Lecture 2 : Resource Virtualization Basic Principles

Dr. Bibhas Ghoshal

Assistant Professor

Department of Information Technology

Indian Institute of Information Technology Allahabad

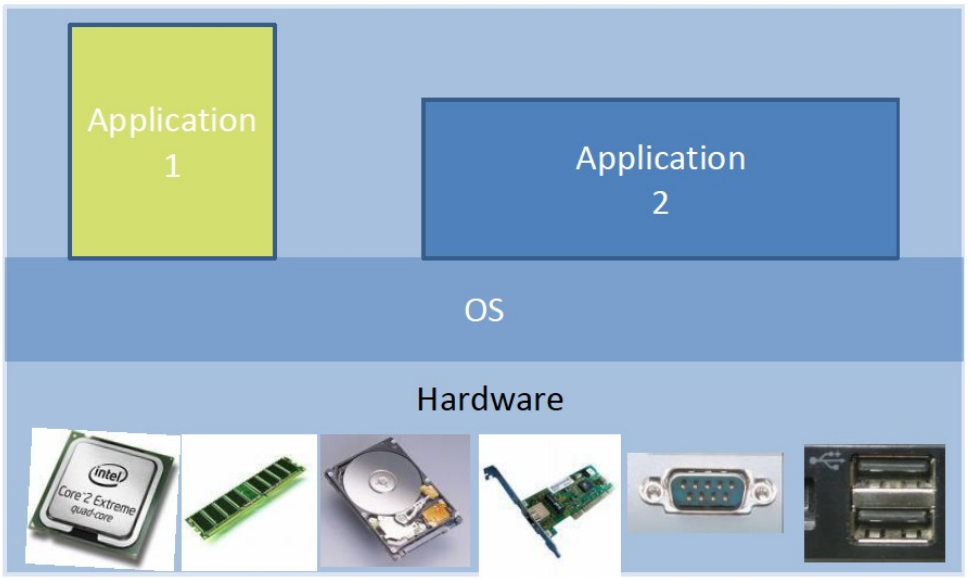
Virtualization : The Basic Principle of Cloud Computing

Virtualization - A computer architecture technology by which multiple virtual machines are multiplexed in the same hardware machine

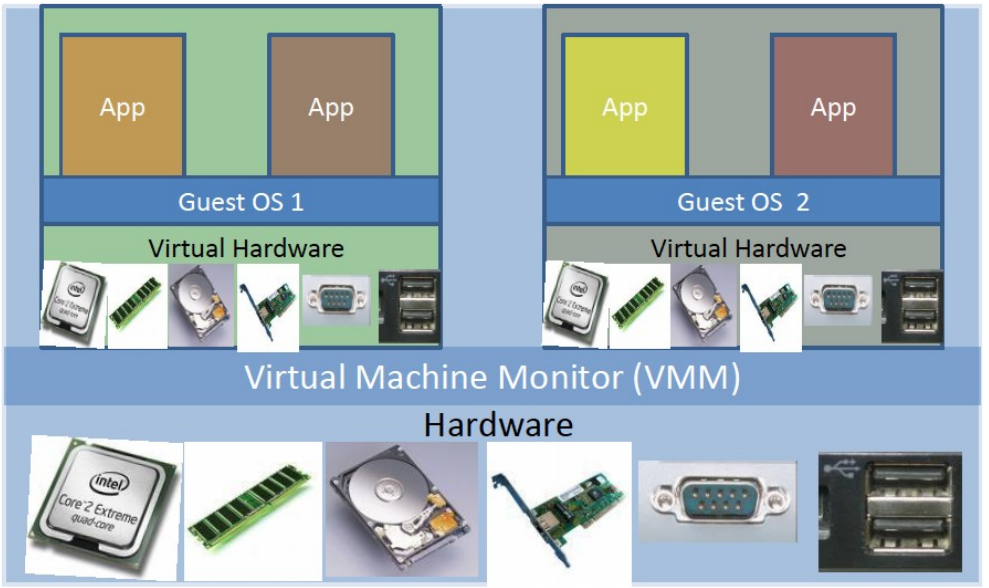
Emulate a physical machine in hardware

1. Provide a hardware-like view to each process
2. Run an OS inside another OS
3. Run multiple OS sessions on a single physical hardware

Virtualization



Traditional Computing



Virtualized Computing

Virtualization

- Simulates the interface to a physical object by:
- Multiplexing: creates multiple virtual objects from one instance of a physical object.

  Example - a processor is multiplexed among a number of processes or threads.
- Aggregation: creates one virtual object from multiple physical objects.
  Example - a number of physical disks are aggregated into a RAID disk.
- Emulation: constructs a virtual object from a different type of a physical object.

  Example - a physical disk emulates a Random Access Memory (RAM).
- Multiplexing and emulation.

  Example - virtual memory with paging multiplexes real memory and disk; a virtual address emulates a real address.

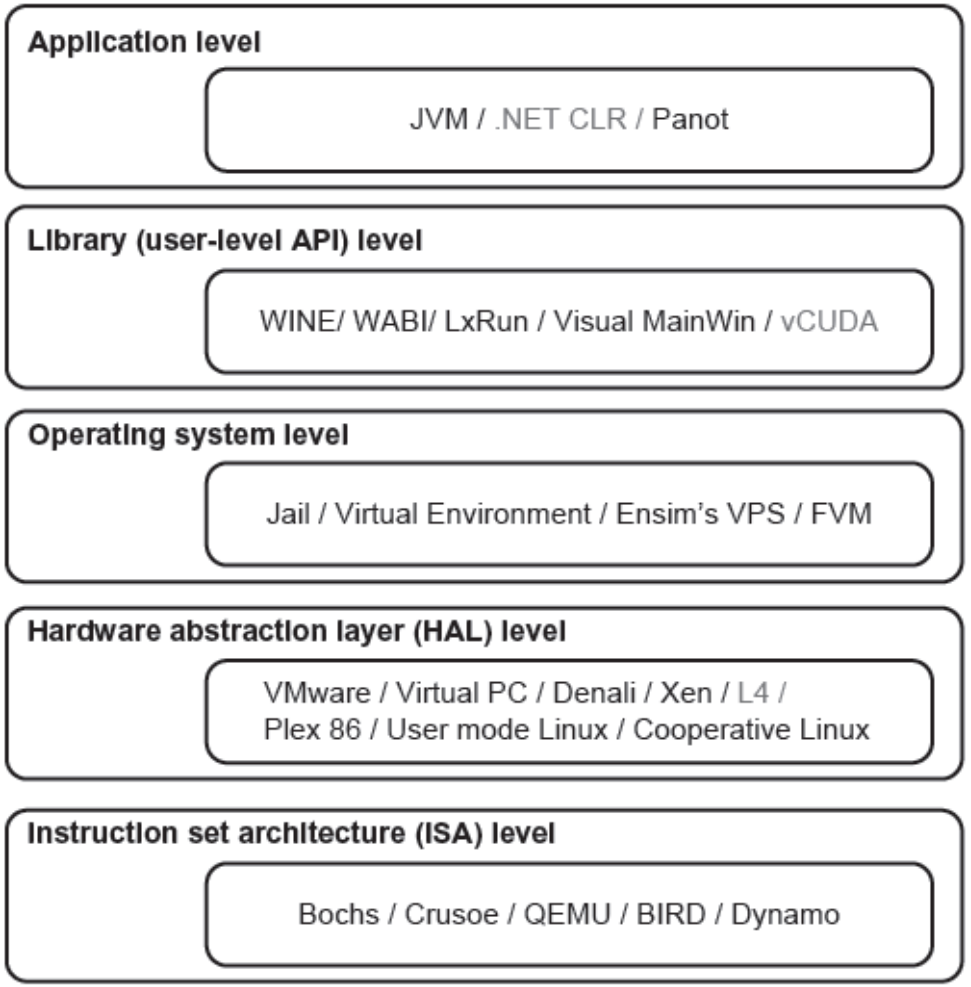
Virtualization

Advantages and Disadvantages :

1. Resource sharing- improve computer performance in terms of resource utilization and application flexibility
2. Performance - allows apps to migrate from one platform to another
3. Ease of use - allows users to operate in familiar environment
4. System Security- allows isolation of services running on the same hardware
5. Development and management of services offered by the provider -
6. Complete isolation among applications - Ex : Mail server VM and Print server VM (Server)
7. Encapsulation - Virtual Machine is a file
8. The state of the Virtual Machine running under Virtual Machine Monitor can be saved and migrated to another server balance the load

Adds overhead and increases execution time. Hypervisor is invoked by the OS when apps make system calls

