INFSCI 2750: Cloud Computing

Lecture 5: Virtual Machine Management in Clouds

Dr. Balaji Palanisamy
Associate Professor
School of Computing and Information
University of Pittsburgh
bpalan@pitt.edu

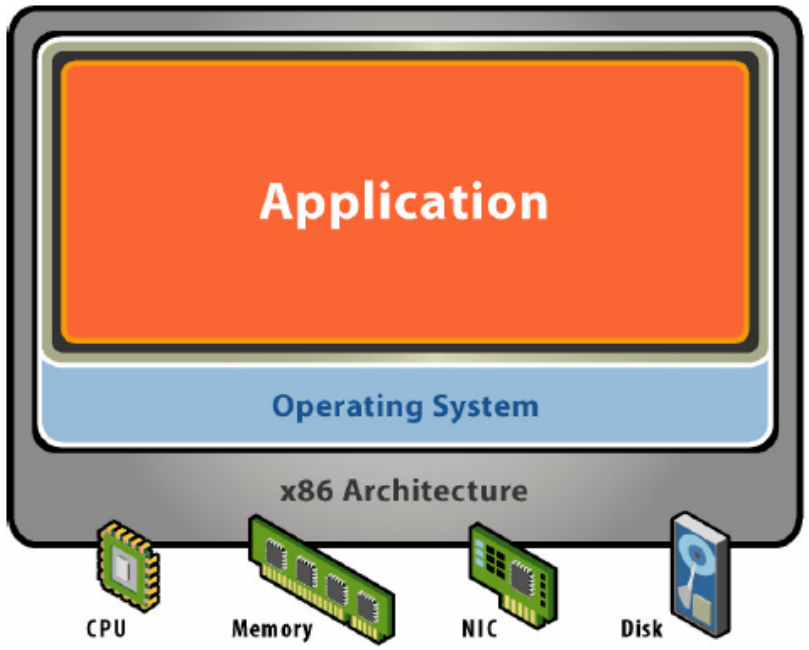
Slides Courtesy: Prof. Keke Chen, Wright State University
Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007

Virtualization

- Virtualization deals with "extending or replacing an existing interface so as to mimic the behavior of another system"

- Virtual system examples: virtual private network, virtual memory, virtual machine

Starting Point: A Physical Machine



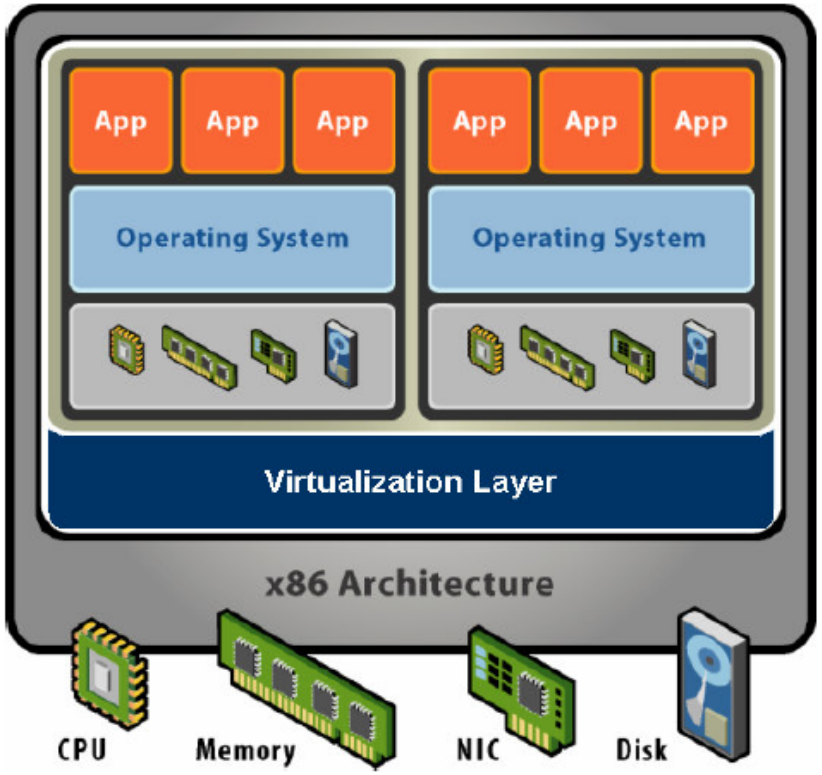
Physical Hardware

- Processors, memory, chipset, I/O bus and devices, etc.
- Physical resources often underutilized

Software

- Tightly coupled to hardware
- Single active OS image
- OS controls hardware

What is a Virtual Machine?



Hardware-Level Abstraction

- Virtual hardware: processors, memory, chipset, I/O devices, etc.
- Encapsulates all OS and application state

Virtualization Software

- Extra level of indirection decouples hardware and OS
- Multiplexes physical hardware across multiple “guest” VMs
- Strong isolation between VMs
- Manages physical resources, improves utilization

VM Isolation

Secure Multiplexing

- Run multiple VMs on single physical host
- Processor hardware isolates VMs, *e.g.* MMU

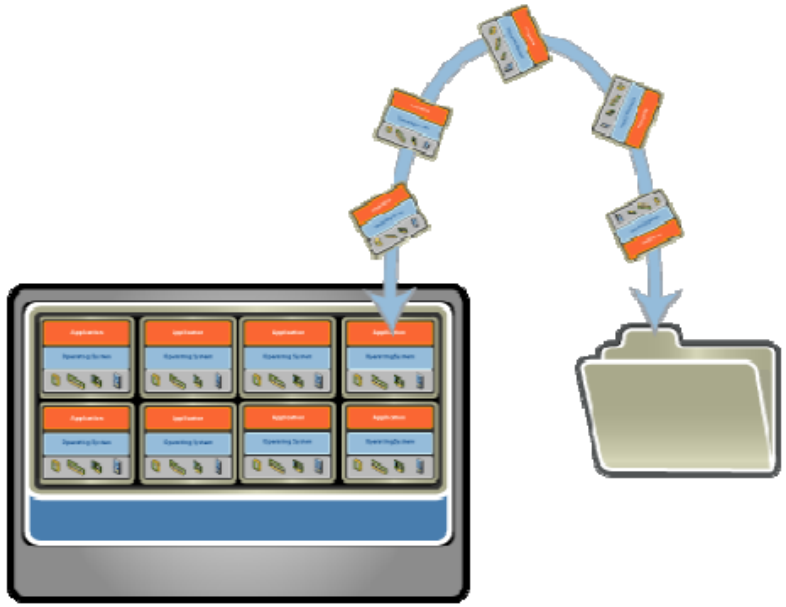
Strong Guarantees

- Software bugs, crashes, viruses within one VM cannot affect other VMs

Performance Isolation

- Partition system resources
- Example: VMware controls for reservation, limit, shares

VM Encapsulation



Entire VM is a File

- OS, applications, data
- Memory and device state

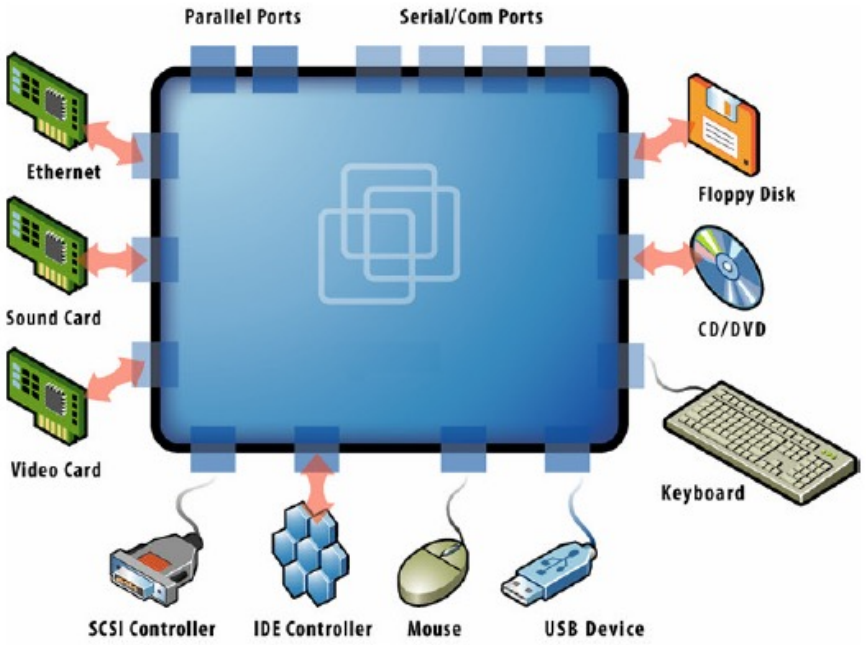
Snapshots and Clones

- Capture VM state on the fly and restore to point-in-time
- Rapid system provisioning, backup, remote mirroring

Easy Content Distribution

- Pre-configured apps, demos
- Virtual appliances

VM Compatibility



Hardware-Independent

- Physical hardware hidden by virtualization layer
- Standard virtual hardware exposed to VM

