x86 ISA Modifications to support Virtual Machines

Douglas Beal

Ashish Kumar Gupta

CSE 548 Project

Outline of the talk

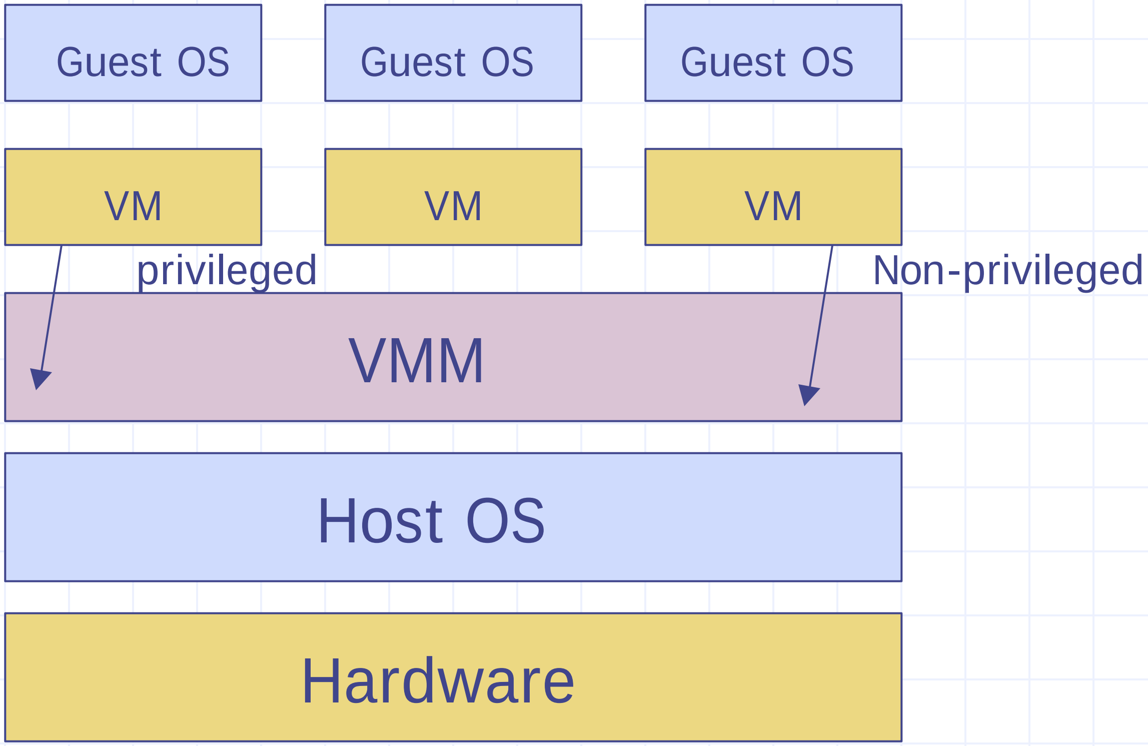
- Review of Virtual Machines
- What complicates Virtualization
- Technique for Virtualization (so far)
- Technique for Virtualization (ours)
- Experiments and Conclusions

Virtual Machines

- Virtual Machine Monitor (VMM) : a software that creates isolated programming environments that provide users with the appearance of direct access to the real machine.
- Virtual Machine (VM) : The isolated programming environments.
- Guest OS : The OS running on top of the VM.
- Host OS : The OS on top of which the VMM is running.

