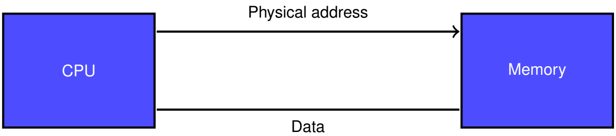
Virtualization and memory hierarchy

Computer Architecture

J. Daniel García Sánchez (coordinator)
David Expósito Singh
Francisco Javier García Blas

ARCOS Group
Computer Science and Engineering Department
University Carlos III of Madrid

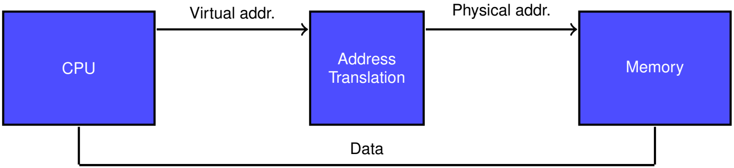
Limits of physical memory addressing



- All programs share a single addressing space.
  - **Physical address space**.

- There is no way to prevent a program to access a resource.

Overcoming the physical limit



- Programs run in a **normalized virtual addresses space**.
- **Address translation**:
  - Performed by hardware.
  - Managed by OS.
- **Supported features**:
  - Protection, Translation, Sharing.

Advantages of virtual memory (I)

- **Translation**:
  - Programs may have a **consistent view** of memory.
  - Decreases **cost** of **multi-threaded** applications.
  - Only the **working set** is needed in main memory.
  - Dynamic structures only use the physical memory that they **really need** (e.g. stack).

Advantages of virtual memory (II)

- **Protection**:
  - Allows to **protect** a process from others.
  - **Attributes** can be set at **page level**.
    - Read only, execution, . . .
  - **Kernel** data protected from programs.
  - Improves protection against **malware**.

- **Sharing**:
  - **A page** can be **mapped** to several processes.
    - e.g. Memory mapped files.

Differences with cache

- **Replacement**:
  - **Cache**: Hardware controlled.
  - **VM**: Software controlled.
- **Size**:
  - Cache size independent from address length.
  - VM size dependent from address length.

Parameters

Parameter	L1 Cache L1	Virtual memory
Block size	16 – 128 bytes	4096 – 65,536 bytes
Hit time	1 – 3 cycles	100 – 200 cycles
Miss penalty	8 – 200 cycles	$10^6 - 10^7$ cycles
Access time	6 – 160 cycles	$8 \cdot 10^5 - 8 \cdot 10^6$ cycles
Transfer time	2 – 40 cycles	$2 \cdot 10^5 - 2 \cdot 10^6$ cycles
Miss rate	0.1% – 10%	0.00001% – 0.001%

Four questions on memory hierarchy

1. Where can a block be placed in the upper level?
   - **Block placement**.
2. How is a block found in the upper level?
   - **Block identification**.
3. Which block must be replaced on miss?
   - **Block replacement**.
4. What happens on a write?
   - **Write strategy**.

Four questions on **virtual memory**

1. Where can a **page** be placed in **main memory**?
   - **Page placement**.
2. How is a **page** found in **main memory**?
   - **Page identification**.
3. Which **page** must be replaced on miss?
   - **Page replacement**.
4. What happens on a **write**?
   - **Write strategy**.

Where is a page placed in main memory?

- A page may be placed in **any page frame** in main memory.
  - Fully associative mapping.
- Managed by the operating system.
- **Goal**: **Minimize miss rate**.
  - Cannot do much with miss penalty.
  - Very high penalty due to slow magnetic disks.

How is a page found in main memory?

- Keep in main memory a **page table per process**.
  - **Mapping table** between **page identifier** and **page frame identifier**.

- Decreasing translation time.
  - **TLB**: *Translation Lookaside Buffer*.
  - Avoids accesses to page table in main memory.

Which page should be replaced on a miss?

- Replacement policy defined by Operating System.
  - Typically **LRU** (*Least-recently used*).

