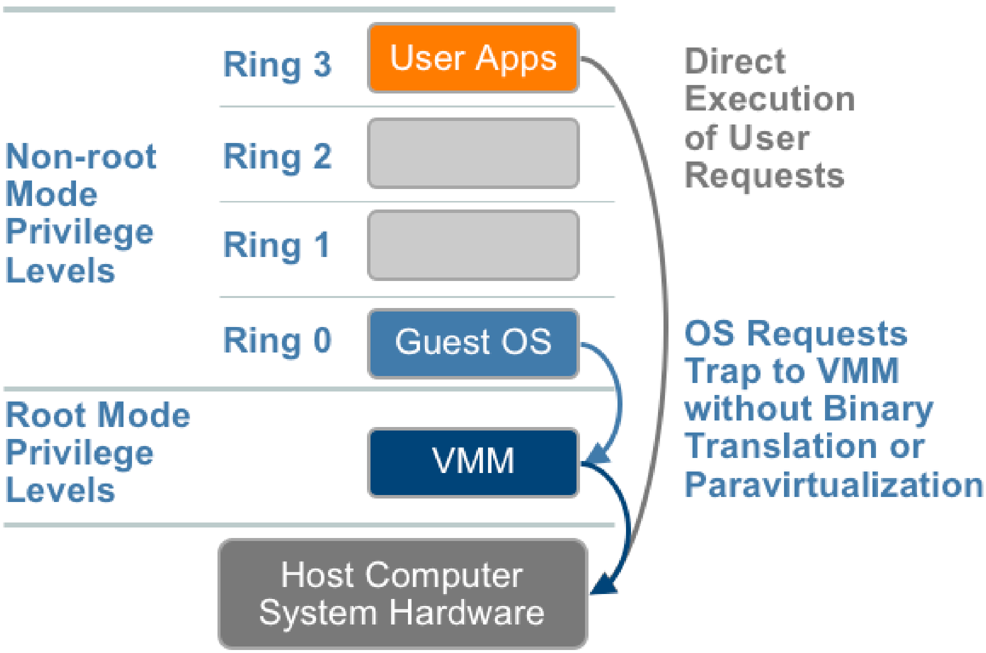


Figure 6 : Xen architecture based on [3][7]

Hardware Assisted Virtualization

- Support from Intel Virtualization Technology (VT-x) and AMD's AMD-V
- Both target privileged instructions with a new CPU execution mode feature that allows the VMM to run in a new root mode below ring 0
- Xen, VMWare



Architectural Support: Intel VT-x

Two modes:

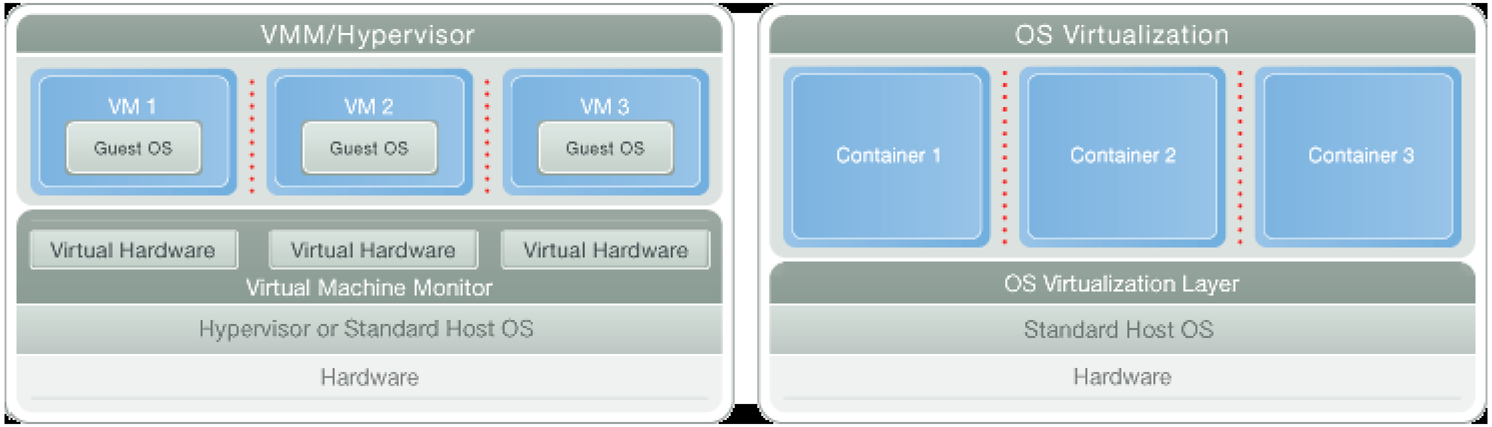
– Root (VMM)