Implementation Levels of Virtualization



Virtualization Details

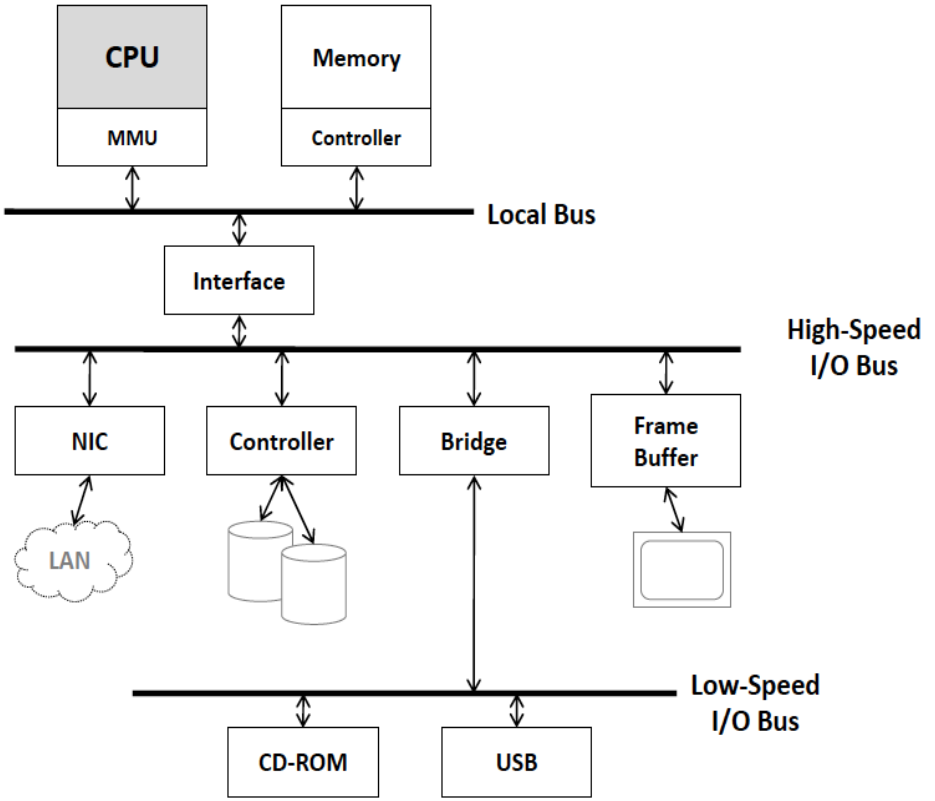
Computer System Organization

Instruction Set Architecture (ISA) :

State visible to programmer
(registers and memory)

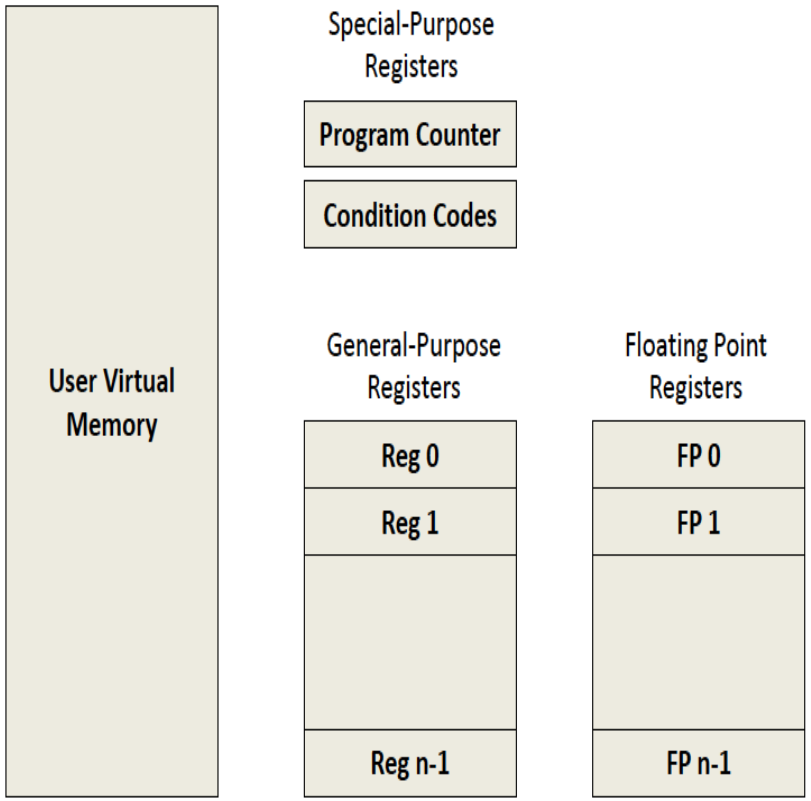
User ISA – Primarily for computation

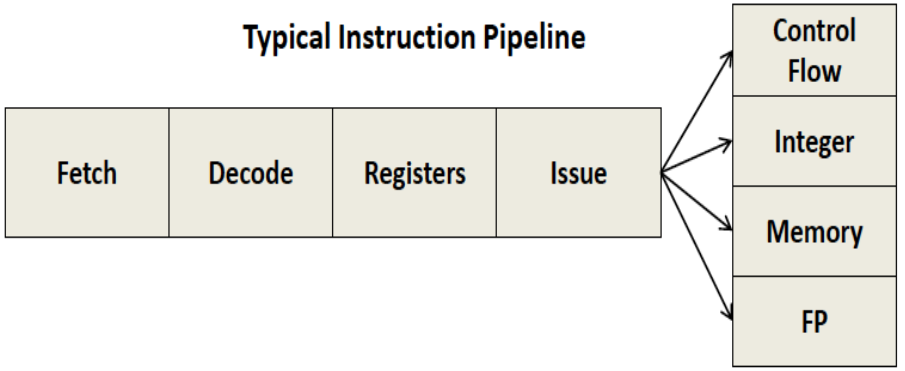
System ISA – For Resource Management



Slide Author : Scott Devine

User ISA State and Instructions





Integer	Memory	Control Flow	Floating Point
Add Sub And Compare ...	Load byte Load Word Store Multiple Push ...	Jump Jump equal Call Return ...	Add single Mult. double Sqrt double ...

Instruction Groupings

Slide Author : Scott Devine

Computer System Organization

Privilege levels

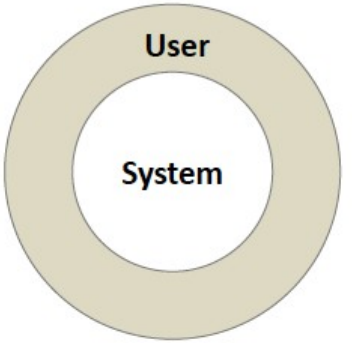
Control registers

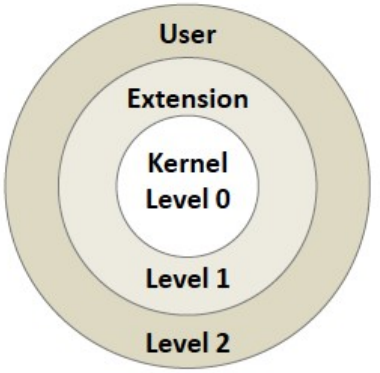
Traps and Interrupts
- hardcoded vectors
- interrupt tables

System Clock

MMU
- Page Tables
- TLB

I/O Device Access





Slide Author : Scott Devine

Computer Instructions

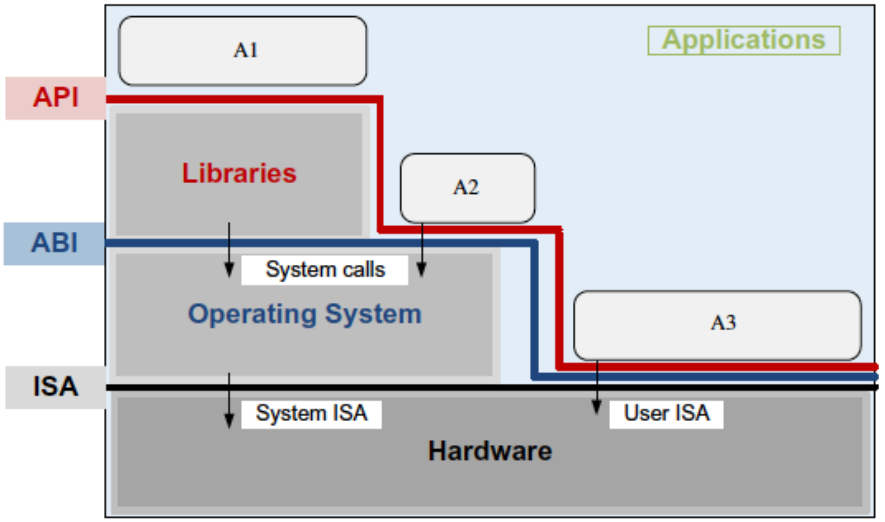
Two classes of machine instructions :

1. **Sensitive** - require special precautions at execution time
2. **Innocuous** - not sensitive

- Control sensitive - instructions that attempt to change either the memory allocation or the privileged mode.