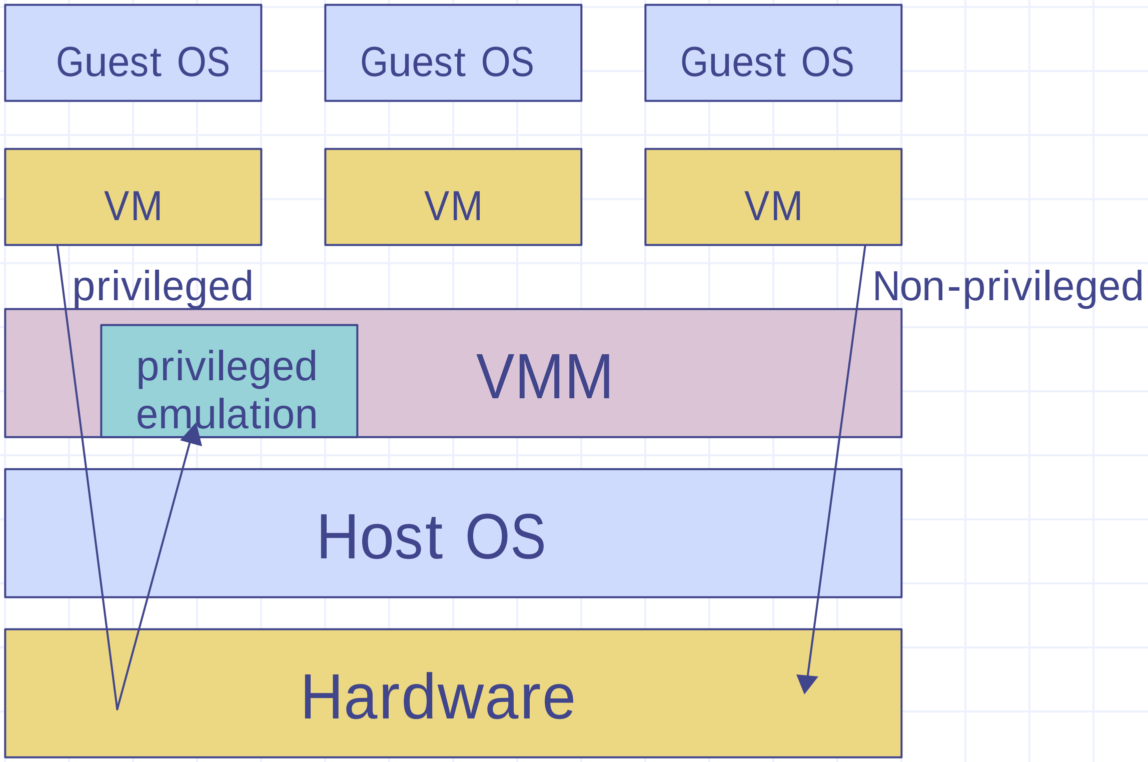
Types of Virtual Machines

- Emulation : processor in software



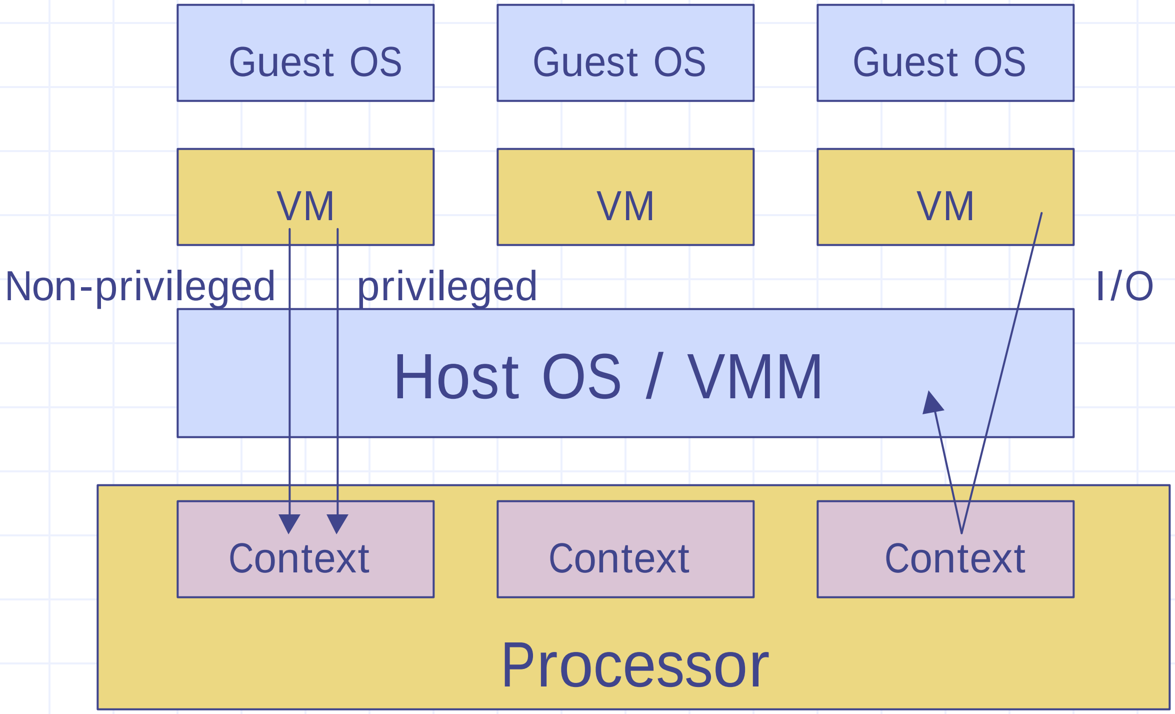
Types of Virtual Machines

- Software virtualization : emulate privileged structures



Types of Virtual Machines

- Hardware virtualization : privileged structures duplicated/swapped.



Why Virtual Machines ?

- Simultaneous support for multiple Operating Systems / Applications
  - E.g. Windows and Unix
- Operating System Debugging
  - Can proceed while system is being used for normal work.
  - If a VM crashes, the other VMs can continue to work.
- Security Isolation
  - Each VM is in an address space of its own. It can’t even name the resources in the address space of other VMs.

What complicates Virtualization

- Hardware devices
  - Static allocation of hardware to the VMs
  - Emulation in software
    - Guest OS can run without modification.
    - Inefficient because of constraints imposed by real hardware interface.
  - Virtual Device Emulation
    - More efficient because VMM can dictate the semantics
    - Driver needs to be written for every different guest OS
  - I/O API
    - VMs can be layered in arbitrary manner to provide various kind of services.

What complicates Virtualization

- Most processors contain two/more privilege levels
  - Most privileged : used by the OS and drivers
  - Least privileged : used by application software
- Privileged instructions, when executed in the *user mode* generate a trap.
- Sensitive instructions can’t be executed directly on the processor.