Create Once, Run Anywhere

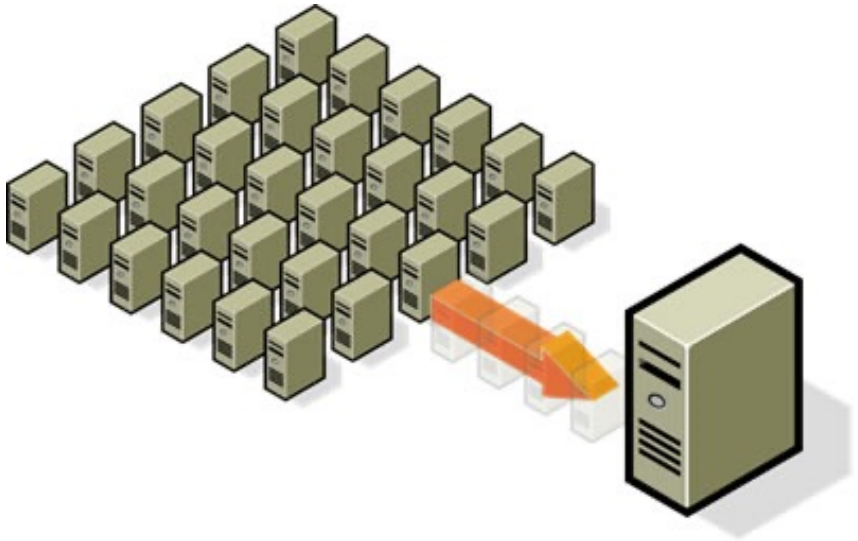
- No configuration issues
- Migrate VMs between hosts

Legacy VMs

- Run ancient OS on new platform
- *E.g.* DOS VM drives virtual IDE and vLance devices, mapped to modern SAN and GigE hardware

Common Virtualization Uses Today

- Reduce costs by consolidating services onto the fewest number of physical machines



http://www.vmware.com/img/serverconsolidation.jpg

Dynamic Data Center

- Virtualization helps us break the "one service per server" model
- Consolidate many services into a fewer number of machines when workload is low, reducing costs
- Conversely, as demand for a particular service increases, we can shift more virtual machines to run that service
- We can build a data center with fewer total resources, since resources are used as needed instead of being dedicated to single services

Outline

- Motivation and introduction
- Example: Xen
  - techniques
  - Evaluation

What is virtualization

- Partitioning one physical server to multiple virtual servers
  - Virtual machines are isolated
  - One VM failure will not affect the others
- Hypervisor software is the key
  - Or called virtualization manager
  - The layer between the hardware/OS and virtual machines
  - Manages the partitioning and isolation of system resources

Broader concept of virtualization

- Combine multiple physical resources into one single virtual resource
  - Storage virtualization
- Application virtualization: JVM, .Net
- Network virtualization
- Desktop virtualization

Benefits

- **Save money.**
  - Many companies require one app on one machine for reliability
- **Save energy**
  - Less physical servers, less energy consumption
- **Save time**
  - Deploy, setup, startup quickly
- **Agile development**
  - Developer can use multiple virtual OSes to simulate and test cross-platform software

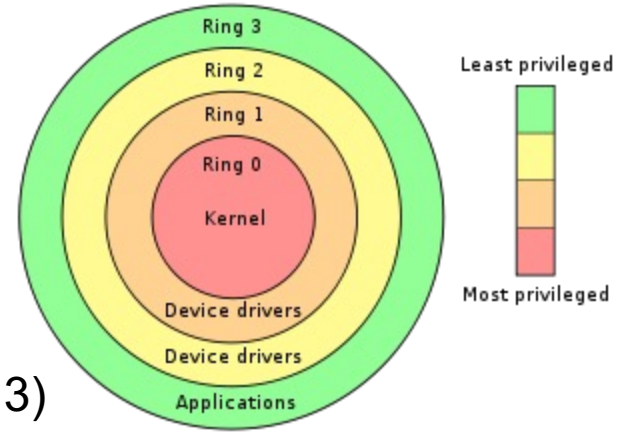
Basic ideas

- Virtualize resources
  - CPU
  - Memory
  - Network
  - Disk

- Key: the layer between hardware and guest OSs – hypervisor software
  - Partitioning, isolating, and scheduling resources between guest Oss

Preliminary (normal OS)

Protection rings



APPS

User space (lower privilege: ring 3)

System call/ trap

Kernel space (high privilege: ring 0)

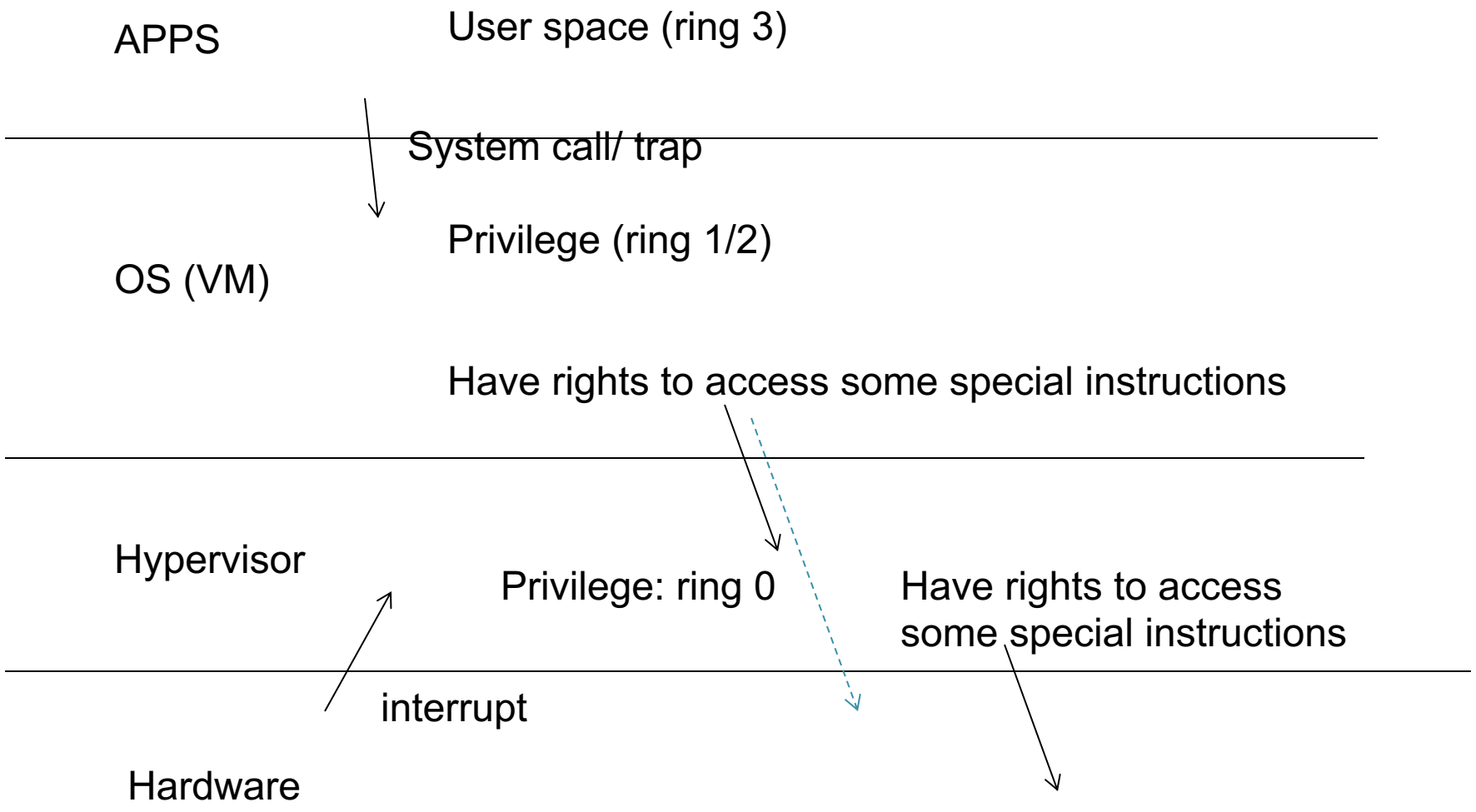
OS
(supervisor mode)