- Mode sensitive - instructions whose behavior is different in the privileged mode.

Layering



API : Application Programming Interface

ABI : Application Binary Interface

ISA : Instruction Set Architecture

A1 : Application uses Library Functions

A2 : Makes System Calls

A3 : Executes Machine Instructions

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu, 1st edition, 2013

Virtual Machine (VM)

- VM - Isolated environment that appears to be a whole computer, but actually only has access to a portion of the computer resources
- Each VM appears to be running on the bare hardware
  - Appearing as multiple instances of the same hardware
  - Events occurring in one VM do not affect other VM
- Process VM- a virtual platform created for an individual process and destroyed once the process terminates
- System VM - supports an OS along with many user processes
- Hypervisor (Virtual Machine Monitor) - A software that partitions resources of computer system into one or more virtual machines

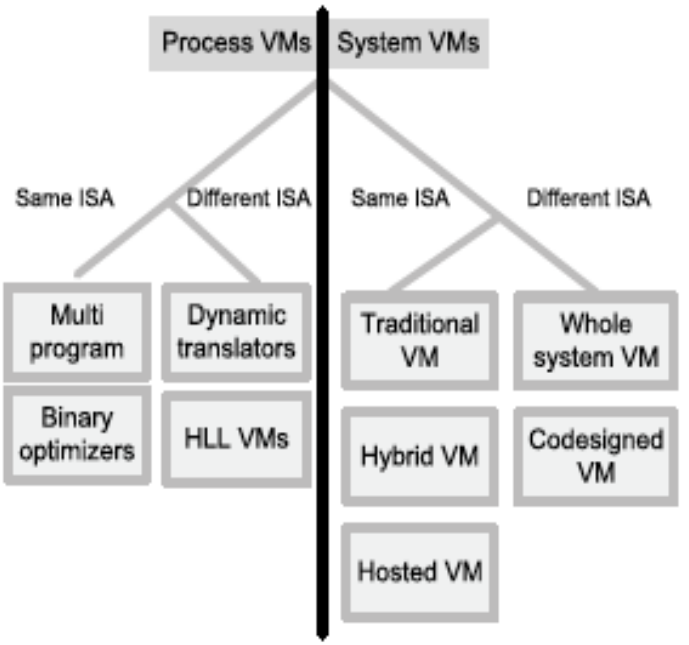
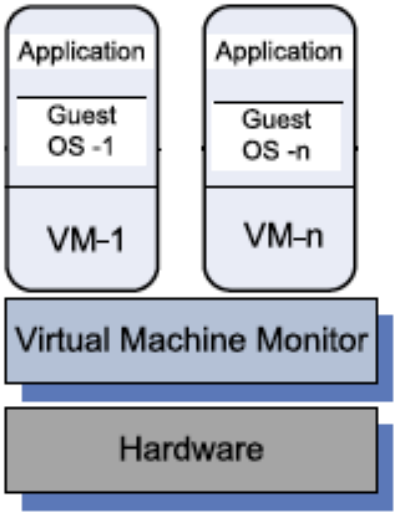
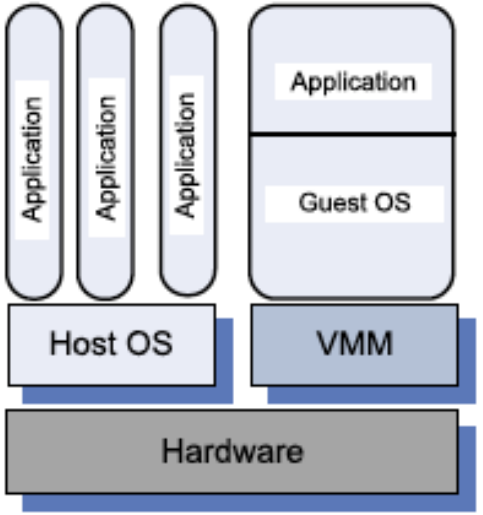


Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu, 2nd edition, 2018

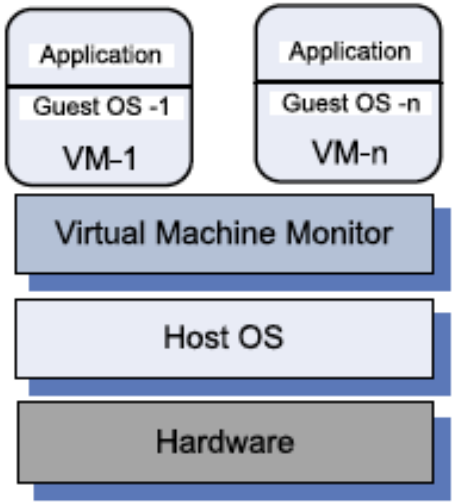
Classes of VM with same ISA



Traditional VM



Hybrid VM



Hosted VM

- Traditional VM - supports multiple virtual machines and runs directly on the hardware.
- Hybrid VM - shares the hardware with a host operating system and supports multiple virtual machines.
- Hosted VM - runs under a host operating system.

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu, 2nd edition, 2018

Virtual Machine Monitor (Hypervisor)

Software that partitions the resources of the computer system into one or more virtual machines. It enables :

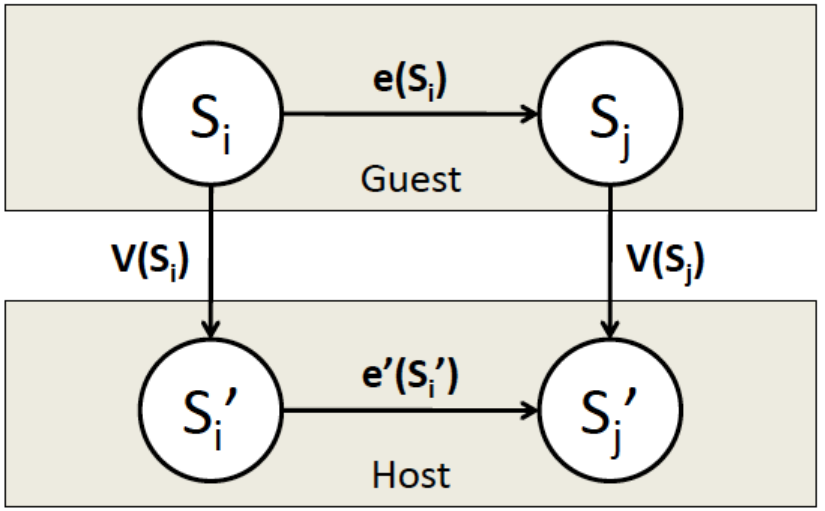
- Allows several OS to run concurrently on same hardware at the same time
- Multiple services to share the same platform
- Movement of a server from one platform to another
- System modification while maintaining backward compatibility with original system

Guest OS : OS that runs under the control of the VMM rather than directly on H/W

VMM runs in kernel mode Guest OS runs in user mode

Virtualization Techniques

- System ISA Virtualization
- Instruction Interpretation
- Trap and Emulate
- Binary Translation
- Hybrid Models



Formally, virtualization involves construction of isomorphism from guest state to host state

Slide Author : Scott Devine

Virtualization Techniques

- Interpretation
  - Interpret each CPU instruction and emulate it
    - Ex : each instruction is emulated by a C function
    - Slow down – 50 x

```
incl (%eax):
    » r = regs[EAX];
    » tmp = read_mem(r);
    » tmp++;
    » write_mem(r, tmp);
```

- Binary Translation
  - Translate each guest instruction into minimal set of host instructions required to emulate it

```
incl (%eax)
    » leal mem0(%eax), %esi
    » incl (%esi)
```

  - **Advantages :**
    - Avoid function-call overhead of interpreter based approach
    - Can reuse translations by maintaining a translation cache
    - Slow down 5-10 x

Virtualization of ISA: CPU State

- Process normal instructions as fast as possible
- Forward privileged instructions to emulation routines

```
static struct {
   uint32   GPR[16];
   uint32   LR;
   uint32   PC;
   int      IE;
   int      IRQ;
} CPUState;
```

