

Operating Systems Design

20. Virtualization

Paul Krzyzanowski
pxk@cs.rutgers.edu

Virtualization

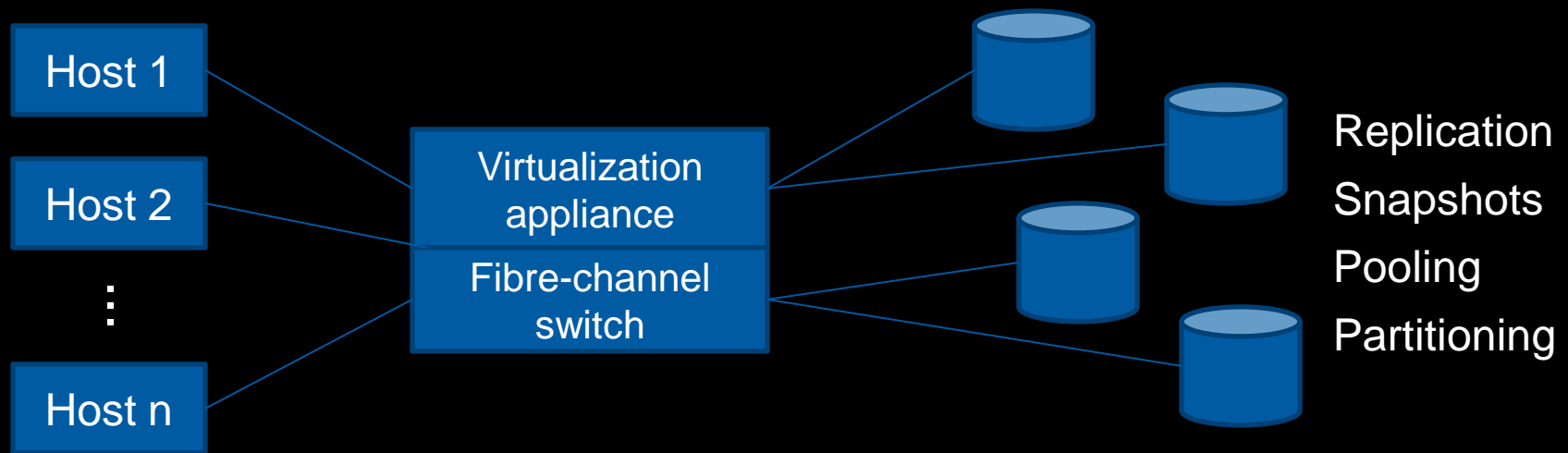
- Memory virtualization
 - Process feels like it has its own address space
 - Created by MMU, configured by OS
- Storage virtualization
 - Logical view of disks “connected” to a machine
 - External pool of storage
- CPU/Machine virtualization
 - Each process feels like it has its own CPU
 - Created by OS preemption and scheduler

Storage Virtualization

- Dissociate knowledge of physical disks
- Software between the computer and the disks manages the view of storage
- Examples:
 - Make four 500 GB disks appear as one 2 TB disk
 - Make one 500 GB disk appear as two 200 GB disks and one 100 GB disk, with each of the 200 GB virtual disks available to different servers while the 100 GB disk can be shared by all.
 - Have all writes get mirrored to a backup disk
- Virtualization software translates read-block/write-block requests for logical devices to read-block/write-block requests for physical devices

Storage Virtualization

- Logical view of disks “connected” to a machine
- Separate logical view from physical storage
- External pool of storage



Virtual CPUs (sort of)

- Each process feels like it has its own CPU
 - But cannot execute privileged instructions (e.g., modify the MMU or the interval timer, halt the processor, access I/O)
- Created by OS preemption and scheduler

Process Virtual Machines

- CPU interpreter running as a process
- Pseudo-machine with interpreted instructions
 - 1966: O-code for BCPL
 - 1973: P-code for Pascal
 - 1995: Java Virtual Machine (JIT compilation added)
 - 2002: Microsoft .NET CLR (pre-compilation)
 - 2008: Dalvik VM for Android
- Advantage: run anywhere, sandboxing capability