Have rights to access some special CPU instructions

interrupt

Hardware

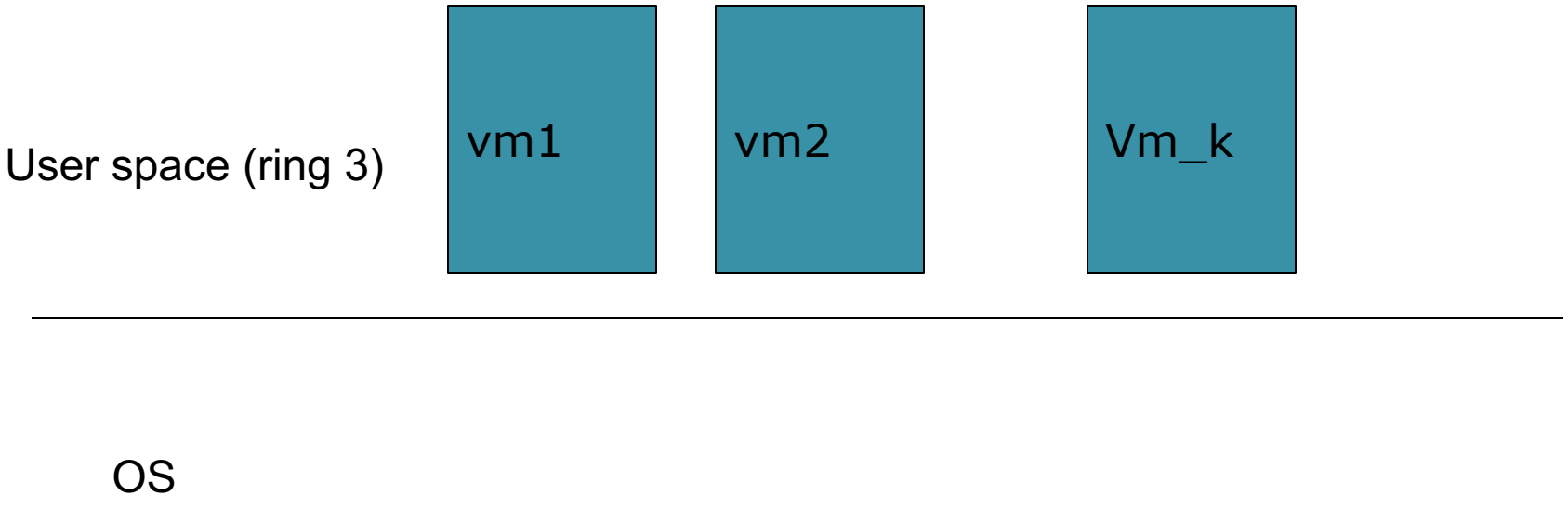
x86 virtualization



Types of virtualization

- Container virtualization
- Full virtualization
- Para-virtualization

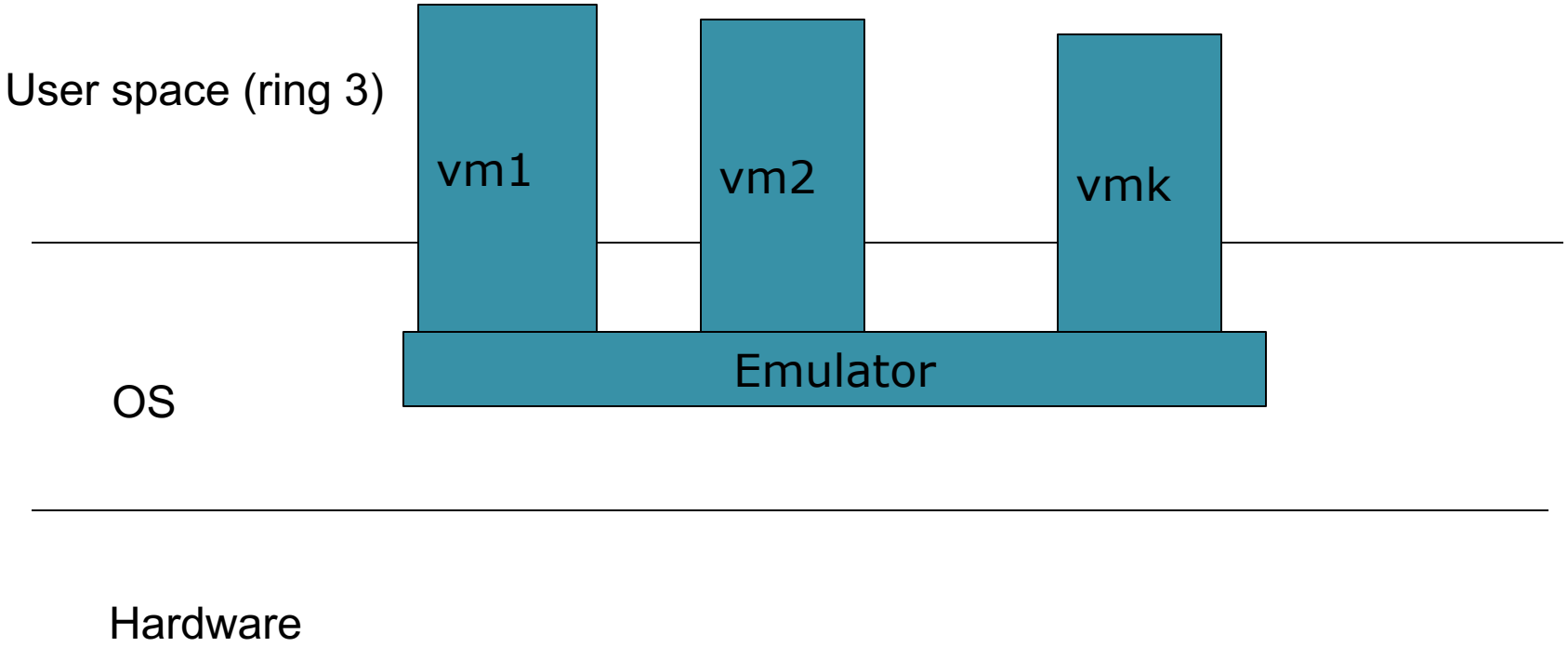
Container virtualization



Container virtualization

- User-space virtual machines
- All guests share the same filesystem tree.
- Same kernel on all virtual machines
- Unprivileged VMs can't mount drives or change network settings
- Provide extra-level of security
- Native Speeds, no emulation overhead
- OpenVZ, Virtuozzo, Solaris Containers, FreeBSD Jails, Linux-Vserver

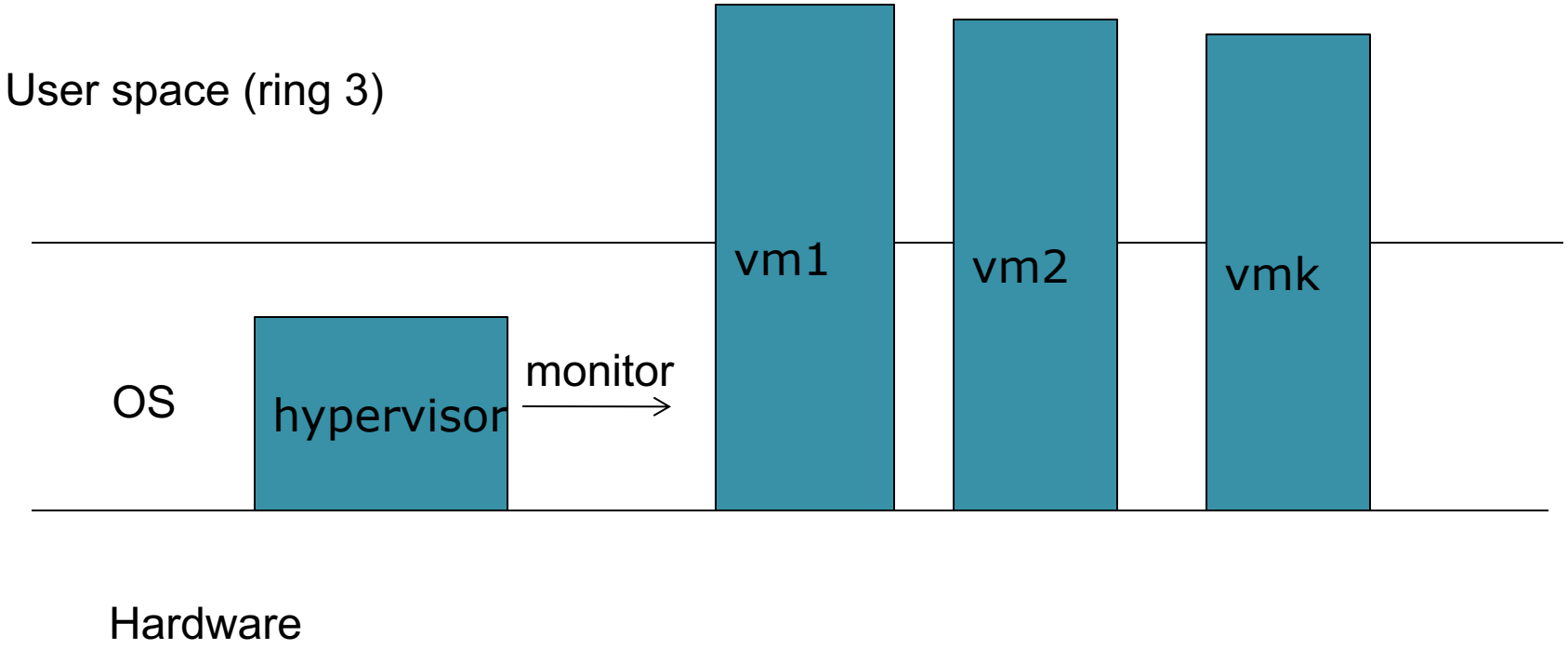
Full virtualization



Full virtualization

- Runs **unmodified** guests
- Simulates bios, BIOS emulation, sometimes custom drivers
  - Guests cannot access hardware
- Generally worst performance, but often acceptable