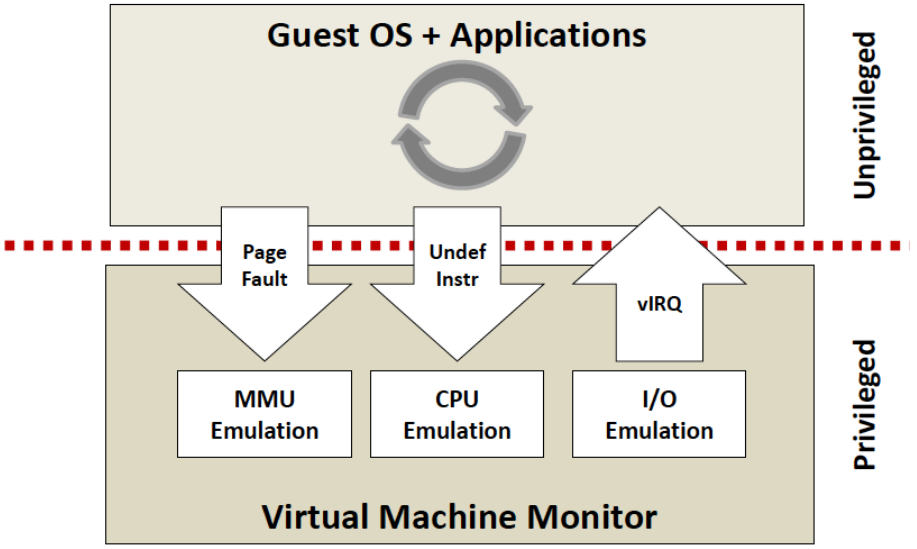
```
void CPU_CLI(void)
{
   CPUState.IE = 0;
}

void CPU_STI(void)
{
   CPUState.IE = 1;
}
```

Slide Author : Scott Devine

Trap And Emulate

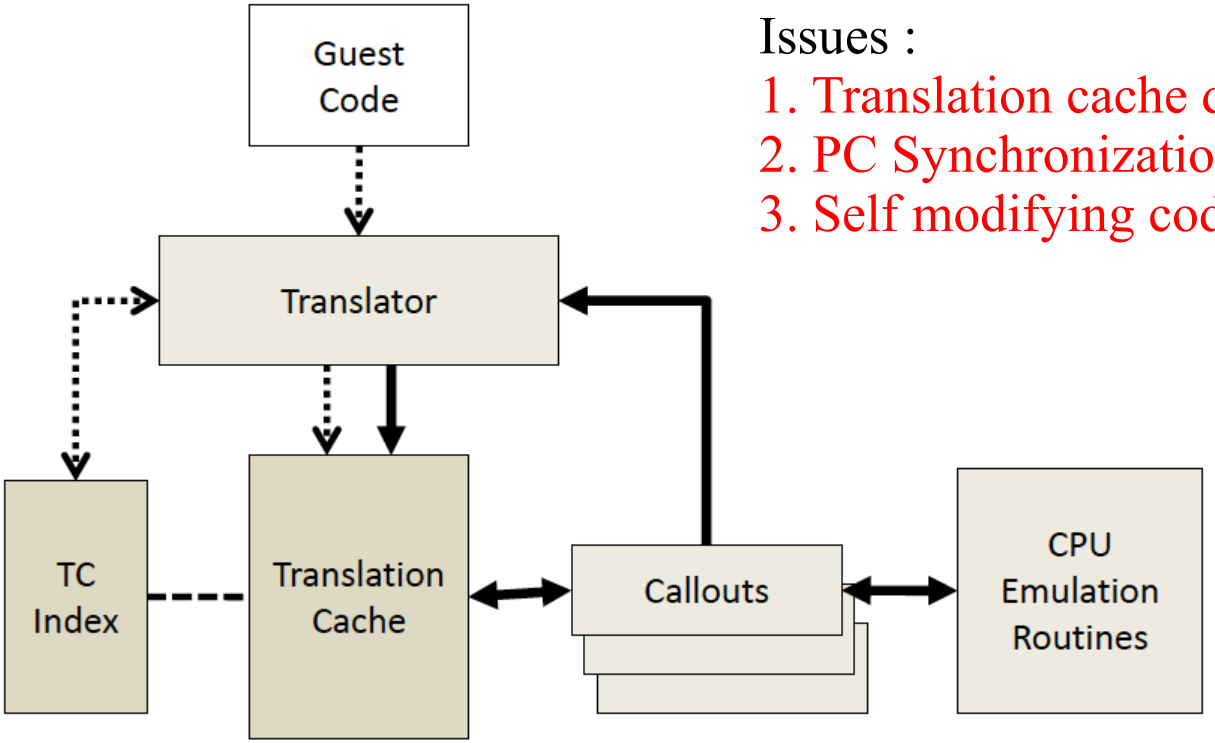
- Traps (Trap in the VMM take control of the situation and trap to Guest OS if needed)
- Interrupts (Deliver interrupts to Guest OS at safe instruction boundaries)



- Not all architectures support it
- Trap costs are high
- Need to virtualize protection levels

Slide Author : Scott Devine

Binary Translation



Issues :

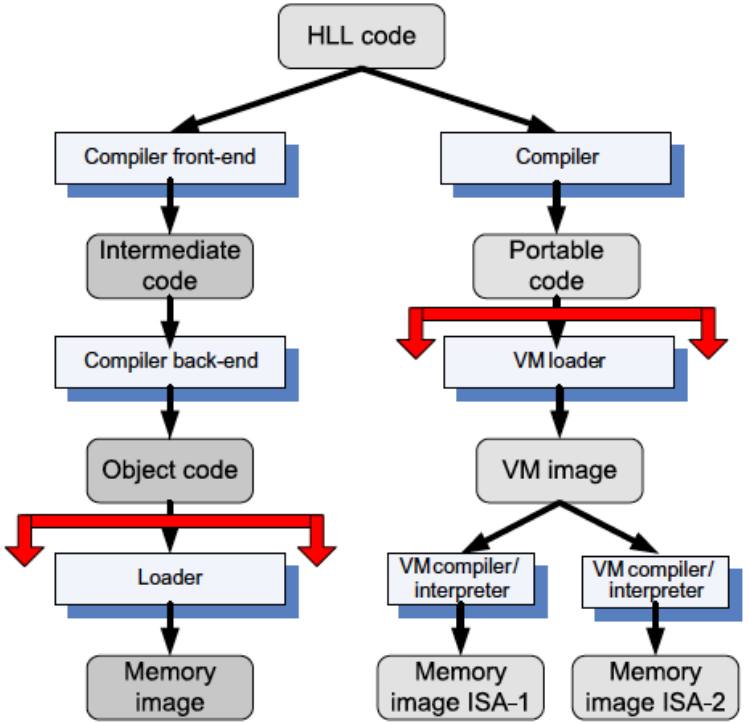
1. Translation cache data structure
2. PC Synchronization on interrupts
3. Self modifying code

Other uses :

1. Cross ISA Translators -Digital Fx 32
2. Optimizing Translators – H.P.Dynamo
3. High Level language byte code translators
   Java, .NET/CLI

Slide Author : Scott Devine

Code Portability

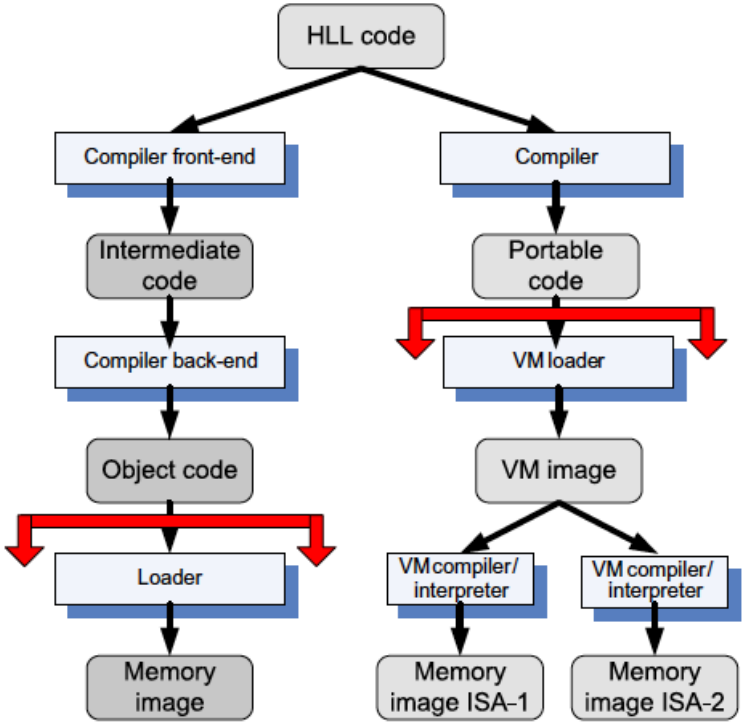


HLL code translated to object code(left) portable code (right)

Binaries created by a compiler for a specific ISA and a specific operating systems are not portable.

