Extended OS

Virtual Machines

Abstraction & Virtualisation

Interface Levels

Instruction Set Architecture
- Interface between software and hardware
- Divided between privileged and unprivileged parts

Application Binary Interface
- Interface between programs hardware + OS
- Consists of system call interface + unprivileged ISA
**Application Programming Interface**

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

**Process versus System Virtual Machine**

- 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**

- HLL program
  - Compiler
  - Intermediate code
  - Compiler back end
  - Object code
  - Distribution
  - Loader
  - Memory image

- HLL program
  - Compiler
  - Portable code
  - VM loader
  - Distribution
  - VM interpreter/translator
  - Host instructions
Issues

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

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

Virtual Machine Monitors

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

IBM VM/370

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
Virtual R3000???

- Interpret
  - System/161
    - slow
  - JIT dynamic compilation

- Run on the real hardware??

R3000 Virtual Memory Addressing

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

Figure 2.10 Virtual Memory Addressing

R3000 Address Space Layout

- kseg:
  - 2 gigabytes
  - MMU translated
  - Cacheable
  - user-mode and kernel mode accessible
  - 0x80000000 - 0x1fffffff virtual = 0x00000000 - 0x1fffffff physical
  - MMU not used
  - Cacheable
  - Only kernel mode accessible
  - Usually where the kernel code is placed
  - 0x80000000 - 0xffffffff physical

- kseg0:
  - 512 megabytes
  - Fixed translation window to physical memory
  - 0x08000000 - 0xffffffff virtual = 0x00000000 - 0x1fffffff physical
  - MMU not used
  - Cacheable
  - Only kernel mode accessible
  - Usually where the kernel code is placed

- kseg1:
  - 512 megabytes
  - Fixed translation window to physical memory
  - 0xa0000000 - 0xbfffffff virtual = 0x00000000 - 0x1fffffff physical
  - MMU not used
  - NOT cacheable
  - Only kernel mode accessible
  - Where devices are accessed (and boot ROM)
R3000 Address Space Layout

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

Issues

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