# ECE 650 Systems Programming & Engineering

#### Spring 2018

#### Hypervisors & CPU Virtualization

Tyler Bletsch Duke University

Slides are adapted from Brian Rogers (Duke)

### **Overview of Virtualization**

- We've discussed a form of virtualization
  - Provided by the OS to user level programs / processes
  - Programs written as if they have full access to an entire machine
    - Resources: memory, disk, I/O, CPU
    - User processes typically do not need to care or know that these resources are in fact shared (concurrently or time-sliced) with others
- We can extend this to the OS
  - Support multiple OS instances (guest OSes) running on a CPU
  - Can be multiple instances of same OS or different OSes
  - Typically, guest OSes run on top of new software layer
    - Called Hypervisor (or VMM, Virtual Machine Monitor)
    - Sits between the guest OSes and CPU hardware in SW stack
  - Very useful for workload consolidation

#### Xen

- "Xen and the Art of Virtualization"
- Xen is a hypervisor
- Has evolved into a widely used one to support virtualization
  - For x86 and ARM
  - IBM SoftLayer, Amazon EC2, Rackspace Cloud
- Now supported in many different forms
  - We'll discuss the fundamental principles described in the paper

## **VM Challenges**

- Need to partition machine to support concurrent execution of multiple operating systems
- 1. Virtual machines must be isolated from each other
  - For protection
  - So that performance of one does not affect performance of another
- 2. Must support a variety of different OSes
  - To enable workload consolidation
- 3. Performance overhead should be small
  - User applications should not slow down (much)

## Xen Approach

- Two main approaches to virtualization
  - Full Virtualization:
    - A virtual SW interface to all machine HW is exposed by VMM
    - Guest OS cannot access hardware directly
    - Advantages: can support unmodified guest OSes
    - Disadvantages: performance (all guest OS-HW interaction goes through VMM)
  - Paravirtualization:
    - This is what Xen uses
    - Exposes HW to the guest OSes where it is critical for performance
    - Exposed virtual SW interface to machine HW otherwise
    - Advantages: less slowdown for user applications
    - Disadvantages: requires some OS modification (but not to apps)
      - Supports same Application Binary Interface (ABI)

## Definitions

#### • Guest OS

- Refers to one of the OSes that can be hosted on the Xen VMM

• Domain

- A running instance of a VM within which a guest OS executes

- Think of as analogous to program vs. process
  - Static vs. dynamic

#### Paravirtualized x86 Interface

- 3 main aspects of the paravirtualized interface
  - Memory management
    - Paging
  - CPU
    - Protection, Exceptions, System Calls, Interrupts, Time
  - Device I/O
    - Network, Disk, etc.
- We've covered each of these 3
  - From the perspective of OS management
  - While all details mentioned in the paper may not be clear, the general concepts should seem familiar

#### **Memory Management Interface**

- This is the difficult part of paravirtualization
- Centers around TLB & page table management
- Two types of TLB
  - SW managed:
    - TLB miss traps to privileged SW; loads new entry from page tables
    - TLB entries tagged with address space IDs to avoid flushing the TLB when moving from CPU execution of one OS to another
  - HW managed (what x86 has):
    - HW in the processor handles TLB misses by "page table walk"
    - No address space IDs
      - Requires TLB flush on every address space switch

## **Memory Management Interface (2)**

- How does Xen do it?
  - Guest OSes do allocation and management of HW page tables
    - Xen is minimally involved to ensure OS safety and isolation
    - Guest OS can only map to memory it owns (physical frames)
    - When a new page table is required (e.g. a process is created):
      - Guest OS allocates & initializes a page that it owns
      - Registers this page with Xen (Xen validates the page table info)
      - Write permission to this page are removed from the guest OS
  - Xen is mapped into top 64MB of every address space
    - No TLB flush required when entering / exiting the VMM

#### CPU

- 4 privilege levels are supported in x86
  - Referred to as ring 0 ring 3 (from highest to lowest priority)
  - Normally, the OS runs in ring 0, user process in ring 3
  - Xen downgrades OS privilege to ring 1; Xen runs in ring 0
- Privileged instructions executed by the OS now fault and trap into the VMM (can only be executed within ring 0)
- Exceptions handled in straightforward way
  - Guest OS registers a table of exception handlers with Xen
  - When HW events occur requiring exception handling, Xen returns control to appropriate exception handler
- Special care for performance-sensitive exceptions:
  - System calls and page faults
  - "Fast handler" installed which is directly accessed by CPU
    - No need to pass through Ring 0 VMM first

# **Device I/O**

- Xen exposes a set of simple device abstractions
- I/O data is transferred between a domain & Xen
  - Via ring buffers
  - In asynchronous manner
- Lightweight event mechanism used to notify domains
  - E.g. of I/O completion or data availability

## **Porting Effort**

OS subsection	# lines	
	Linux	XP
Architecture-independent	78	1299
Virtual network driver	484	_
Virtual block-device driver	1070	_
Xen-specific (non-driver)	1363	3321
Total	2995	4620
(Portion of total x86 code base	1.36%	0.04%)

Table 2: The simplicity of porting commodity OSes to Xen. The cost metric is the number of lines of reasonably commented and formatted code which are modified or added compared with the original x86 code base (excluding device drivers).

#### **Control & Management**

- Separate policy vs. mechanism
  - Xen VMM implements mechanism (mostly)
  - Management control software handles policy
    - Running on a Guest OS
- Domain 0 hosts app-level management SW
- Control Interface:
  - Create & Terminate domains
  - Partition physical resources
    - Across domains
    - CPU, physical memory, disk, ...



## Xen / Guest OS Interaction

- Hypercalls: From a domain to Xen
  - Synchronous trap calls for service from a domain to Xen
  - Analagous to a system call from user process to OS
  - E.g. request page table updates
- Events: From Xen to a domain
  - Asynchronous notifications to a domain from Xen
  - Replaces device interrupts to an OS
  - Uses mechanism similar to UNIX signals (recall, like SW interrupts)
  - Pending events stored in a per-domain bitmask
  - Xen updates bit mask & invokes an event-callback handler specified by a guest OS
  - Guest OS handles events & resets bits

## I/O Data Transfer

Domains allocate shared buffers for device I/O



Unused descriptors

## **CPU Scheduling**

- Schedules domains on the CPU
  - According to some set policy to set allocation per domain
  - Uses a scheduling algorithm called Borrowed Virtual Time
- Per-domain scheduling parameters can be adjusted by the management software running in Domain0

#### **Time and Timers**

- Xen provides 3 views of time to domains
  - Real time
    - ns since machine boot time
  - Virtual time
    - A time that advances only while domain is executing
    - Can be used by guest OS to ensure it gets correct timeshare
  - Wall clock
    - Offset added to current real time

#### **Evaluation**



Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

## Virtualization beyond Xen: x86 extensions

- Modern x86 CPUs have hardware support for virtualization
- Intel virtualization (VT-x)
  - Adds instructions for dealing with virtualization
  - Allows guest OS to think it's ring 0 but still have key operations trapped to real VMM (removes needs for paravirtualization)
  - Most modern virtualization uses this instead of Xen-style paravirtualization (but principles are much the same; the CPU just does most of that work now)
- I/O MMU virtualization (Intel VT-d)
  - Allows guest VMs to directly access hardware peripherals
  - Example: Allow a VM to use a video card GPU