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu, 1st edition, 2013

Code Portability : Virtual Machines

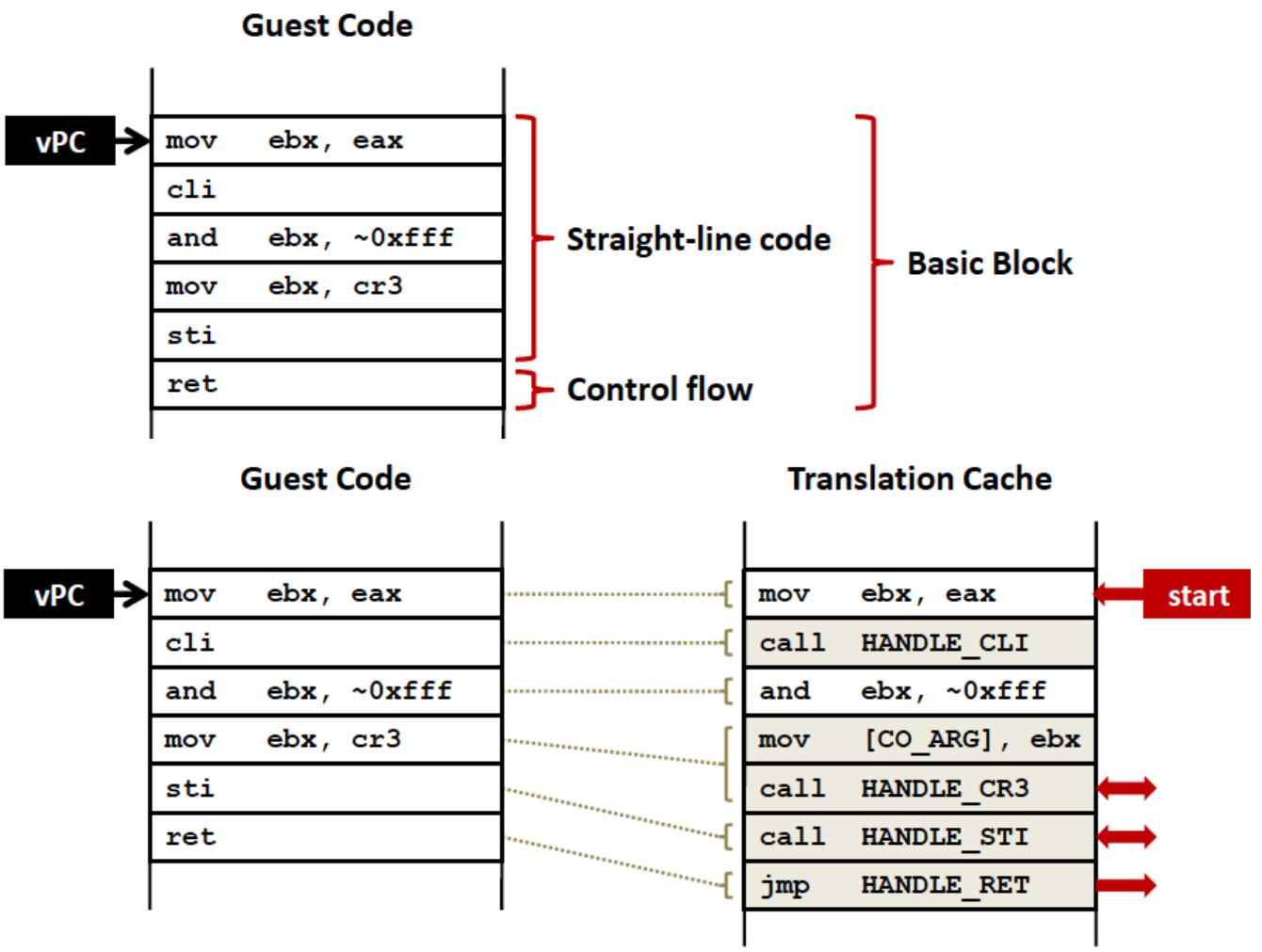


HLL code translated to object code(left) portable code (right)

Compile a HLL program for a VM : Portable code is produced and distributed and then converted by binary translators to the ISA of the host system.

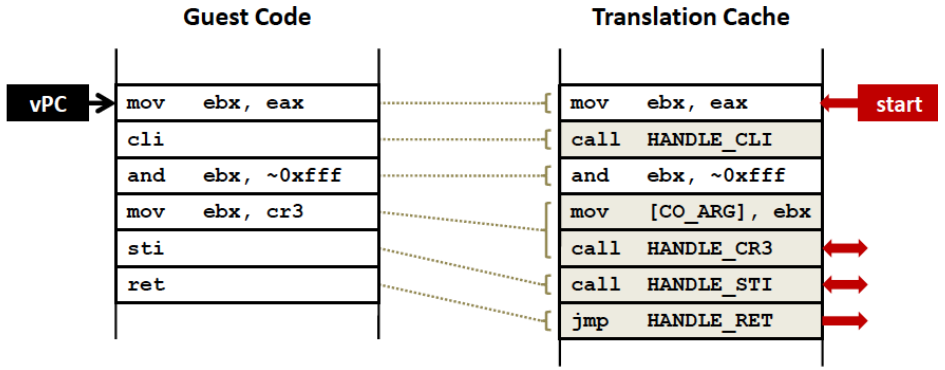
Dynamic binary translation : converts blocks of guest instructions from the portable code to the host instructions; significant performance improvement, as such blocks are cached and reused.

