


CSE


Extended OS



CSE

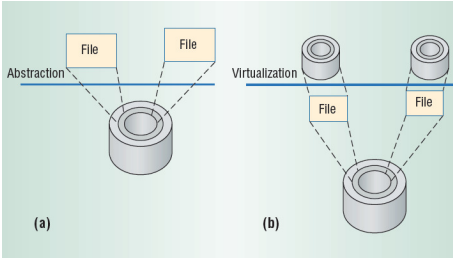

Virtual Machines

References:
 Smith, J.E.; Ravi Nair; , "The architecture of virtual machines,"
Computer , vol.38, no.5, pp. 32- 38, May 2005



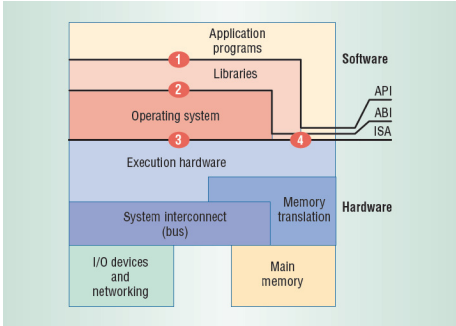

CSE

Abstraction & Virtualisation

CSE

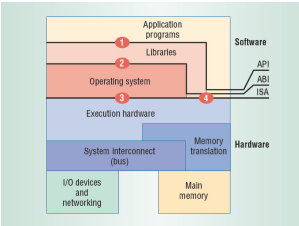

Interface Levels

CSE

Instruction Set Architecture

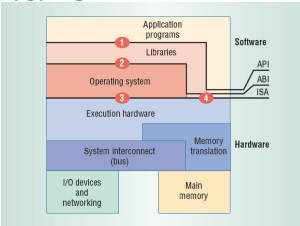

- Interface between software and hardware
- Divided between privileged and un-privileged parts

CSE

Application Binary Interface

- Interface between programs hardware + OS
- Consists of system call interface + un-privileged ISA

Application Programming Interface

- Interface between programs hardware + OS
- Consists of library calls + un-privileged ISA
 - Syscalls usually called through library.

The diagram illustrates the layers of an application programming interface. At the top, 'Application programs' and 'Libraries' (labeled 1 and 2) interact with the 'Operating system' (labeled 3). The OS then interacts with 'Execution hardware' (labeled 4). Below the hardware, there is a 'System interconnect (bus)', 'Memory translation', 'I/O devices and networking', and 'Main memory'. The top layer is labeled 'Software' and the bottom layer is labeled 'Hardware'. An 'API/ABI/ISA' interface is shown connecting the software and hardware layers.

Process versus System Virtual Machine

(a) Process virtual machine: A guest OS runs on hardware. An application process runs on top of the OS. Virtualizing software is used to create a process virtual machine that runs the application process.

(b) System virtual machine: A guest OS runs on hardware. Applications run on top of the OS. Virtualizing software is used to create a system virtual machine that runs the OS and applications.

OS is an extended virtual machine

- Multiplexes the "machine" between applications
 - Time sharing, multitasking, batching
- Provided a higher-level machine for
 - Ease of use
 - Portability
 - Efficiency
 - Security
 - Etc....

JAVA – Higher-level Virtual Machine

- write a program once, and run it anywhere
 - Architecture independent
 - Operating System independent
- Language itself was clean, robust, garbage collection
- Program compiled into bytecode
 - Interpreted or just-in-time compiled.
 - Lower than native performance

Conventional versus Emulation/Translation

(a) Conventional compilation: HLL program is processed by a compiler front end to produce intermediate code, then a compiler back end to produce object code. The object code is distributed and loaded into a memory image.

(b) Emulation/translation: HLL program is processed by a compiler to produce portable code. This code is distributed and loaded by a VM loader into a virtual memory image. The VM image is then interpreted or compiled by a VM interpreter/compiler to produce host instructions.

Issues

- Legacy applications
- No isolation nor resource management between applets
- Security
 - Trust JVM implementation? Trust underlying OS?
- Performance compared to native

CSE

Is the OS the “right” level of extended machine?

- Security
 - Trust the underlying OS?
- Legacy application and OSs
- Resource management of existing systems suitable for all applications?
- What about activities requiring “root” privileges

THE UNIVERSITY OF NEW SOUTH WALES

CSE

Virtual Machine Monitors

- Provide scheduling and resource management
- Extended “machine” is the actual machine interface.