Machine Virtualization

- Machine virtualization
 - Partition a physical computer to act like several real machines
 - Migrate an entire OS + applications from one machine to another

- 1972: IBM System 370

Machine Virtualization

- **Privileged** vs. **unprivileged** instructions
- Regular applications use unprivileged instructions
 - Easy to virtualize
- If regular applications execute privileged instructions, they **trap**
- VM catches the trap and emulates the instruction

Intel Ugliness

- Intel x86 arch < Core 2 Duo doesn't support trapping privileged instructions
- Two approaches
 - **Binary translation** (BT)
 - Scan instruction stream and replace privileged instructions with something the VM can intercept.
(VMware approach)
 - **Paravirtualization**
 - Don't use non-virtualizable instructions (Xen approach)

Virtual Machine Monitor

- Program in charge of virtualization
 - Aka **Hypervisor**
 - Arbitrates access to physical resources
 - Presents a set of virtual device interfaces to each host
- Guest OS runs until:
 - Privileged instruction traps
 - System interrupts
 - Exceptions (page faults)
 - Explicit call: VMCALL (intel) or VMCALL (AMD)

Two Approaches

- **Hosted VM**

- Primary OS responsible for access to the raw machine
- Guest operating systems run under a VMM
- VMM invoked by host OS
 - Serves as a proxy to the host OS for access to devices

- **Native VM**

- No primary OS
- Hypervisor is in charge of access to the devices and scheduling

Architectural Support

- Intel Virtual Technology (Intel Core 2 Duo)
 - AMD Opteron
-
- Certain privileged instructions are intercepted as VM exits to the VMM
 - Exceptions, faults, and external interrupts are intercepted as VM exits
 - Virtualized exceptions/faults are injected as VM entries