Paravirtualization



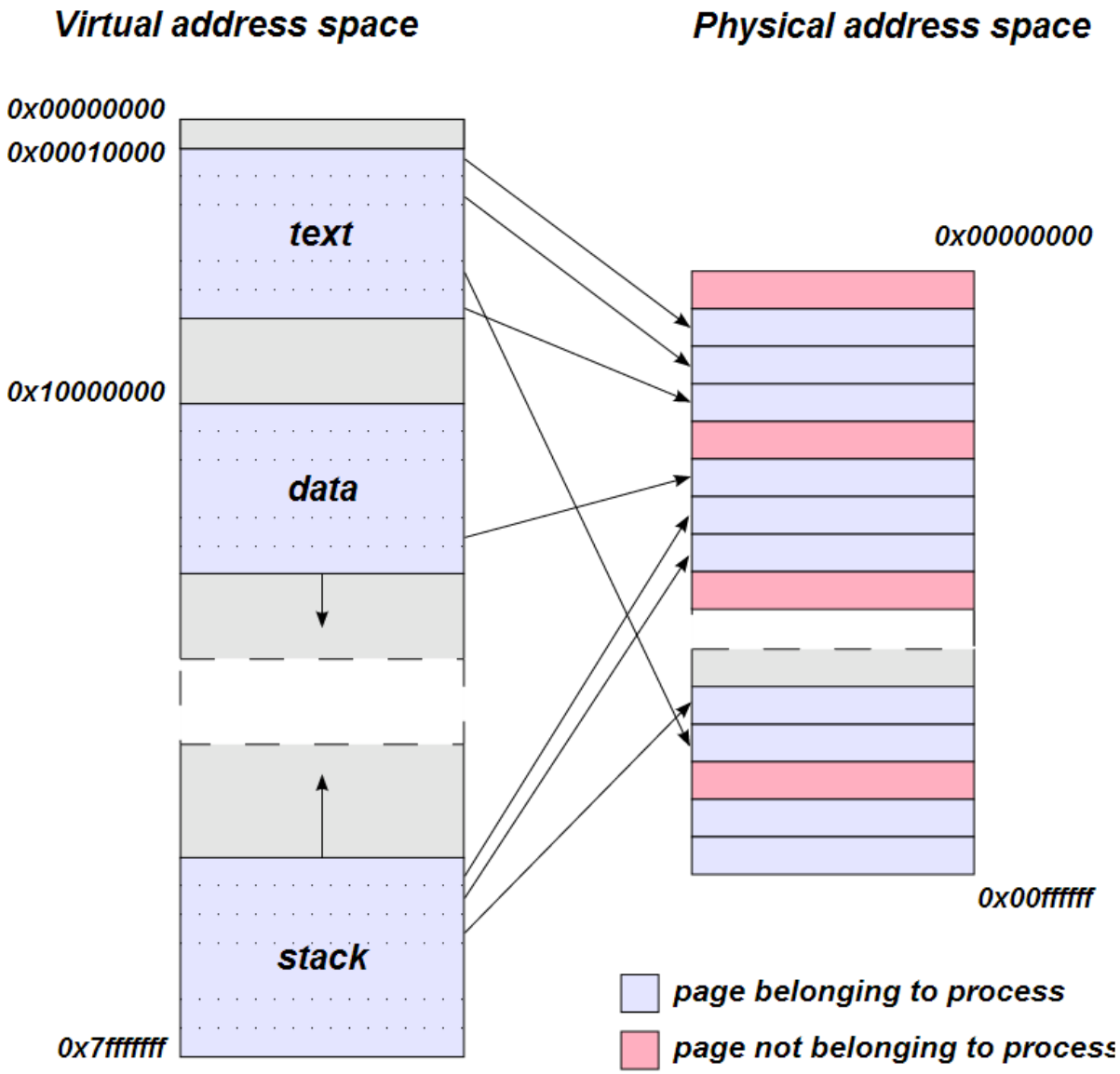
Paravirtualization

- Do not try to emulate everything
  - Work as a guard
  - Pass safe instructions directly to CPU and device
  - Guests have some exposure to the hardware
- Better performance
- Need to slightly modify guest OS, but no need to modify applications
- Xen, Sun Logical Domains

Xen: introduction

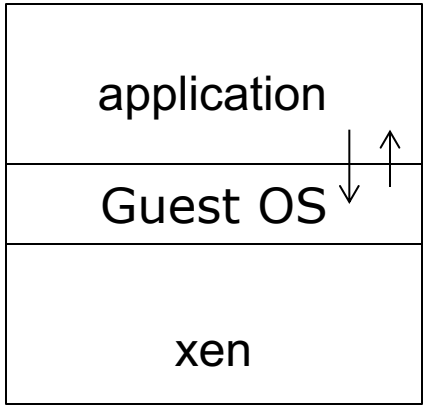
- Paravirtualization
- Faster than full virtualization
- Need to slightly change some guest OS
- Domain (1-) : guest OS

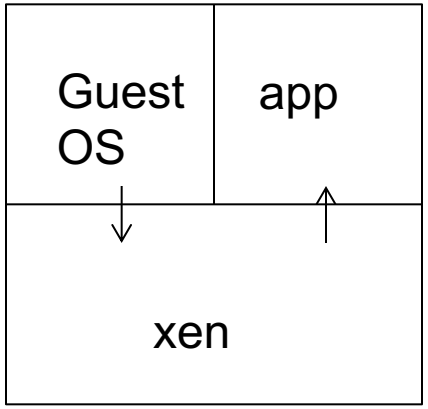
virtual memory management



Xen: CPU scheduling

- Guest OS runs at a lower privilege level than Xen
- Guest OS must register exception (trap) handlers with Xen
  - Xen will check the handler
  - Page fault is handled differently
- System calls : no Xen intervention
- Use a lightweight event system to handle hardware interrupts





More than two privilege levels

only two privilege levels for some processors

X86 provides 4 levels of privilege – rings
Xen at ring 0, guest OS at ring 1, apps at ring 3

- Two types of frequent exception
  - System calls
  - Page faults
- Improve performance of system calls
  - A fast exception handler accessed directly by the processor without via ring 0; validated before installing it in the hardware exception table
  - Validation: check the handler's code segment – no execution in ring 0

Xen: device I/O

- Only Domain0 has direct access to disks
- Other domains need to use virtual block devices
  - Use the I/O ring
  - Reorder requests prior to enqueuing them on the ring
  - use DMA (zero copy)

Xen: network

- Virtual firewall-router attached to all domains
- To send a packet, enqueue a buffer descriptor into the I/O ring
- Use DMA (no packet copying)

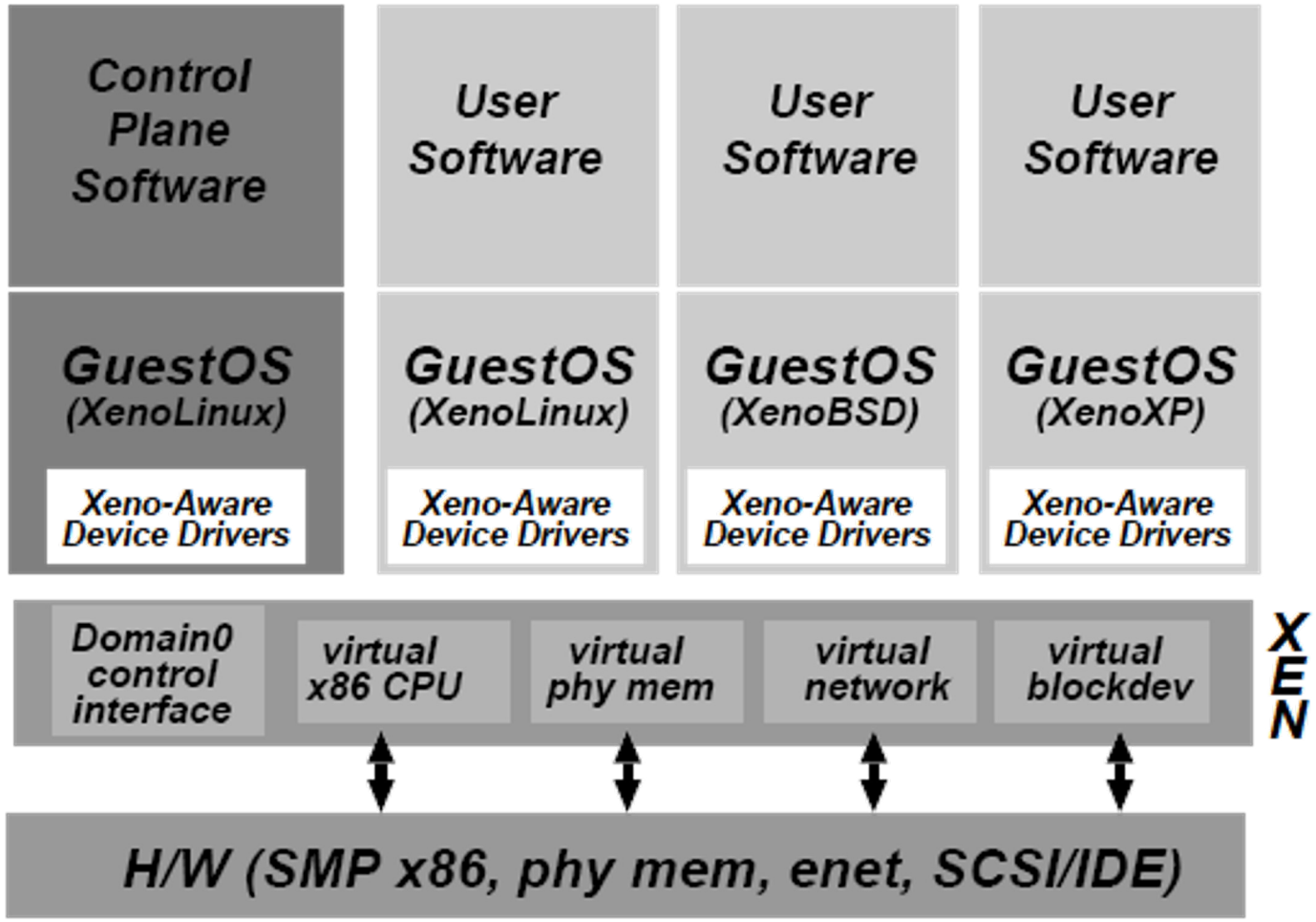
Partitioning resources between guest OSes