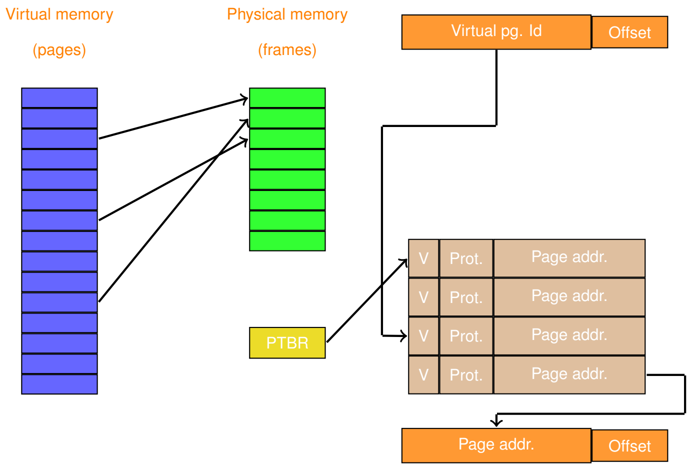
- Architecture must supply support to operating system.
  - **Use bit**: Enabled when page is accessed.
    - Really, only when TLB miss.
  - Operating system periodically zeroes this bit.
    - Records values later.
    - Allows to determine pages that have been modified within an interval.

What happens on a miss?

- Write strategy is always **write-back**.
  - No VM systems with write-through ever built.
  - **Don't be tempted!**

- Disk write costs extremely high.
  - Disk writes minimization.
  - Use **dirty bit** to annotate when a page has been modified.

Page table


Virtual memory
(pages)
Physical memory
(frames)
Virtual pg. Id
Offset
PTBR
V
Prot.
Page addr.
V
Prot.
Page addr.
V
Prot.
Page addr.
V
Prot.
Page addr.
Page addr.
Offset

Page table size

- Assuming **32 bits virtual addresses**, **4 KB pages** and **4 bytes** per **table entry**:
  - **Table size**:

$$\frac{2^{32}}{2^{12}} \times 2^2 B = 2^{22} B = 4MB$$

- **Alternatives**:
  - Multi-level page tables.
  - Inverted page tables.

- **Example**: IA-64
  - Offers both alternatives to OS developer.

TLB: Translation Lookaside Buffer

- **Ideal case**.
  - Each access requires two memory accesses.
    1. Access to page table.
    2. Access to memory.
  - Worse scenario in case of multi-level pages.

- **Solution**:
  - Use translation cache to avoid accesses to page table.
    - **Tag**: Portion of virtual address.
    - **Data**: Frame number, protection bits, validity bit, and dirty bit.

Virtual machines

- Developed in late 60's.
  - Used since in *mainframe* environments.
  - Ignored in single user machines until late 90's.

- Recovered popularity due to:
  - Increasing importance of **isolation** and **security** in modern systems.
  - **Security** and **reliability** failures in operating systems.
  - **Sharing** a single computer by multiple unrelated users.
  - Dramatic increase in processor **performance**.
    - Overhead of VMMs more acceptable now.

Virtual Machine Monitor

A virtual machine is taken to be an efficient, isolated duplicate of the real machine. We explain these notions through the idea of a Virtual Machine Monitor (VMM) . . .
. . . a VMM has three essential characteristics.

- First, the VMM provides an environment for programs which is essentially identical with the original machine,
- second, programs run in this environment show at worst only minor decreases in speed;
- and last, the VMM is in complete control of system resources.

Source: Popek, G. y Goldberg, R. **Formal requirements for virtualizable third generation architectures**.

Communications of the ACM, July 1974

Virtualization

- **General definition**: Any emulation method offering a standard software interface to the physical machine.
  - Java VM? .NET?

- **System level virtual machines**: Offer a complete system environment at binary ISA level.
  - Usually assuming that VM ISA and hardware ISA are identical.
  - **Examples**:
    - IBM VM/370.
    - VMWare ESX Server.
    - Xen.

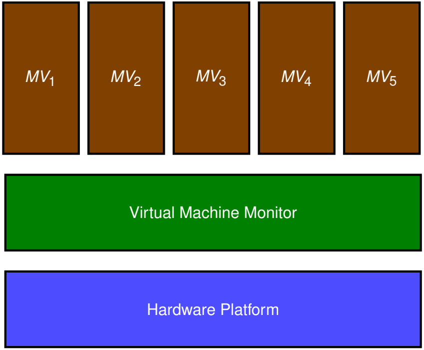
Virtual machine

- Offers to the user the illusion that they have a **complete computer**.
  - Including their own copy of the operating system.

- A computer can run **several virtual machines**.
  - May support several operating systems.
  - All operating systems sharing same hardware.