– Non-root (Guest OS)

- Transition operations:
  – VM entry, VM exit
  – Transitions occur for sensitive instructions
- VM control structure
  – Stores VM state (e.g., registers)
  – Similar to process control block

OS level/Shared OS Virtualization

- Shared OS image
- Isolation, abstraction and security
- Examples - Parallels, Virtuozzo



Virtualization in cloud

”A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements (SLA) established through negotiation between the service provider and consumers. [7]”

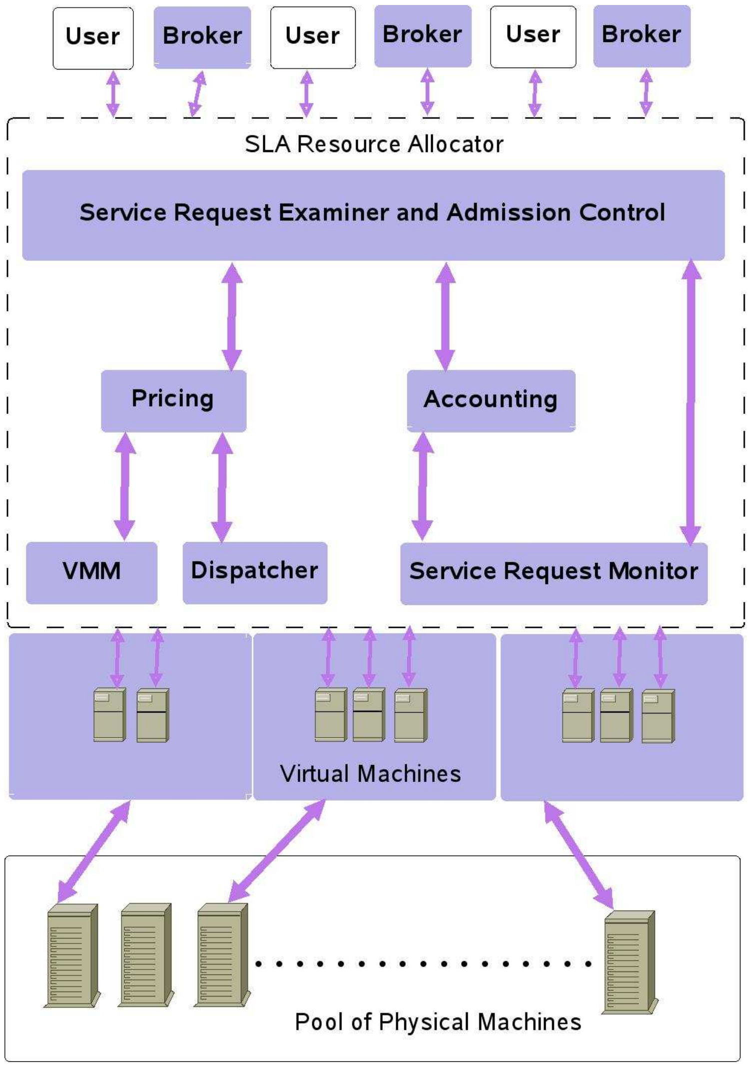
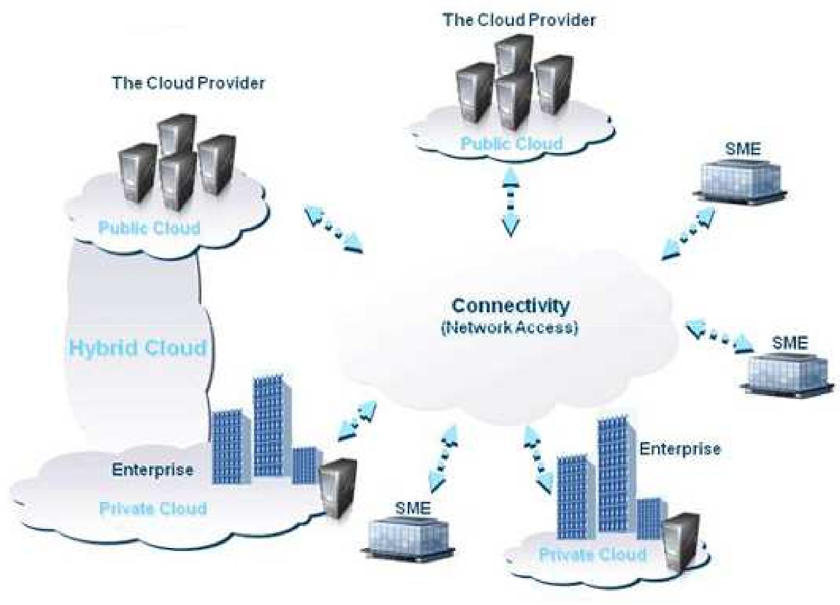
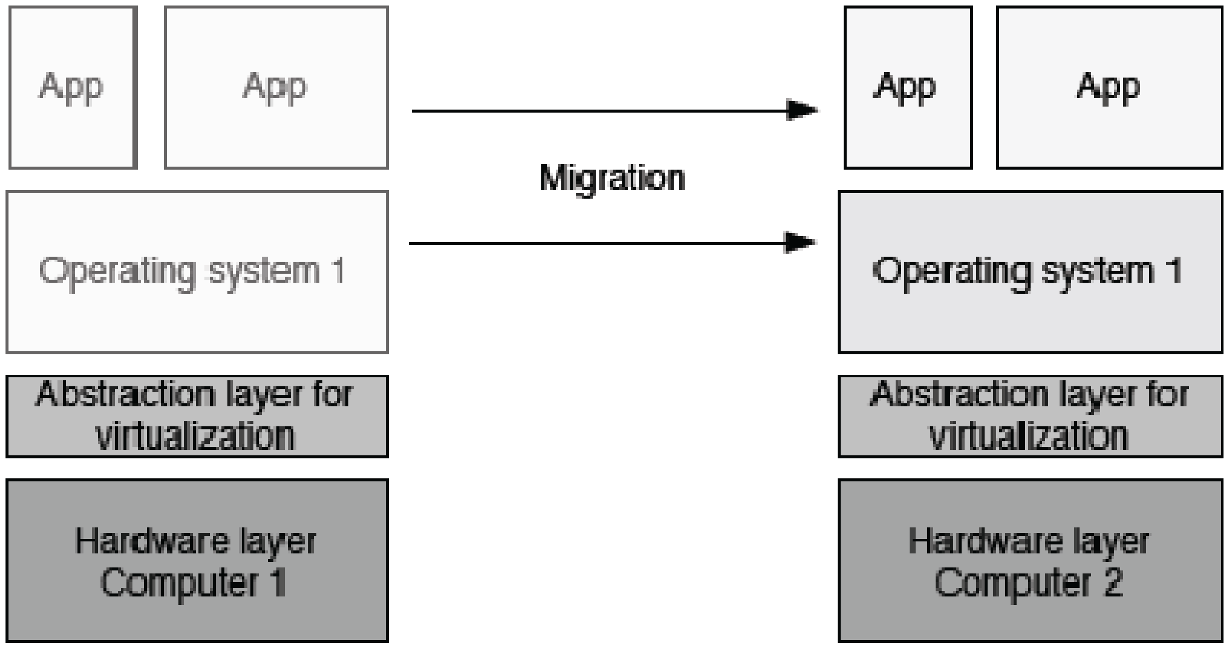