- Presence of sensitive non-privileged instructions makes virtualization difficult.

What complicates Virtualization

- Processor Resources
  - All processor resources need not be virtualized.
  - Virtualizing a subset can simplify the logic of the VMM.
  - In Denali [WSD02], no virtual memory is provided to guest OS.
  - To be able to run all operating systems without any modifications, all the resources need to be virtualized.

Techniques for Virtualization

- Scan Before Execute (SBE)
  - Creates virtual code pages which mirror the real code pages.
  - Place a breakpoint before the sensitive non-privileged instructions so that they can be emulated in the software.

Techniques for Virtualization

- Dynamic Scan Before Execute (e.g. Plex86)
  - SBE working on page-by-page code basis.
  - Fetches and decodes a sequence of instructions up to a branch instruction and places a breakpoint there.
  - When code gets executed, a breakpoint exception is generated at the branch instruction.
  - The VMM receives the exception and repeats the same process for the next code sequence.

Techniques for Virtualization

- Para-virtualization (e.g. Denali)
  - Virtualizing everything is difficult. So, virtualize only a subset of resources.
  - Denali : Goal is not to run legacy OS, but to take advantage of the security that VM provides.

Techniques for Virtualization

- Support for virtualization in hardware (e.g. IBM S/390).
  - Applications run at full native speed except a few privileged instructions which are emulated in software.
  - The virtualization overhead is minimal.