- Memory- preallocated physical memory
- Disk – quota
- CPU and network
  - Involves more complicated procedures

Domain 0

- The representative to the Xen hypervisor
- Provide bootstrap code for different types of VMs
- Creating/deleting virtual network interfaces and virtual block devices for other domains

System looks like



Cost of porting a guest OS to Xen

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%)**

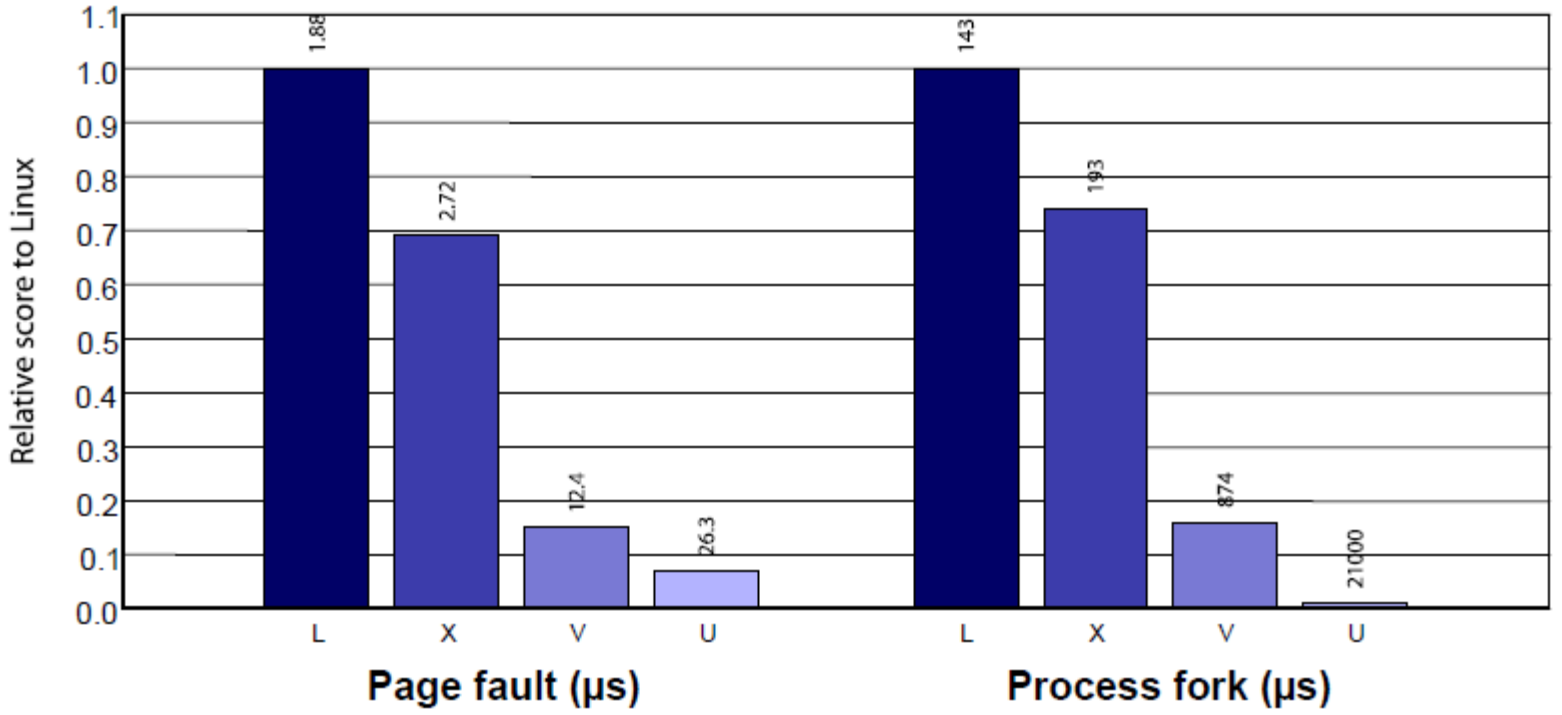
Table 2: The simplicity of porting commodity OSes to Xen. The cost metric is the number of lines of reasonably commented and formatted code which are modified or added compared with the original x86 code base (excluding device drivers).

Linux kernel 2.4

Xen: performance

- Hardware (2003)
  - Dell 2650 dual processor
  - 2.4 GHz Xeon server
  - 2GB RAM
  - 3 Gb Ethernet NIC
  - 1 Hitachi DK32eJ 146 GB 10k RPM SCSI disk
  - Linux 2.4.21 (native)

MMU (memory management) performance



Imbench results on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)

Various benchmarks

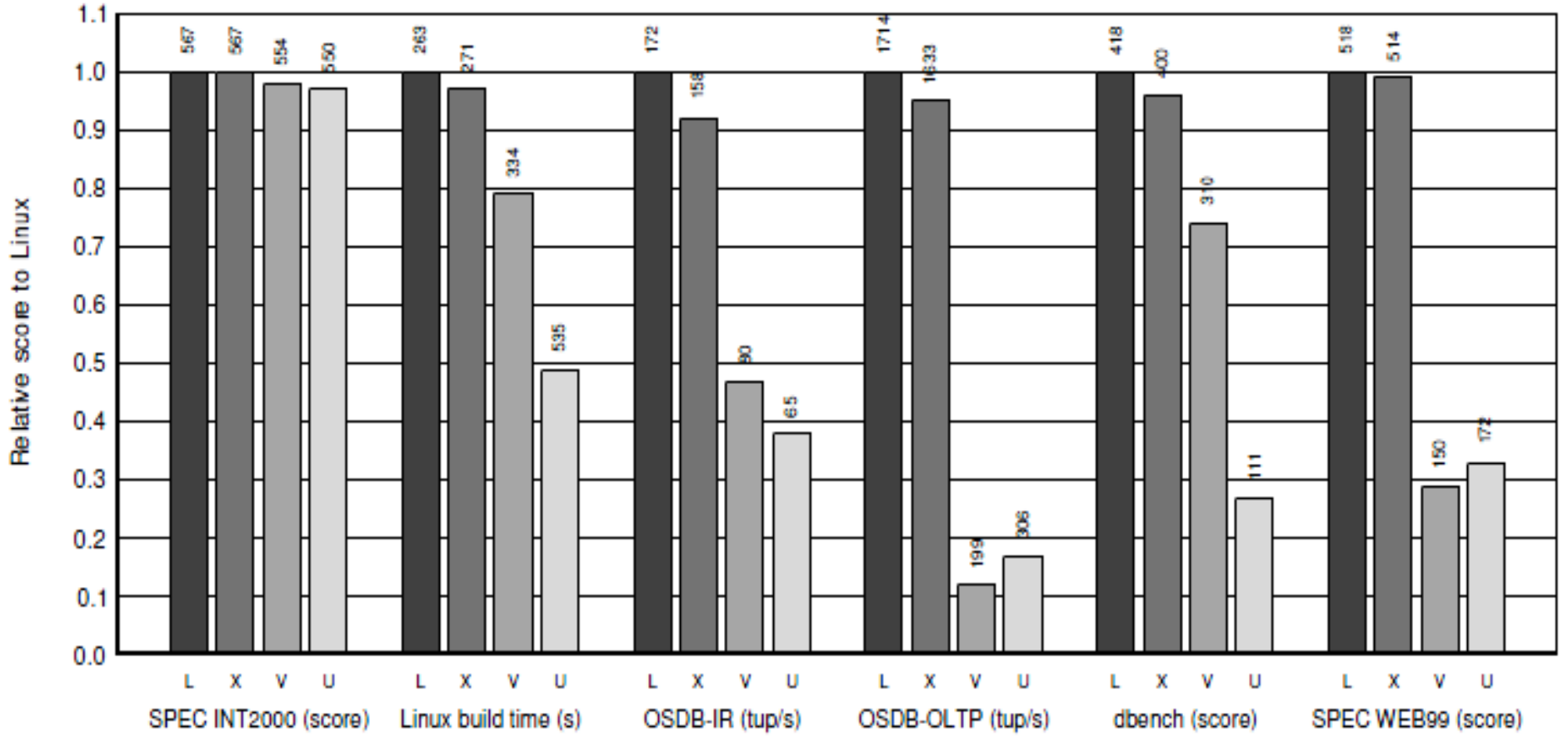


Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

Issues

- Performance isolation vs. maximizing overall system utilization
  - Easy to partition memory and disk
  - Not easy to partition CPU and network
    - Time issue