- **Terminology**:
  - **Host**: Underlying hardware platform.
  - **Guest**: Each virtual machine sharing resources.

VM y VMM: Layers



- **VMM** $\rightarrow$ Software system layer.
  - Monitor runs on hardware platform.
  - Allows execution of multiple virtual machines on single hardware platform.
  - Each virtual machine has its own operating system and applications.
  - Allows running applications without modification.

VMM

- **Virtual Machine Monitor** or **hypervisor**:
  - Software supporting virtual machines.
- VMM determines mapping between: **virtual resources** and **physical resources**.
- **Alternatives for physical resource sharing**:
  - Time sharing.
  - Partitioning.
  - Software emulation.
- A VMM is **smaller** than a traditional OS.

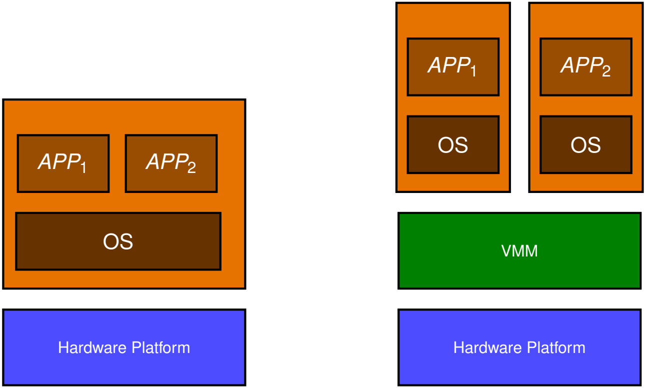
Overhead of VMM

- Depends on workloads.
- User level**processor bound** programs:
  - **Example**: SPEC.
  - **Overhead**: 0.
  - Invocations to OS are rare.
- **I/O intensive** programs → **OS intensive**.
  - Many system calls → Privileged instructions.
  - May lead to **a lot of virtualization overhead**.
- **I/O intensive** and **I/O bound** programs.
  - Low processor utilization.
  - **Virtualization** may be **hidden**.
  - **Low virtualization overhead**.

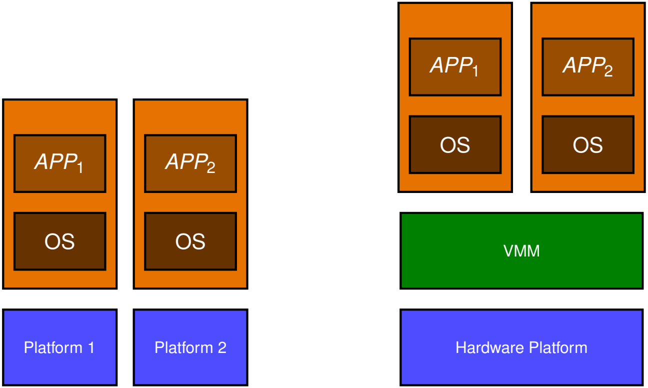
Other uses (in addition to protection)

- **Software management**.
  - VM offers an **abstraction** allowing to run a **complete software stack**.
    - Old operating Systems (DOS?).
  - **Combined** deployment:
    - Stable OS, legacy OS, and next OS version.
- **Hardware management**.
  - VM allows to run **separate software stacks** but on top of a single hardware platform.
    - Servers consolidation.
    - Independence *rightarrow* Higher reliability.
  - **Migrating** VMs in execution.
    - Load balancing.
    - Hardware evacuation due to failures.

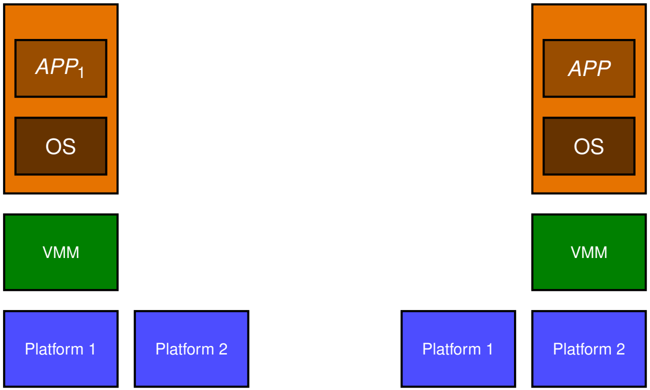
Uses: isolation



Uses: consolidation



Uses: migration



VMM requirements (I)

- A **VMM**:
  - Offers a **software interface** to guest software.
  - **Isolates** a guest state from the rest.
  - **Protects** itself form guests.