THE UNIVERSITY OF NEW SOUTH WALES

CSE

IBM VM/370

The diagram illustrates the IBM VM/370 architecture. At the top, 'Virtual 370s' are shown as a layer above three 'CMS' (Control Macro System) instances. Below the CMS instances is the 'VM/370' layer, which acts as the virtual machine monitor. At the bottom is the '370 Bare hardware'. Arrows indicate 'System calls here' between the CMS instances and the VM/370 layer. On the left, 'I/O instructions here' and 'Trap here' are shown with arrows pointing to the VM/370 layer. On the right, 'Trap here' is shown with an arrow pointing from the CMS instances to the VM/370 layer.

THE UNIVERSITY OF NEW SOUTH WALES

CSE

Advantages

- Legacy OSes (and applications)
- Server consolidation
- Concurrent OSes
 - Linux – Windows
 - Primary – Backup
 - High availability
- Test and Development
- Security
 - VMM (hopefully) small and correct
- Performance near bare hardware
 - For some applications

THE UNIVERSITY OF NEW SOUTH WALES

CSE

The diagram classifies VMs into two main categories: Process VMs and System VMs. Process VMs are further divided into 'Same ISA' (Multiprogrammed systems, Same-ISA dynamic binary optimizers) and 'Different ISA' (Dynamic translators, High-level-language VMs). System VMs are divided into 'Same ISA' (Classic system VMs, Hosted VMs) and 'Different ISA' (Whole-system VMs, Codesigned VMs).

THE UNIVERSITY OF NEW SOUTH WALES

CSE

Figure 1-29 illustrates two types of hypervisors. (a) A Type 1 hypervisor, where multiple operating systems (Windows, Linux, etc.) run directly on the hardware. (b) A Type 2 hypervisor, where a single host operating system runs on the hardware, and the hypervisor runs as a process within that OS, managing guest OS processes.

Figure 1-29. (a) A type 1 hypervisor. (b) A type 2 hypervisor.

THE UNIVERSITY OF NEW SOUTH WALES

Virtual R3000???

- Interpret
 - System/161
 - slow
 - JIT dynamic compilation
- Run on the real hardware???

THE UNIVERSITY OF NEW SOUTH WALES

R3000 Virtual Memory Addressing

- MMU
 - address translation in hardware
 - management of translation is software

Figure 2.10 Virtual Memory Addressing

THE UNIVERSITY OF NEW SOUTH WALES

R3000 Address Space Layout

	0xFFFFFFFF	
	0xC0000000	kseg2
	0xA0000000	kseg1
	0x80000000	kseg0
	0x00000000	kuseg

- kuseg:
 - 2 gigabytes
 - MMU translated
 - Cacheable
 - user-mode and kernel mode accessible

THE UNIVERSITY OF NEW SOUTH WALES

R3000 Address Space Layout

	0xFFFFFFFF	
	0xC0000000	kseg2
	0xA0000000	kseg1
	0x80000000	kseg0
	0x00000000	kuseg

- kseg0:
 - 512 megabytes
 - Fixed translation window to physical memory
 - 0x80000000 - 0x9ffffff virtual = 0x00000000 - 0x1ffffff physical
 - MMU not used
 - Cacheable
 - Only kernel-mode accessible
 - Usually where the kernel code is placed

Physical Memory

}

THE UNIVERSITY OF NEW SOUTH WALES

R3000 Address Space Layout

	0xFFFFFFFF	
	0xC0000000	kseg2
	0xA0000000	kseg1
	0x80000000	kseg0
	0x00000000	kuseg

- kseg1:
 - 512 megabytes
 - Fixed translation window to physical memory
 - 0xa0000000 - 0xbffffff virtual = 0x00000000 - 0x1ffffff physical
 - MMU not used
 - **NOT** cacheable
 - Only kernel-mode accessible
 - Where devices are accessed (and boot ROM)

Physical Memory

}

THE UNIVERSITY OF NEW SOUTH WALES

R3000 Address Space Layout

	0xFFFFFFFF	
	0xC0000000	kseg2
	0xA0000000	kseg1
	0x80000000	kseg0
	0x00000000	kuseg


- kseg2:
 - 1024 megabytes
 - MMU translated
 - Cacheable
 - Only kernel-mode accessible

THE UNIVERSITY OF NEW SOUTH WALES

Cse

Issues

- Privileged registers (CP0)
- Privileged instructions
- Address Spaces
- Exceptions (including syscalls, interrupts)
- Devices



THE UNIVERSITY OF
NEW SOUTH WALES