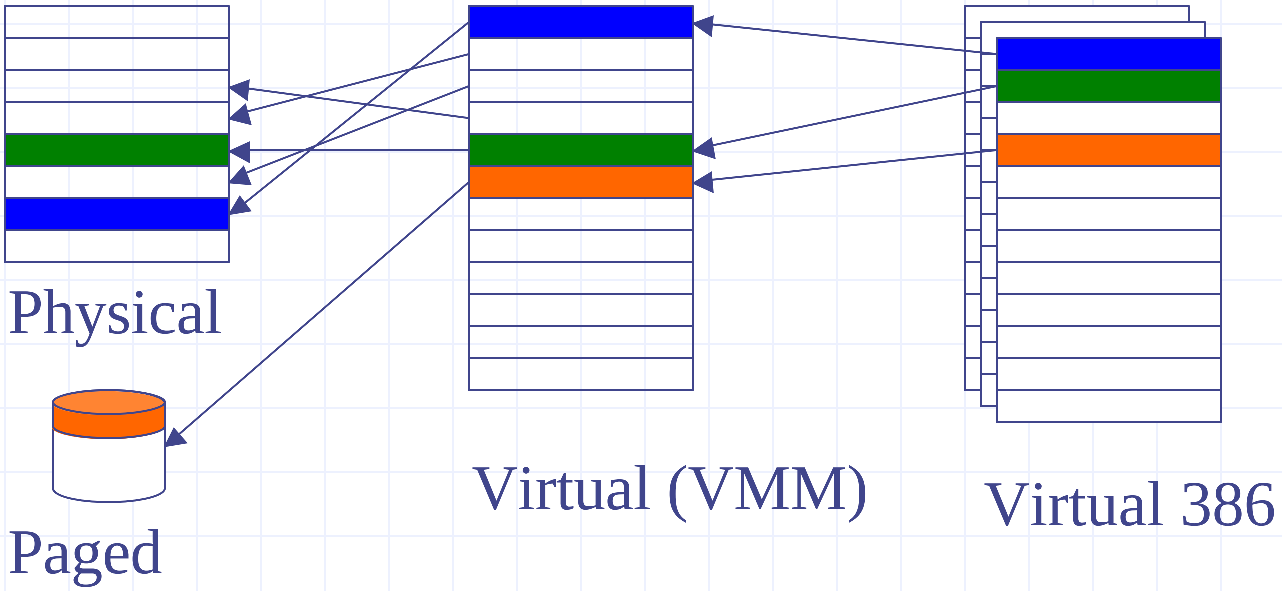
Outline of the talk

- Review of Virtual Machines
- What complicates Virtualization
- Technique for Virtualization (so far)
- Technique for Virtualization (ours)
- Experiments and Conclusions

x86 Changes for Virtualization

- Make non-privileged sensitive instructions privileged
- Add another level of virtual memory
- Add V386 mode
  - Bad instructions now see their own state
  - Processors already have V86 mode

Virtual 80386 Mode



- VMM virtual memory looks like physical from V386 view

Prototype – x86 ISA

- Bochs – LGPL x86 Emulator
  - Easy modification of x86 ISA
    - Create V386 Mode
      - EFLAGS bit, just like V86
    - Add another VM level
      - Two translations necessary in V386 Mode
    - New trap/interrupt
      - IO instructions
        - traps to VMM for hardware emulation
      - Timer
        - VM preemption

Prototype - VMM

- Linux Kernel
  - Modify to launch processes in V386 mode
  - Add timer support for VM preemption
  - Add traps for hardware emulation

Measurements

- Run benchmarks on VMs
- Run benchmarks on UML
  - User Mode Linux
    - Port of Linux to Linux
      - Privileged instructions changed to syscalls
- Compare V386 and UML results
  - Bochs x86 instruction trace

What we hope to learn

- Speed difference between V386 and UML
  - Convincing for implementation in real processor?
- Difficulty of adding VM process support to Kernel
  - Useful for other virtualization techniques
  - Subjects virtualization to Kernel scheduling

x86 ISA Modifications to support Virtual Machines

Douglas Beal

Ashish Kumar Gupta

CSE 548 Project

Features of Virtual Machines

- VMM provides an execution environment almost identical to the original machine. Some differences arise due to
  - Resource sharing
  - Timing dependencies
- A large fraction of the virtual processor’s instructions must be executed directly on the real machine.
- VMM must be in control of the system resources.

Components of a VMM

- Dispatcher
  - The top level control module of the VMM.
  - Jump to the dispatcher is placed in every location to which the machine traps.
- Allocator
  - Decides what system resources are to be provided.
  - Makes sure that the same resource is not provided to more than one VM concurrently.
- Interpreter
  - One per privileged instruction, it simulates the effect of the instruction which trapped.