- **Guest software** should behave as if there was no VMM, except for:
  - Performance dependent behavior.
  - Limitations of fixed resources when shared among multiple VMMs.

VMM requirements (II)

- Guest software must not be able to modify directly real resources allocation.
- VMM must control everything, even if used by guests.
  - Access to privileged state, address translation, I/O, exceptions, interruptions, . . .
- VMM must run in a more privileged mode than guests.
  - Execution of privileged instructions by VMM.
- Requirements of VMM (equivalent to requirements for virtual memory).
  - A minimum of two processor modes.
  - Subset of privileged instructions, only in privileged mode.
    - Trap if executed in user mode.

ISA support

- If VM considered in ISA design, it is easy to reduce instructions that VMM must execute and emulation time.
  - **But**, Most desktop ISAs designed before VMs.
- VMM must ensure that guests only interact with virtual resources.
  - Guest OS running in user mode.
  - Trying to access hardware leads to a trap.
- If ISA is not VM-aware, VMM must intercept problematic instructions.
  - Introduction of virtual resource.

Impact on virtual memory

- Each guest manages virtual memory.
  - Virtualizing virtual memory?
- VMM distinguishes between **real memory** and **physical memory**.
  - **Real Memory**: Intermediate level between **virtual memory** and **physical memory**.
  - **Guest**: Mapping between **virtual memory** and **real memory**.
  - **VMM**: Mapping between **real memory** and **physical memory**.
- To decrease indirection level, VMM keeps a **shadow page table**.
  - Mapping between **virtual memory** and **physical memory**.
  - VMM must capture changes in **page table** and **pointer to page table**.

ISA support for virtual memory virtualization

- IBM 370 includes additional indirection level managed by VMM.
  - Eliminates the need of shadow the page table.

- TLB virtualization.
  - VMM manages TLB and keeps copies of TLB in each guest.
  - TLB accesses generate a trap.
  - TLB with process identifiers simplifies management.
    - Allow entries from multiple VMs over the VMM at the same time.

Input/Output impact

- **Most complex** part in virtualization.
  - Increasing number of I/O devices.
  - Increasing diversity of I/O devices.
  - Sharing devices among VMs.
  - Support for an increasing variety of drivers.
- **General part** of driver remains on the guest side.
  - **Specific part** in VMM.
- **Device dependent** method.
  - **Disks**: Partitioned by VMM for creating virtual disks.
  - **Network interfaces**: Multiplexed over time.
    - VMM manages **virtual network addresses**.

7 Virtualization technologies

- Impure virtualization
- ISA Technologies

Impure virtualization

- Solution for **non-virtualizable** architectures and for decreasing **performance problems**:
- **Approaches**:
  - **Paravirtualization**: Port guest OS code to a modified ISA.
    - Development effort.
    - Need to adapt code for every OS.
    - Source code must be available.
  - **Binary translation**: Replace non-virtualizable instructions by emulation code or VMM calls.
    - Does not require source code.
    - Some emulations are possible in user space.

Example: XEN

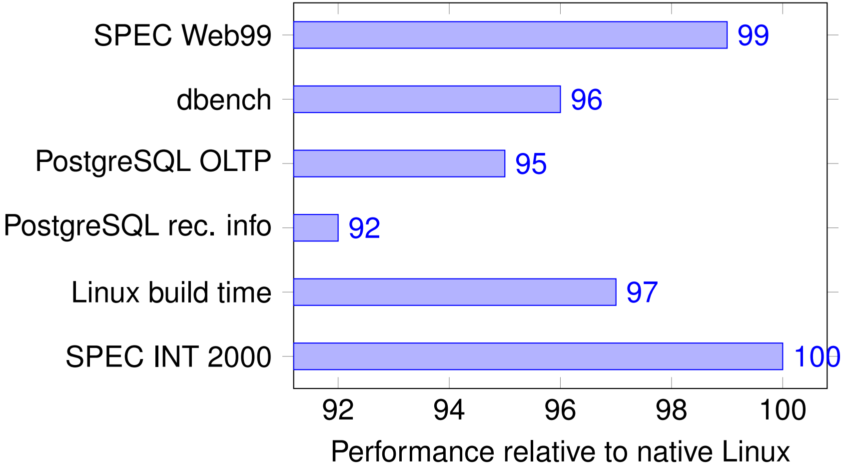
- **Xen**: VMM *open-source* for x86.
- **Strategy**: **Paravirtualization.**
  - Small modifications into the OS to simplify virtualization.
- **Examples of paravirtualization**:
  - Avoid TLB **flush** when the VMM is invoked.
    - Xen mapped to upper 64MB in each VM.
  - Allow guest to **allocate** pages.
    - Check if protection restrictions are not violated.
  - **Protection** between programs and guest → Use **protection levels** from x86:
    - Xen (0), Guest (1), Programs (3).