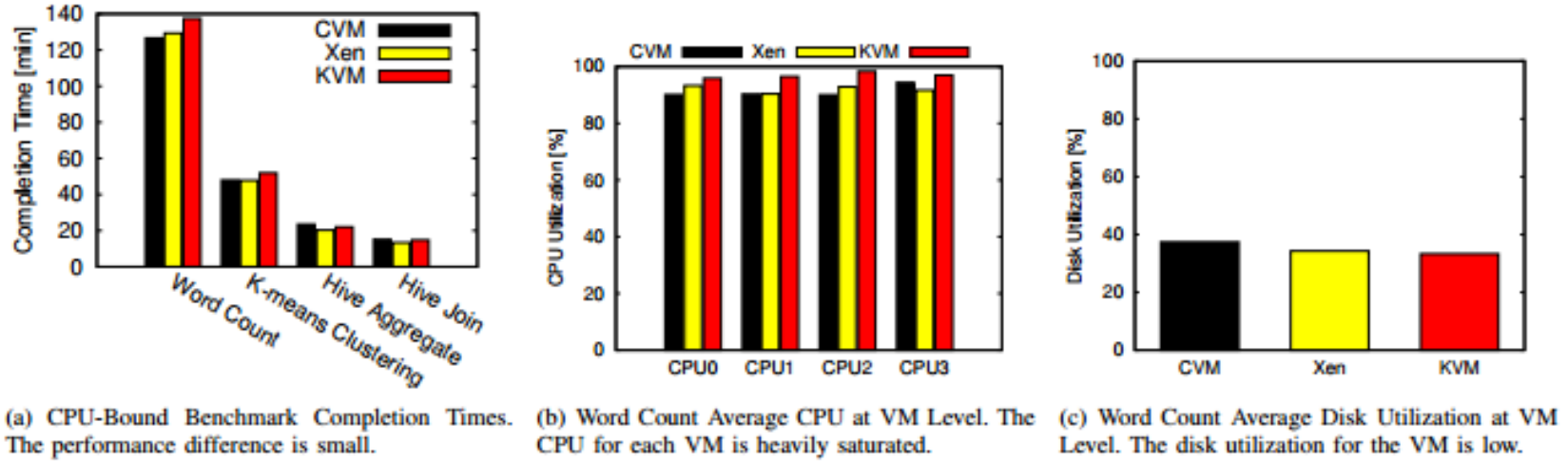
Recent development

- Kernel based virtual machine (KVM)
  - A part of the linux kernel (vs. Xen as a standalone hypervisor
  - 2008 result

Table 1. Overall performance of base Linux, Xen, and KVM

	Linux	Xen	KVM
CPU	1.000	0.999	0.993
Kernel Compile	1.000	0.487	0.384
IOzone Write	1.000	0.855	0.934
IOzone Read	1.000	0.852	0.994

- Hadoop workloads (2013)



(a) CPU-Bound Benchmark Completion Times. The performance difference is small.

(b) Word Count Average CPU at VM Level. The CPU for each VM is heavily saturated.

(c) Word Count Average Disk Utilization at VM Level. The disk utilization for the VM is low.

Fig. 1: CPU-Bound Benchmark Results and Word Count Statistics. The performance difference for these benchmarks as seen in Figure 1(a) is negligible between the different hypervisors. A representative benchmark, Wordcount, shows high CPU utilization and low disk utilization during the job as seen in Figure 1(b) and 1(c).

Timothy Wood, Prashant Shenoy,
Arun Venkataramani, and Mazin Yousif*

University of Massachusetts Amherst
*Intel, Portland

Black-box and Gray-box Strategies for Virtual Machine Migration

Enterprise Data Centers

- Data Centers are composed of:
  - Large clusters of servers
  - Network attached storage devices
- Multiple applications per server
  - Shared hosting environment
  - Multi-tier, may span multiple servers
- Allocates resources to meet Service Level Agreements (SLAs)
- Virtualization increasingly common

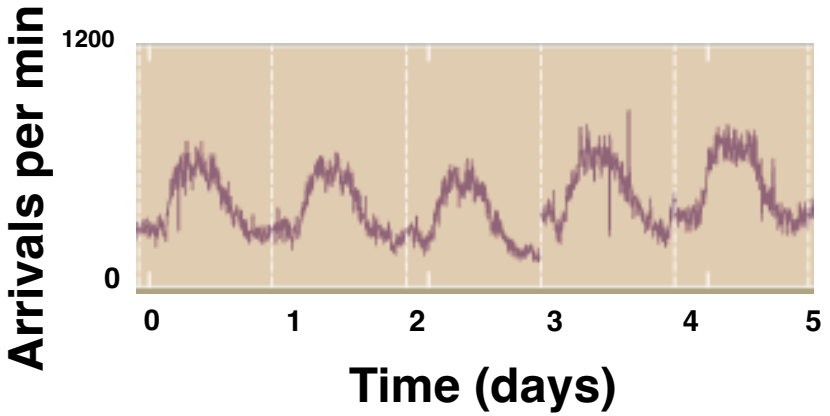
Benefits of Virtualization

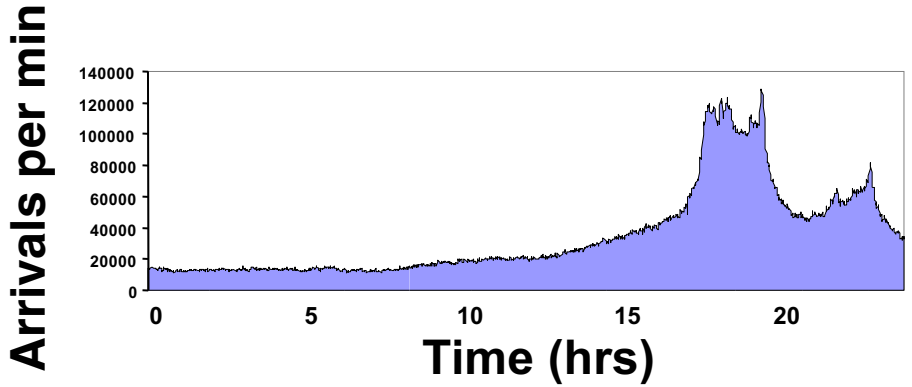
- Run multiple applications on one server
  - Each application runs in its own virtual machine
- Maintains isolation
  - Provides security
- Rapidly adjust resource allocations
  - CPU priority, memory allocation
- VM migration
  - "Transparent" to application
  - No

How can we use virtualization to more efficiently utilize data center resources?

Data Center Workloads

- Web applications see highly dynamic workloads
  - Multi-time-scale variations
  - Transient spikes and flash crowds





How can we provision resources to meet these changing demands?

Provisioning Methods

- **Hotspots** form if resource demand exceeds provisioned capacity

- Static over-provisioning
  - Allocate for peak load
    - Wastes resources
    - Not suitable for dynamic workloads
    - Difficult to predict peak resource requirements

- Dynamic provisioning
  - Adjust based on workload
    - Often done manually
    - Becoming easier with virtualization

Problem Statement

How can we automatically detect and eliminate hotspots in data center environments?

Use VM migration and dynamic resource allocation!

Outline

- Introduction & Motivation
- **System Overview**
- When? How much? And Where to?
- Implementation and Evaluation
- Conclusions

Research Challenges

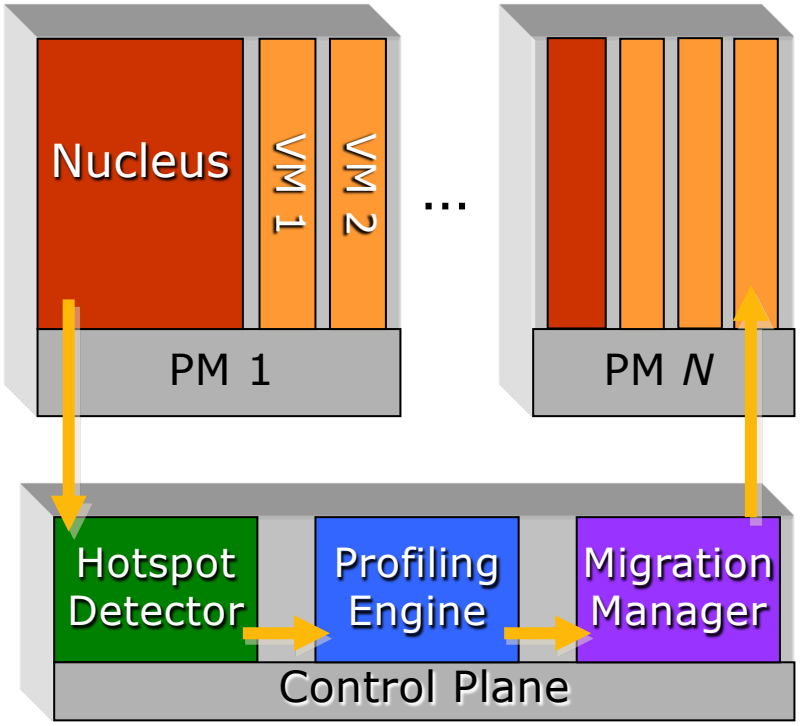
- Sandpiper: automatically detect and mitigate hotspots through virtual machine migration

A migratory bird

- *When* to migrate?
- *Where* to move to?
- *How much* of each resource to allocate?
- How much information needed to make decisions?