Figure 2 :Architecture of cloud computing[7]

Public/Private/Hybrid Clouds



VM Migration



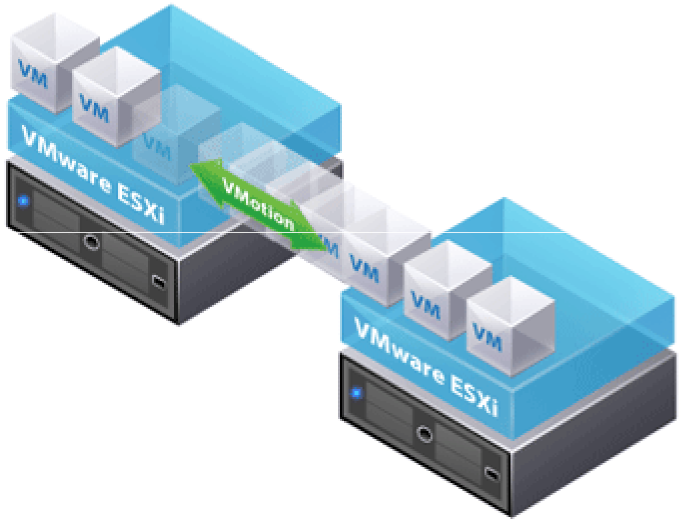
Motivation : VM Migration

- Load management
- Maintenance of original host
- Why VM Migration?
  - Avoid difficulties of process-level migration approaches, such as residual dependencies
  - In-memory state can be transferred in a consistent and efficient fashion. We can migrate an on-lime game server or streaming server without requiring clients to reconnect.
  - Separation of concerns between users and operators. Very powerful tool for cluster administrators. Separate hardware and software considerations, consolidate clustered hardware into a single coherent managed domain

Approaches

- **Static**
  - To provide mobility to users who work on different physical hosts at different times, e.g. transfer OS from work to home while on the subway.
  - Optimize for slow links and longer time spans, stops OS execution for the duration of the transfer, with a set of enhancements to reduce the transmitted image size.
- **Live Migration**
  - Move a running virtual machine or application between different physical machines without disconnecting the client or application.
  - Requires shared storage provided by SAN or NAS.