Hypervisor in a hosted VM

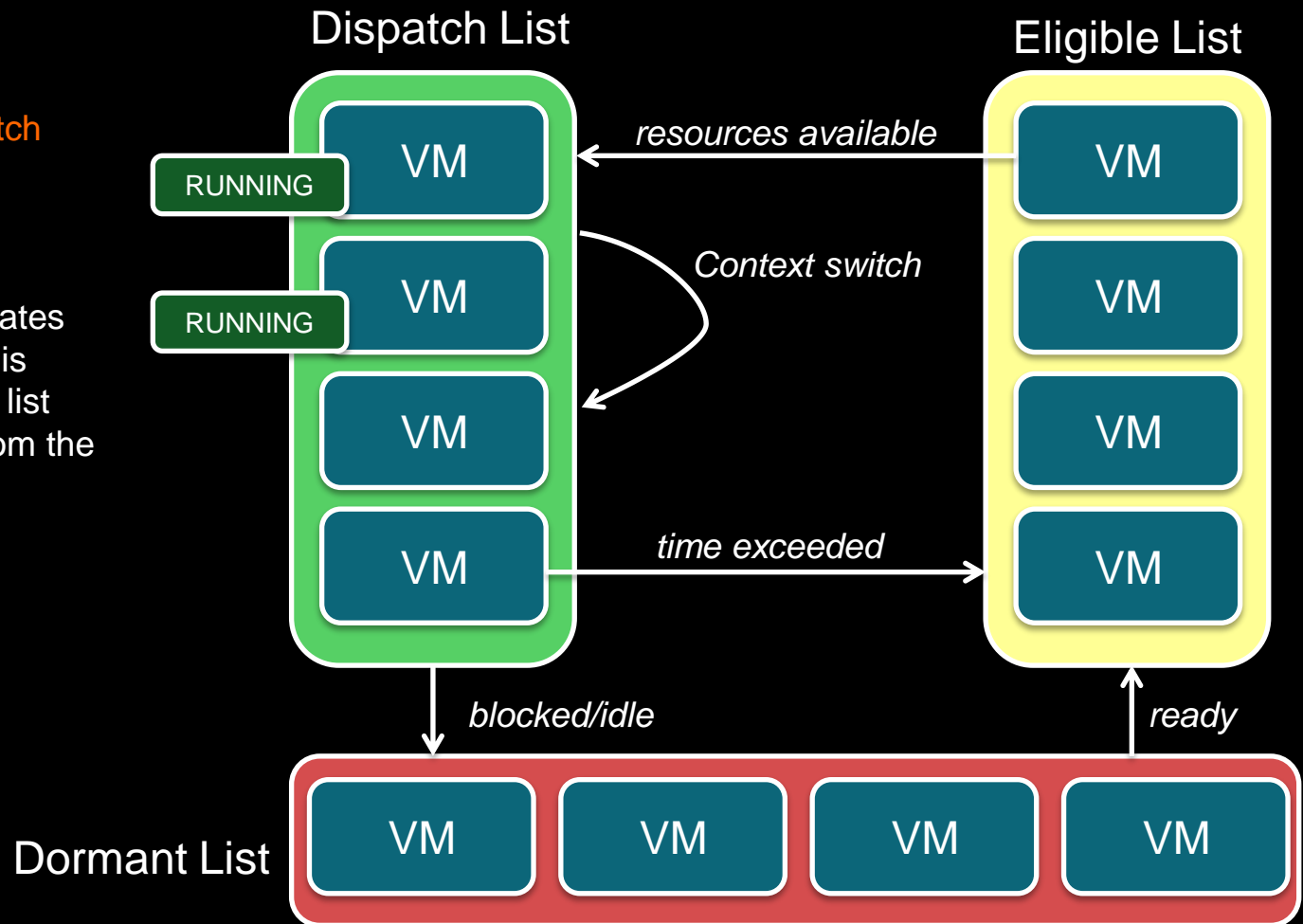
- Responsible for managing access to all system resources
 - Memory, disks, buses, I/O adapters
 - CPU scheduling
- Guest OS gets device drivers that interface to an abstract device implementation provided by the VMM

Hypervisor in a hosted VM

- Each VM competes for a physical CPU
 - Typically # VMs > # CPUs
- VMs need to get scheduled
 - Allocate CPU to a single-CPU VM
 - Allocate multiple CPUs to multi-CPU VMs: **co-scheduling**
 - Strict co-scheduler: VM with two virtual CPUs gets two real CPUs
 - Relaxed co-scheduler: if two CPUs are not available, use one
 - CPU affinity: try to run the VM on the same CPU
- VM scheduler controls the level of multiprogramming of VMs

Scheduling VMs (IBM z/VM example)

- Run until a VM accumulates a **dispatch time slice**
- Readjust priority
- When a VM accumulates **elapsed time slice**, it is moved to the eligible list and a VM is taken from the eligible list



OS-Level Virtualization

- Not full machine virtualization
- Multiple instances of the same operating system
 - Each has its own environment
 - Process list, mount table, file descriptors, virtual network interface
- Advantage: low overhead: no overhead to system calls

- Examples:
 - Linux Vserver, Solaris Containers, FreeBSD Jails
 - Altris Software Virtualization Services
 - Windows registry & directory tweaking
 - Allows multiple instances of applications to be installed

Popular VM Platforms

- VMWare
- Xen
 - Runs under an OS and provides virtual containers for running other operating systems. Runs a subset of x86. Routes all hardware accesses to the host OS.
- VirtualBox
- Microsoft Virtual Server
- Parallels
- IBM z/VM (mainframe)
- Sun xVM

Security Threats

- Hypervisor-based rootkits
- A system with no virtualization software installed but with hardware-assisted virtualization can have a hypervisor-based rootkit installed.
- Rootkit runs at a higher privilege level than the OS.
 - It's possible to write it in a way that the kernel will have a limited ability to detect it.

The End