Sandpiper Architecture

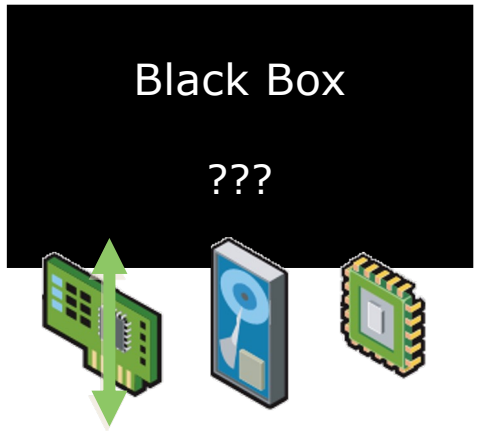
- Nucleus
  - *Monitor* resources
  - Report to control plane
  - One per server
- Control Plane
  - Centralized server
- Hotspot Detector
  - Detect *when* a hotspot occurs
- Profiling Engine
  - Decide *how much* to allocate
- Migration Manager
  - Determine *where* to migrate

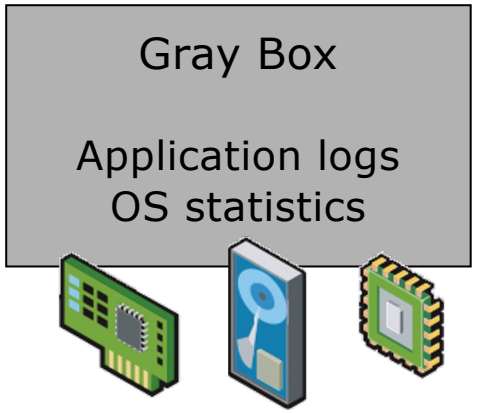


PM = Physical Machine
VM = Virtual Machine

Black-Box and Gray-Box

- Black-box: only data from outside the VM
  - Completely OS and application agnostic





- Gray-Box: access to OS stats and application logs
  - Request level data can improve detection and profiling
  - Not always feasible – customer may control OS

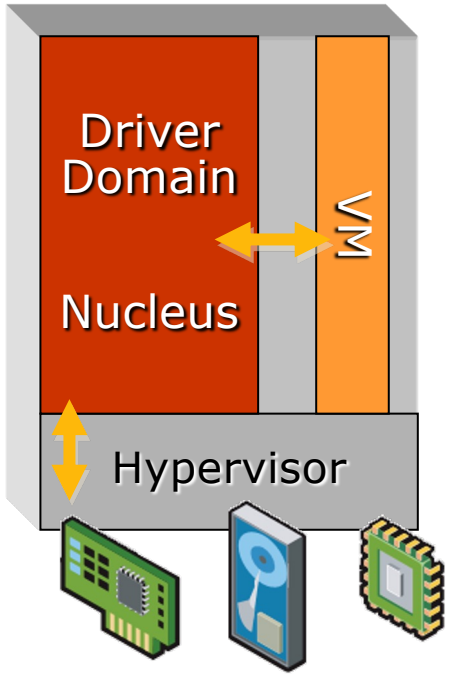
Is black-box sufficient?
What do we gain from gray-box data?

Outline

- Introduction & Motivation
- System Overview
- **When? How much? And Where to?**
- Implementation and Evaluation
- Conclusions

Black-box Monitoring

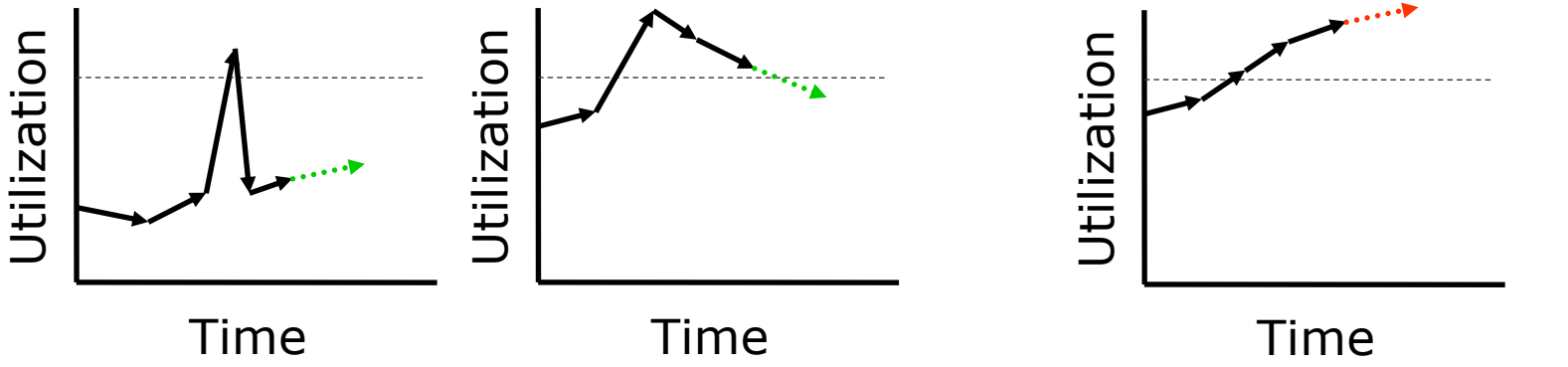
- Xen uses a "Driver Domain"
  - Special VM with network and disk drivers
  - Nucleus runs here



- CPU
  - Scheduler statistics
- Network
  - Linux device information
- Memory
  - Detect swapping from disk I/O
  - Only know when performance is poor

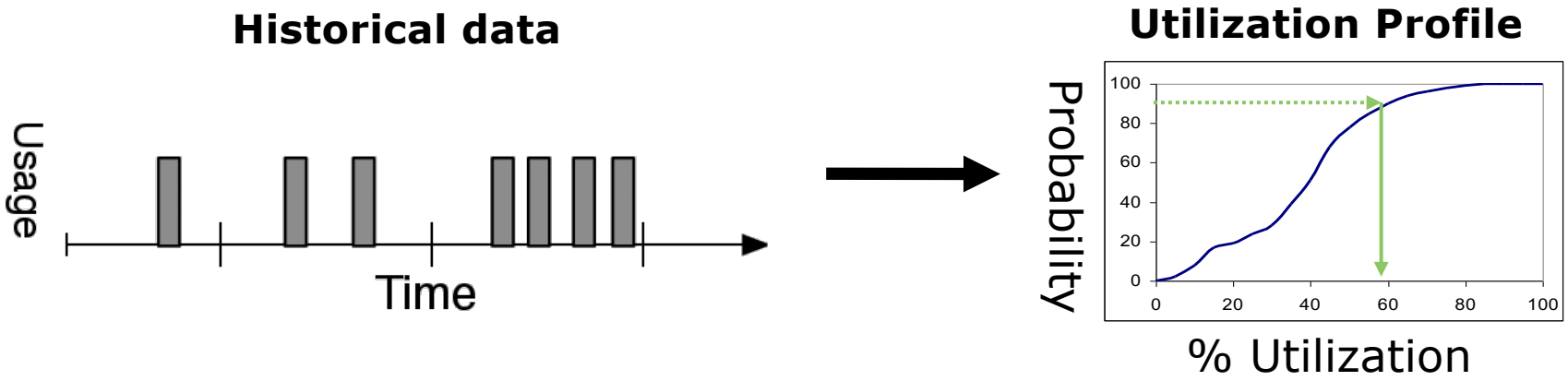
Hotspot Detection – When?

- Resource Thresholds
  - Potential hotspot if utilization exceeds threshold
- Only trigger for *sustained* overload
  - Must be overloaded for k out of n measurements
- Autoregressive Time Series Model
  - Use historical data to predict future values
  - Minimize impact of transient spikes



Resource Profiling – How much?

- How much of each resource to give a VM
  - Create distribution from time series
  - Provision to meet peaks of recent workload

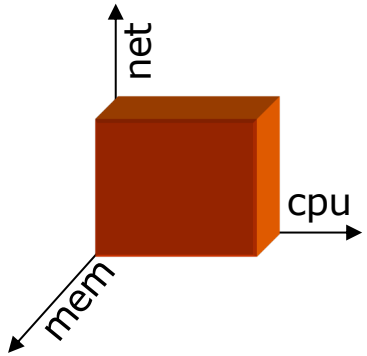


- What to do if utilization is at 100%?
- Gray-box
  - Request level knowledge can help
  - Can use application models to determine requirements

Determining Placement – Where to?