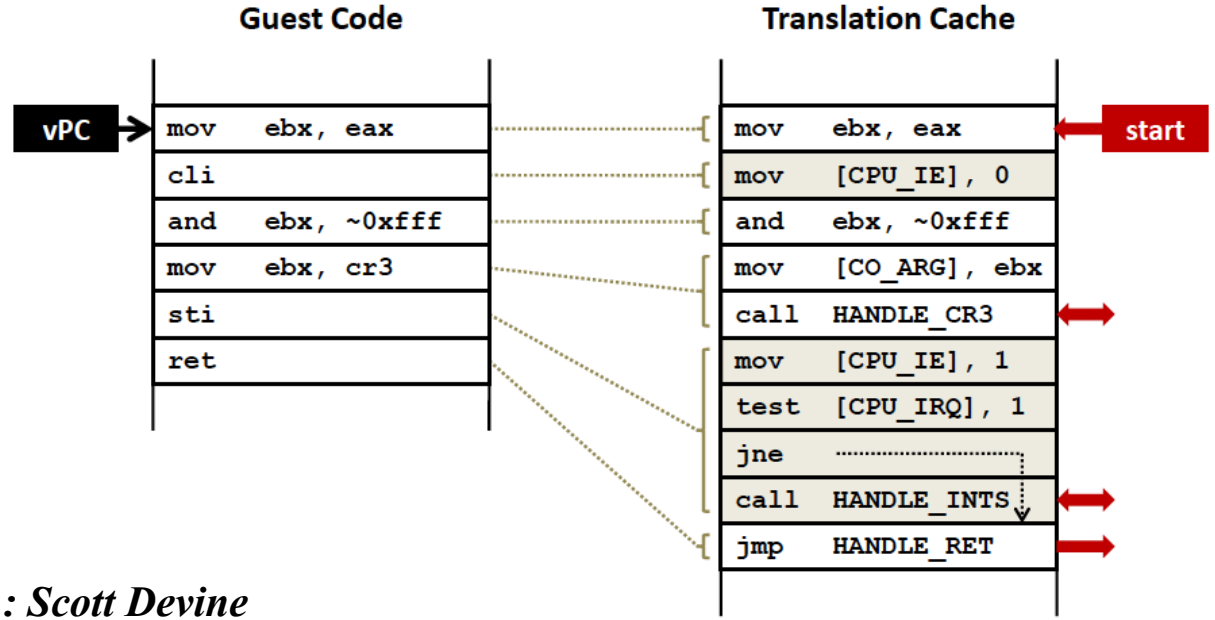
Binary Translation : Basic blocks



Slide Author : Scott Devine

Binary Translation : Basic blocks



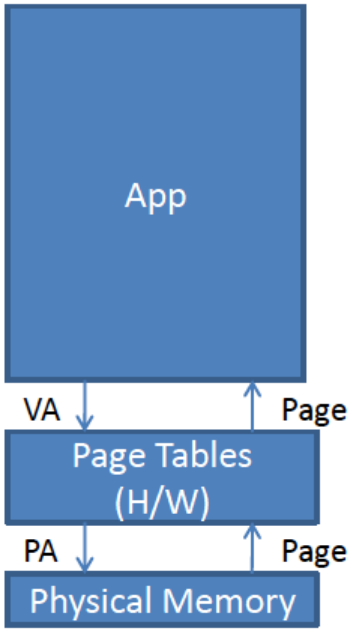


Slide Author : Scott Devine

Memory Virtualization

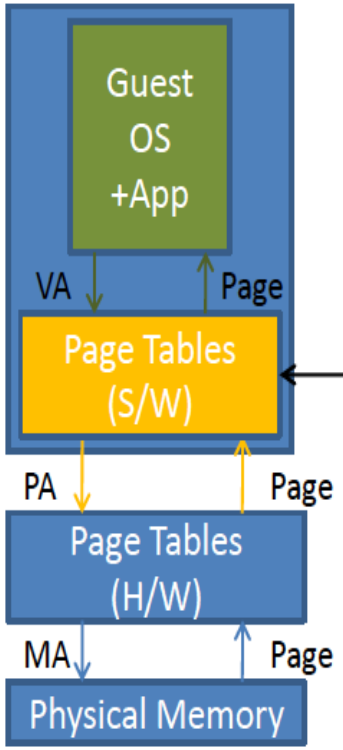
- OS is responsible for setting up and switching between page tables for different processes
- Hardware implements fast table look up using Translational Look Aside Buffer
- Without TLB support, each memory read would require 3-5 extra memory access to look up Page tables
- Two levels of Translation :

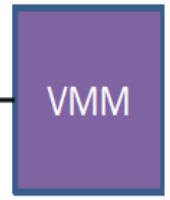
  Guest VA → Guest PA → Host PA
- Guest VA : Guest Virtual Address
- Guest PA : Guest Physical Address
- Host PA : Host Physical Address



Slide Author : Sorav Bansal

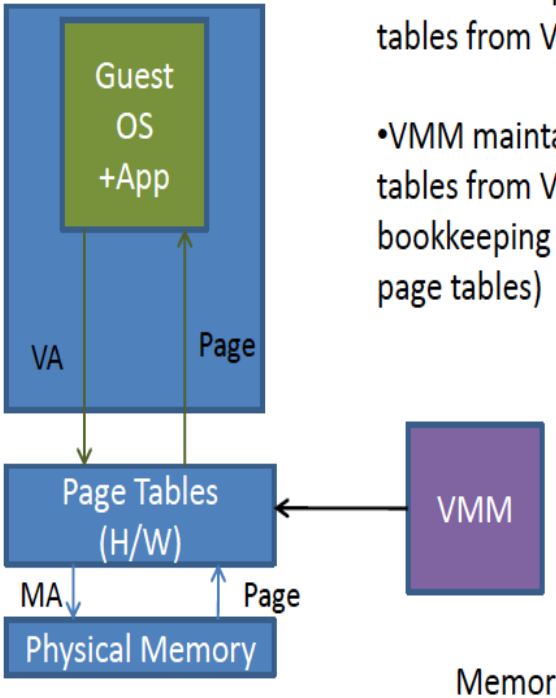
Memory Virtualization





VMM sets up the translation tables from VA to PA

SLOW!!
Each memory access needs to be translated in software ☹



•VMM sets up the translation tables from VA → MA directly.

•VMM maintains separate page tables from VA→PA to perform bookkeeping (also called shadow page tables)

Memory access is at near-native speed ☺

Slide Author : Sorav Bansal

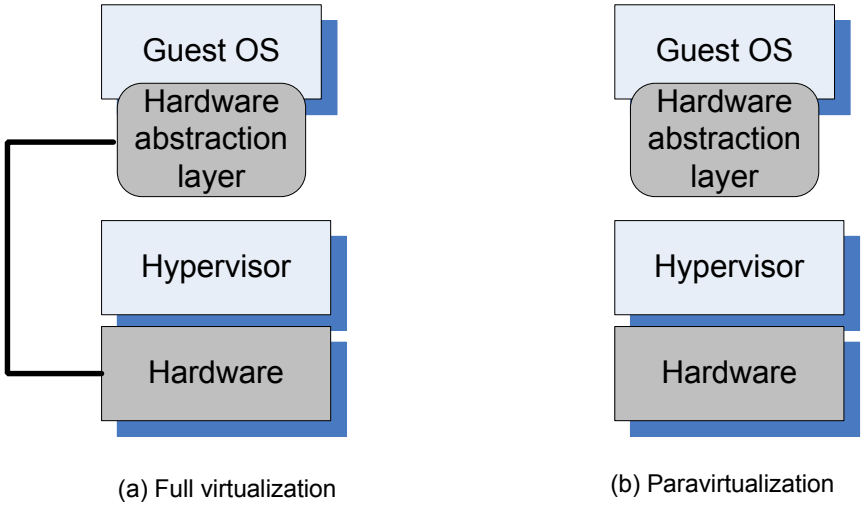
Types of Virtualization

Full virtualization : Guest OS can run unchanged under the VMM as if running directly on the hardware platform. Requires a virtualizable architecture - a statistically significant fraction of machine instructions must be executed without the intervention of the VMM.

Examples: VMware.

- **Para virtualization** : Guest operating system is modified to use only instructions that can be virtualized because some aspects of the hardware cannot be virtualized. It present a simpler interface

  Examples: Xen, Denaly



(a) Full virtualization (b) Paravirtualization

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu, 2nd edition, 2018

Summary of what VMM Does : Virtualizes CPU and Memory

- Traps privileged instructions executed by a guest OS and enforces operation correctness and safety.
- Traps interrupts and dispatches them to the individual guest operating systems.
- Controls virtual memory management.
- Maintains a shadow page table for each guest OS and replicates any modification made by the guest OS in its own shadow page table. This shadow page table points to the actual page frame and it is used by the Memory Management Unit (MMU) for dynamic address translation.
- Monitors the system performance and takes corrective actions to avoid performance degradation. For example, the VMM may swap out a Virtual Machine to avoid thrashing

Condtion for Efficient Virtualization : Popek Goldberg Theorem

- A program running under the VMM should exhibit a behavior essentially identical to that demonstrated when running on an equivalent machine directly.
- The VMM should be in complete control of the virtualized resources.
- A statistically significant fraction of machine instructions must be executed without the intervention of the VMM.

An architecture can be virtualized if the set of instructions that could affect the correct functioning of the VMM are a subset of the privileged instructions

[Popek, Goldberg 1974]

Computer Architecture and Virtualization

- x86 was not designed to be virtualizable
  - VmWare solution
    - Binary translate sensitive instructions to force them to trap into VMM
    - Most instructions execute identically
- Intel VT and AMD-V (2008)
  - Support for virtualization in hardware
  - Obey the principles required to make hardware virtualizable
  - Modern machines no longer require binary translation

Types of Virtualization

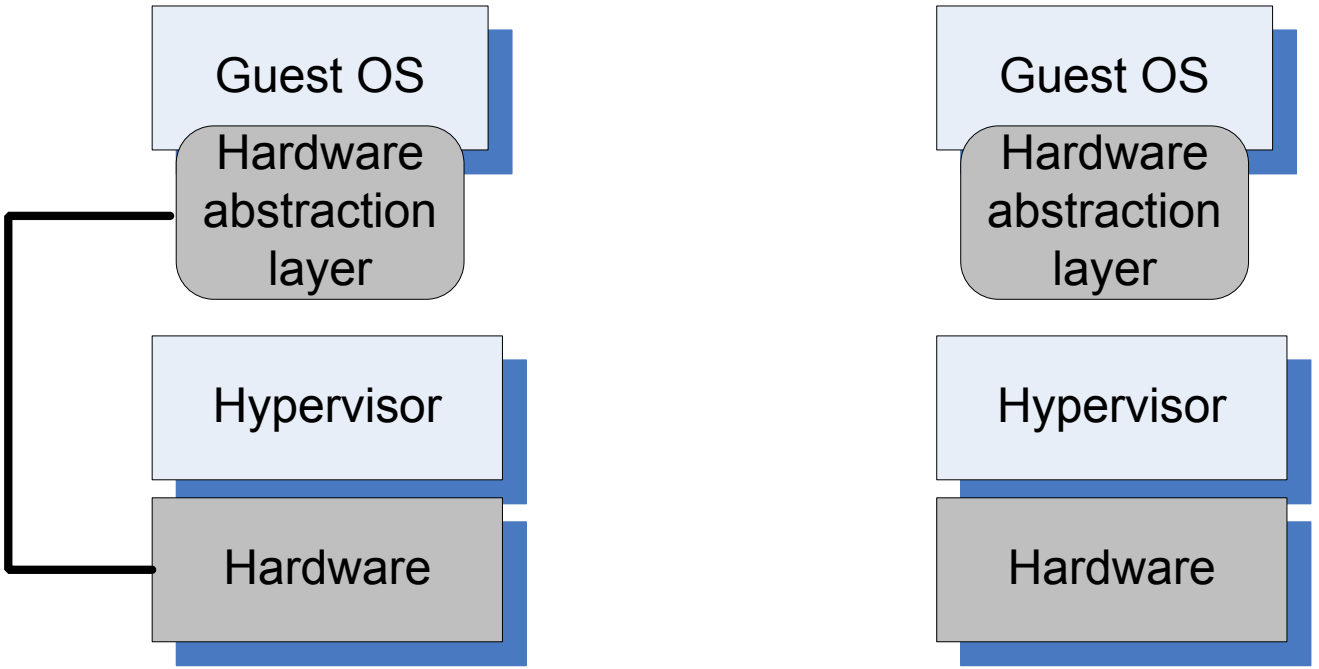
Full virtualization :

- Guest OS can run unchanged under the VMM as if running directly on the hardware platform.
- Requires a virtualizable architecture.
- Examples: VMware.