Live Migration [13]



Design Issues

- Migrating Memory
  - How to move while minimizing downtime and total migration time?
- Delta Copying
- Don't disrupt active services through resource contention by migrating the OS.
- Local Resources
  - What to do with resources associated with the physical machine when they are migrating away from?
    - Memory
    - Connections to local devices such as disks and network interfaces

Design Overview

- Stage 0: Pre-Migration
- Stage 1: Reservation
- Stage 2: Iterative Pre-Copy
- Stage 3: Stop-and-Copy
- Stage 4: Commitment
- Stage 5: Activation

Where does innovation lie?

- Resource Allocation
- VM Placement
- VM Configuration
- SLA fulfillment
- Migration over internet/public clouds

References

[1] Gerald J. Popek and Robert P. Goldberg, “Formal Requirements for Virtualizable Third generation Architectures, *Communications of ACM*, Volume 17, Number 7, 1974.

[2] Jeanna N. Matthews, Eli M. Dow, Todd Deshane, Wenjin Hu, Jeremy Bongio, Patrick F. Wilbur, and Brendan Johnson. *Running Xen: A Hands-On Guide to the Art of Virtualization. Prentice Hall PTR*, Upper Saddle River, NJ, USA, 2008.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *In SOSP ’03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[4] David Chisnall. *The Definitive Guide to the Xen Hypervisor (Prentice Hall Open Source Software Development Series).* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[5] Ludmila Cherkasova and Diwaker Gupta and Amin Vahdat, “Comparison of the three CPU schedulers in Xen, *ACM SIGMETRICS Perform. Eval. Rev*., Volume 35, Number 2, 2007.

[6] Diego Ongaro, Alan L. Cox, and Scott Rixner. Scheduling i/o in virtual machine monitors. *In VEE ’08: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 1–10, New York, NY, USA, 2008. ACM.

[7] Paul R. Barham, Boris Dragovic, Keir A. Fraser, Steven M. Hand, Timothy L. Harris, Alex C. Ho, Evangelos Kotsovinos, Anil V.S. Madhavapeddy, Rolf Neugebauer, Ian A. Pratt, Andrew K. Wareld, “Xen 2002”, *Technical Report, University of Cambridge Computer Laboratory*, 2003.

[8] Jeanna Neefe Matthews, Wenjin Hu, Madhujith Hapuarachchi, Todd Deshane, Demetrios Dimatos, Gary Hamilton, Michael McCabe, and James Owens. Quantifying the performance isolation properties of virtualization systems. *In ExpCS ’07: Proceedings of the 2007 workshop on Experimental computer science*, page 6, New York, NY, USA, 2007. ACM.

References

[9]Ludmila Cherkasova and Rob Gardner. Measuring cpu overhead for i/o processing in the Xen virtual machine monitor. *In ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference,* pages 24– 24, Berkeley, CA, USA, 2005. USENIX Association.

[10] Yubin Xia; Yan Niu; Yansong Zheng; Ning Jia; Chun Yang; Xu Cheng. Analysis and enhancement for interactive- oriented virtual machine scheduling. *In Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, pages pp.393–398, 17-20 Dec. 2008.

[11] Dongsung Kim, Hwanju Kim, Myeongjae Jeon, Euiseong Seo, and Joonwon Lee. Guest- aware priority-based virtual machine scheduling for highly consolidated server. *In Euro-Par'08: Proceedings of the 14th international Euro-Par conference on Parallel Processing*, pages 285–294, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] Bin Lin and Peter A. Dinda. Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. *In SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 8, Washington, DC, USA, 2005. IEEE Computer Society.

[13] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. 2005. Live migration of virtual machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2* (May 02 - 04, 2005). USENIX Association, Berkeley, CA, 273-286.

[14] VMWare White papers "Understanding *Full* Virtualization, *Paravirtualization,* and Hardware Assist" , http://www.vmware.com/ /files/pdf/VMware_paravirtualization.pdf