- Migrate VMs from overloaded to underloaded servers

$$Volume = \frac{1}{1\text{-cpu}} * \frac{1}{1\text{-net}} * \frac{1}{1\text{-mem}}$$



- Use *Volume* to find most loaded servers
  - Captures load on multiple resource dimensions
  - Highly loaded servers are targeted first

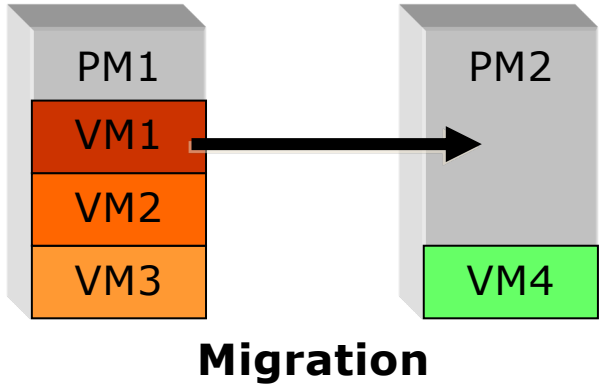
- Migrations incur overhead
  - Migration cost determined by RAM
  - Migrate the VM with highest Volume/RAM ratio

Maximize the amount of load transferred while minimizing the overhead of migrations

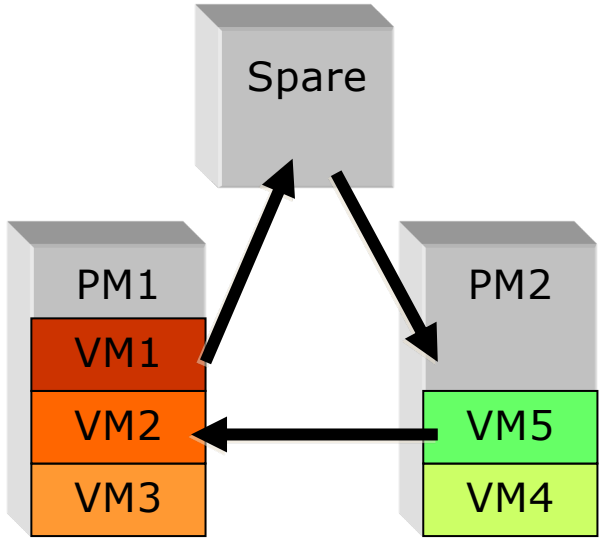
Placement Algorithm

- First try **migrations**
  - Displace VMs from high *Volume* servers
  - Use *Volume/RAM* to minimize overhead
- Don't create new hotspots!
  - What if high average load in system?
- **Swap** if necessary
  - Swap a high *Volume* VM for a low *Volume* one
  - Requires 3 migrations
    - Can't support both at once

Swaps increase the number of hotspots we can resolve



Migration



Swap

Outline

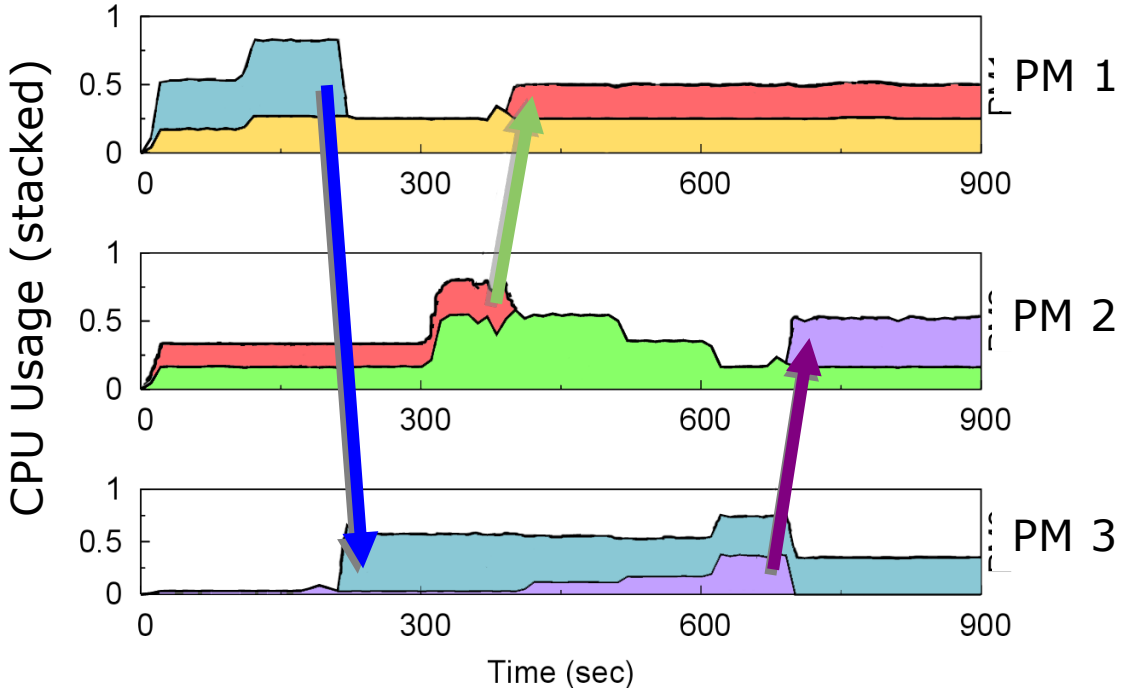
- Introduction & Motivation
- System Overview
- When? How much? And Where to?
- **Implementation and Evaluation**
- Conclusions

Implementation

- Use Xen 3.0.2-3 virtualization software
- Testbed of twenty 2.4Ghz P4 servers
- Apache 2.0.54, PHP 4.3.10, MySQL 4.0.24
- Synthetic PHP applications
- RUBiS – multi-tier ebay-like web application

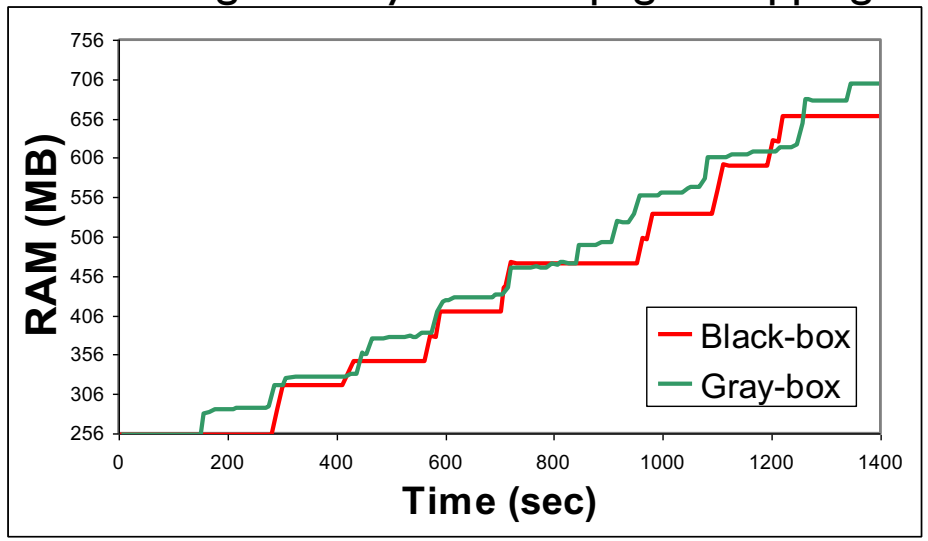
Migration Effectiveness

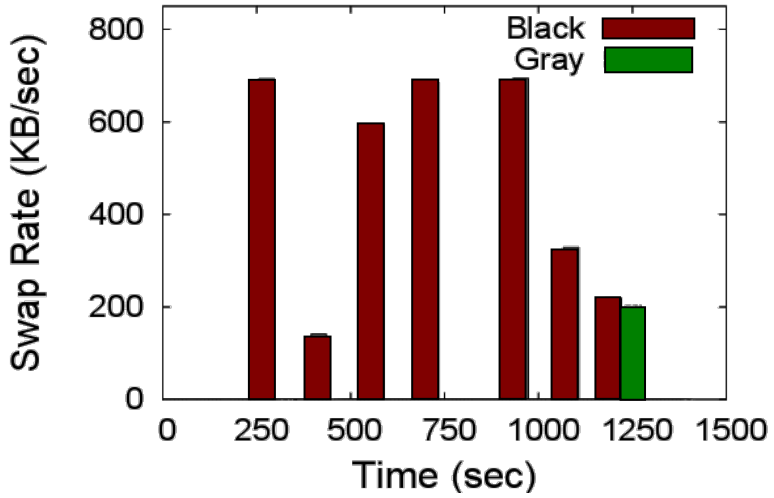
- 3 Physical servers, 5 virtual machines
  - VMs serve CPU intensive PHP scripts
- Migration triggered when CPU usage exceeds 75%
- Sandpiper detects and responds to 3 hotspots



Memory Hotspots

- Virtual machine runs SpecJBB benchmark
  - Memory utilization increases over time
- Black-box increases by 32MB if page-swapping observed
- Gray-box maintains 32 MB free
  - Significantly reduces page-swapping





Gray-box can improve application performance by proactively increasing allocation