Para virtualization :

- Guest operating system is modified to use only instructions that can be virtualized. Reasons for para virtualization:
- Some aspects of the hardware cannot be virtualized.
- Improved performance.
- Present a simpler interface.
- Examples: Xen, Denaly

Types of Virtualization



(a) Full virtualization

(b) Paravirtualization

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu, 2nd edition, 2018

Virtualization of x86

- Ring de-privileging - VMMs forces OS and the apps to run at a privilege level > 0
- Ring aliasing - guest OS forced to run at a privilege level other than that it was originally designed for.
- Address space compression - VMM uses parts of the guest address space to store several system data structures.
- Non-faulting access to privileged state - several store instructions can only be executed at privileged level 0 because they operate on data structures that control the CPU operation.

Several Store instructions fail silently when executed at a privilege level other than 0.

Virtualization of x86 Architecture

- Guest system calls which cause transitions to/from privilege level 0 must be emulated by the VMM.
- Interrupt virtualization - in response to a physical interrupt, the VMM generates a virtual interrupt and delivers it later to the target guest OS which can mask interrupts.
- Access to hidden state - elements of the system state
  Example : descriptor caches for segment registers are hidden
- There is no mechanism for saving and restoring the hidden components when there is a context switch from one VM to another.

Virtualization of x86 Architecture

- Ring compression - paging and segmentation protect VMM code from being overwritten by guest OS and applications.
  - Systems running in 64-bit mode can only use paging
  - Paging does not distinguish between privilege levels 0, 1, and 2
  - Guest OS must run at privilege level 3 (0/3/3) mode.
  - Privilege levels 1 and 2 cannot be used thus, the name ring compression.

- The task-priority register is frequently used by a guest OS
- VMM must protect the access to this register and trap all attempts to access it. This can cause a significant performance degradation.

Xen - A VMM Based on Para Virtualization

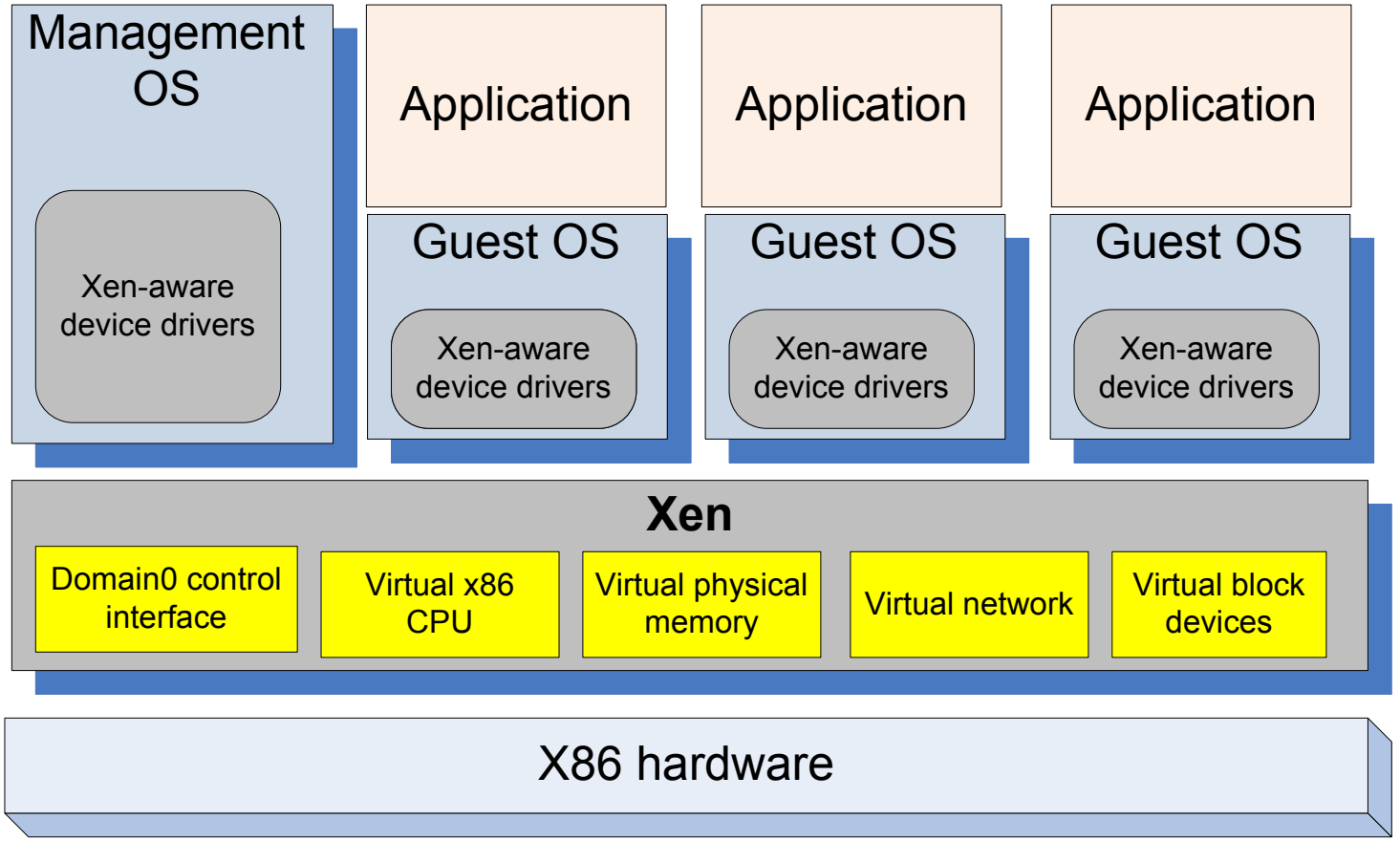


Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu

Xen Implementation on x86 Architecture

- Xen runs at privilege Level 0, the guest OS at Level 1, and applications at Level 3.

- The x86 architecture does not support either the tagging of TLB entries or the software management of the TLB. Thus, address space switching, when the VMM activates a different OS, requires a complete TLB flush; this has a negative impact on the performance.

- Solution - load Xen in a 64 MB segment at the top of each address space and delegate the management of hardware page tables to the guest OS with minimal intervention from Xen. This region is not accessible or re-mappable by the guest OS.

Xen schedules individual domains using the Borrowed Virtual Time (BVT) scheduling algorithm.

A guest OS must register with Xen a description table with the addresses of exception handlers for validation.

Source : Cloud Computing, Theory and Practice, Dan Marinescu

Dom 0 Components : XenStore and Tool Stack

- Xen Store – a Dom0 process.
- Supports a system-wide registry and naming service.
- Implemented as a hierarchical key-value storage.
- A watch function informs listeners of changes of the key in storage they have subscribed to.
- Communicates with guest VMs via shared memory using Dom0 privileges.

- Tool stack - responsible for creating, destroying, and managing the resources and privileges of VMs.
- To create a new VM, a user provides a configuration file describing memory and CPU allocations and device configurations.
- Tool stack parses this file and writes this information in Xen Store.
- Takes advantage of Dom0 privileges to map guest memory, to load a kernel and virtual BIOS and to set up initial communication channels with Xen Store and with the virtual console when a new VM is created.