Changes in Xen

- Changes needed in Linux $\rightarrow$ around $3,000$ lines of code.
  - 1% x86 specific code.

Operating System	Runs as *host*	Runs as *guest*
Linux 2.4	Yes	Yes
Linux 2.6	Yes	Yes
NetBSD 2.0	No	Yes
NetBSD 3.0	Yes	Yes
Plan 9	No	Yes
FreeBSD 5	No	Yes

Performance in Xen

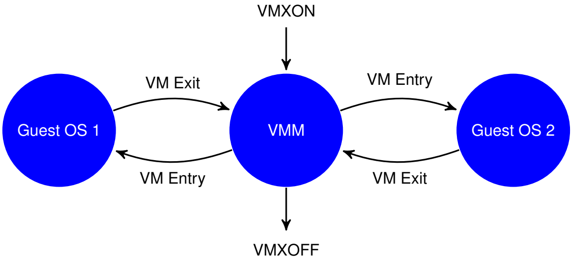


7 Virtualization technologies

- Impure virtualization
- ISA Technologies

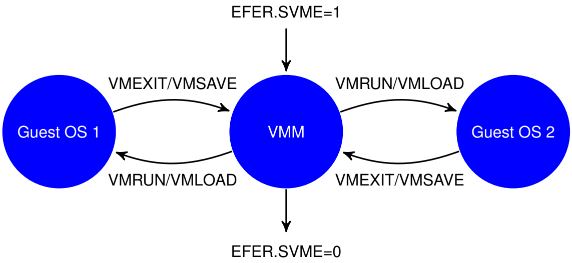
Intel Virtualization Technology

- Adds new instructions:
  - VMXON
  - VMXOFF
  - VMLAUNCH
  - VMRESUME
  - ...



AMD Secure Virtual Machine

- Adds new instructions:
  - VMRUN/VMLOAD.
  - VMCALL/VMSAVE.
  - ...



Operation modes

- **VMX root**:
  - Fully privileged.
  - Designed to be used with VMM.

- **VMX non-root**:
  - Non privileged.
  - Designed to be used by guest software.

Entering and exiting virtual machines

- **VM Entry**:
  - Transition from VMM to host.
  - Enters non-root mode.
  - Loads guest state.
  - **VMLAUNCH** instruction used for initial entry.
  - **VMRESUME** instruction used for subsequent entries.

- **VM Exit**:
  - **VMEXIT** instruction used to enter VMM mode.
  - Enters in root mode.
  - Saves guest state.
  - Loads VMM state.
  - There are instructions and events that cause a **VMEXIT**.

Benefits from VT technology

- Decreases OS dependency.
  - Removes needs for binary translation.
  - Facilitates the support for old operating systems.
- Improves robustness.
  - Removes need for complex techniques.
  - Smaller and simpler VMMs.
- Improves performance.
  - Less transitions to VMM.

Summary

- Virtual memory offers a mechanism for **translation**, facilitating **protection** and **sharing**.
- Virtual memory policies:
  - Placement: Fully associative.
  - Identification: Page Table.
  - Replacement: Usually LRU with TLB support.
  - Writing: Always write-back.
- Virtual machines: isolation, security, reliability, and sharing.
- Uses of VMM: protection, management sw/hw (isolation, consolidation, migration).
- Technologies: Impure virtualization and solutions in ISA.

References

- **Computer Architecture. A Quantitative Approach**
  5th Ed.
  Hennessy and Patterson.
  Sections: B.4, 2.4.

- **Recommended exercises**:
  - B.12, B.13, B.14, 2.20, 2.21, 2.22, 2.23

Virtualization and memory hierarchy

Computer Architecture

J. Daniel García Sánchez (coordinator)
David Expósito Singh
Francisco Javier García Blas

ARCOS Group
Computer Science and Engineering Department
University Carlos III of Madrid