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu

Xen : Strategies for Virtualization

Function	Strategy
Paging	A domain may be allocated discontinuous pages. A guest OS has direct access to page tables and handles pages faults directly for efficiency; page table updates are batched for performance and validated by *Xen* for safety.
Memory	Memory is statically partitioned between domains to provide strong isolation. `XenoLinux` implements a *balloon driver* to adjust domain memory.
Protection	A guest OS runs at a lower priority level, in ring 1, while *Xen* runs in ring 0.
Exceptions	A guest OS must register with *Xen* a description table with the addresses of exception handlers previously validated; exception handlers other than the page fault handler are identical with *x86* native exception handlers.
System calls	To increase efficiency, a guest OS must install a "fast" handler to allow system calls from an application to the guest OS and avoid indirection through *Xen*.
Interrupts	A lightweight event system replaces hardware interrupts; synchronous system calls from a domain to *Xen* use *hypercalls* and notifications are delivered using the asynchronous event system.
Multiplexing	A guest OS may run multiple applications.
Time	Each guest OS has a timer interface and is aware of "real" and "virtual" time.
Network and I/O devices	Data is transferred using asynchronous I/O rings; a ring is a circular queue of descriptors allocated by a domain and accessible within *Xen*.
Disk access	Only *Dom0* has direct access to IDE and SCSI disks; all other domains access persistent storage through the Virtual Block Device (VBD) abstraction.

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu

Performance and Security Isolation

- The run-time behavior of an application is affected by other applications running concurrently on the same platform and competing for CPU cycles, cache, main memory, disk and network access. It is difficult to predict the completion time!

- Performance isolation - a critical condition for Quality of Service (QoS) guarantees in shared computing environments.

- A VMM is a much simpler and better specified system than a traditional operating system.

  Example:

  Xen has approximately 60,000 lines of code

  Denali has only about half, 30,000.

- Security vulnerability of VMMs is considerably reduced as the systems expose a much smaller number of privileged functions.

Inventory of System VMs

Name	Host ISA	Guest ISA	Host OS	Guest OS	Company
Integrity VM	x86-64	x86-64	HP-Unix	Linux, Windows HP Unix	HP
Power VM	Power	Power	No host OS	Linux, AIX	IBM
z/VM	z-ISA	z-ISA	No host OS	Linux on z-ISA	IBM
Lynx Secure	x86	x86	No host OS	Linux, Windows	LinuxWorks
Hyper-V Server	x86-64	x86-64	Windows	Windows	Microsoft
Oracle VM	x86, x86-64	x86, x86-64	No host OS	Linux, Windows	Oracle
RTS Hypervisor	x86	x86	No host OS	Linux, Windows	Real Time Systems
SUN xVM	x86, SPARC	same as host	No host OS	Linux, Windows	SUN
VMware EX Server	x86, x86-64	x86, x86-64	No host OS	Linux, Windows Solaris, FreeBSD	VMware
VMware Fusion	x86, x86-64	x86, x86-64	MAC OS x86	Linux, Windows Solaris, FreeBSD	VMware
VMware Server	x86, x86-64	x86, x86-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Workstation	x86, x86-64	x86, x86-64	Linux, Windows	Linux, Windows Solaris, FreeBSD	VMware
VMware Player	x86, x86-64	x86, x86-64	Linux Windows	Linux, Windows Solaris, FreeBSD	VMware
Denali	x86	x86	Denali	ILVACO, NetBSD	University of Washington
Xen	x86, x86-64	x86, x86-64	Linux Solaris	Linux, Solaris NetBSD	University of Cambridge

Figure Source : Cloud Computing, Theory and Practice, Dan Marinescu, 2nd edition, 2018

Security Challenges in VMs

In a layered structure, a defense mechanism at some layer can be disabled by malware running at a layer below it.

It is feasible to insert a *rogue VMM,* a Virtual-Machine Based Rootkit (VMBR) between the physical hardware and an operating system.

Rootkit - malware with a privileged access to a system.

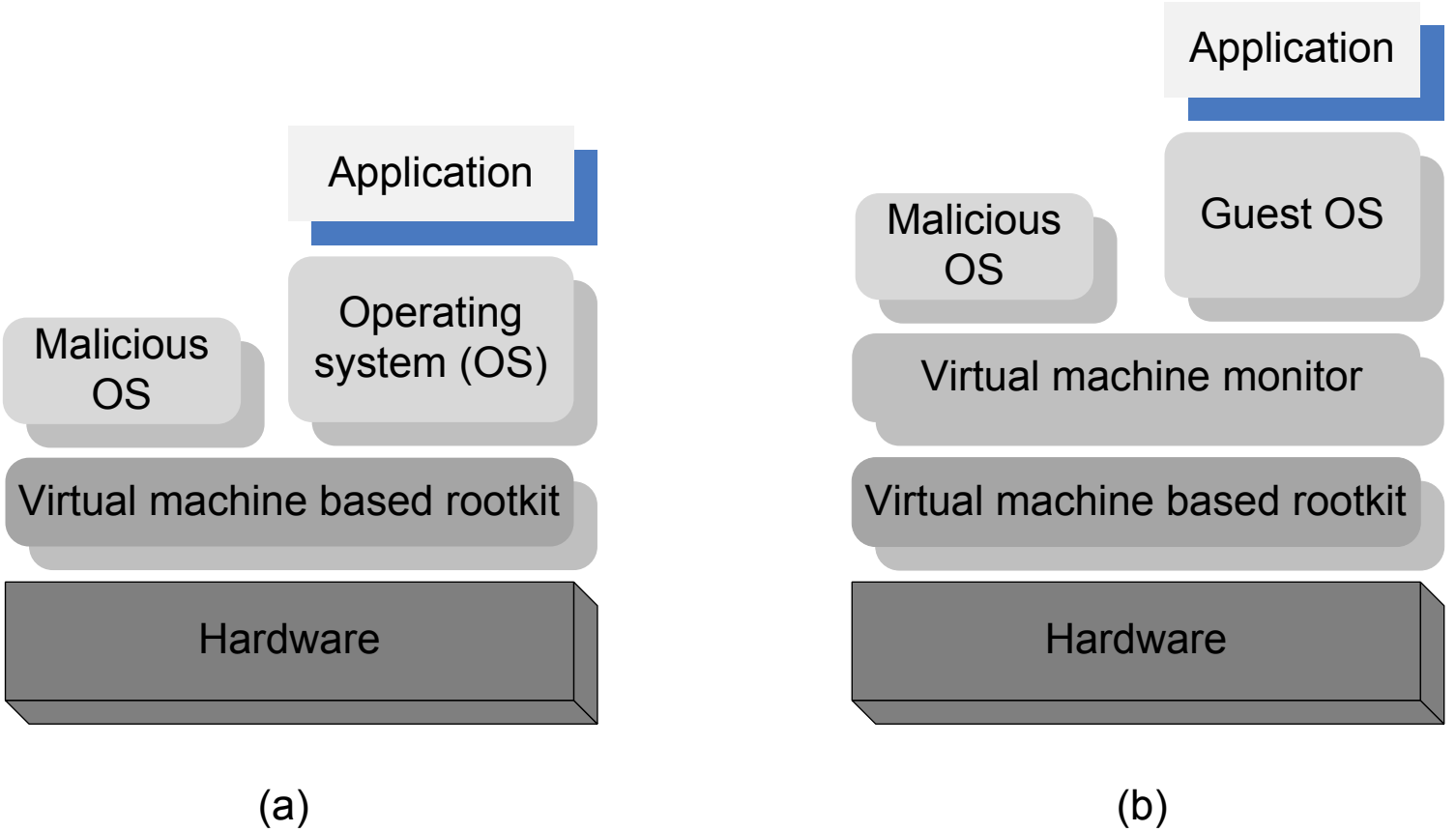
VMBR can enable a separate malicious OS to run surreptitiously and make the malicious OS invisible to the guest OS and to the application running under it.

Under VBMR protection malicious OS could:

1. observe data, events, or state of the target system.
2. run services, such as spam relays or distributed denial-of-service attacks.
3. interfere with the application.

Source : Cloud Computing, Theory and Practice, Dan Marinescu

Security Challenges in VMs



Source : Cloud Computing, Theory and Practice, Dan Marinescu

Security Risks posed by Shared Images

Image sharing poses security risks for IaaS.

A study conducted during the period November 2010 - May 2011 analyzed AMIs available through the public catalog at Amazon

5,303 Linux AMIs

1,202 Windows AMIs.

Many images analyzed allowed a user to undelete files, recover credentials, private keys, or other types of sensitive information with little effort and using standard tools.

Critical software vulnerability revealed by the audit:

98% of the Windows AMIs (249 out of 253)

58% of Linux AMIs (2005 out of 3,432).

The average number of vulnerabilities per AMI: 46 for Windows AMIs and 11 for Linux AMIs.

Source : Cloud Computing, Theory and Practice, Dan Marinescu