COMP5349 – Cloud Computing

Week 2: Data Center and Virtualization Technology

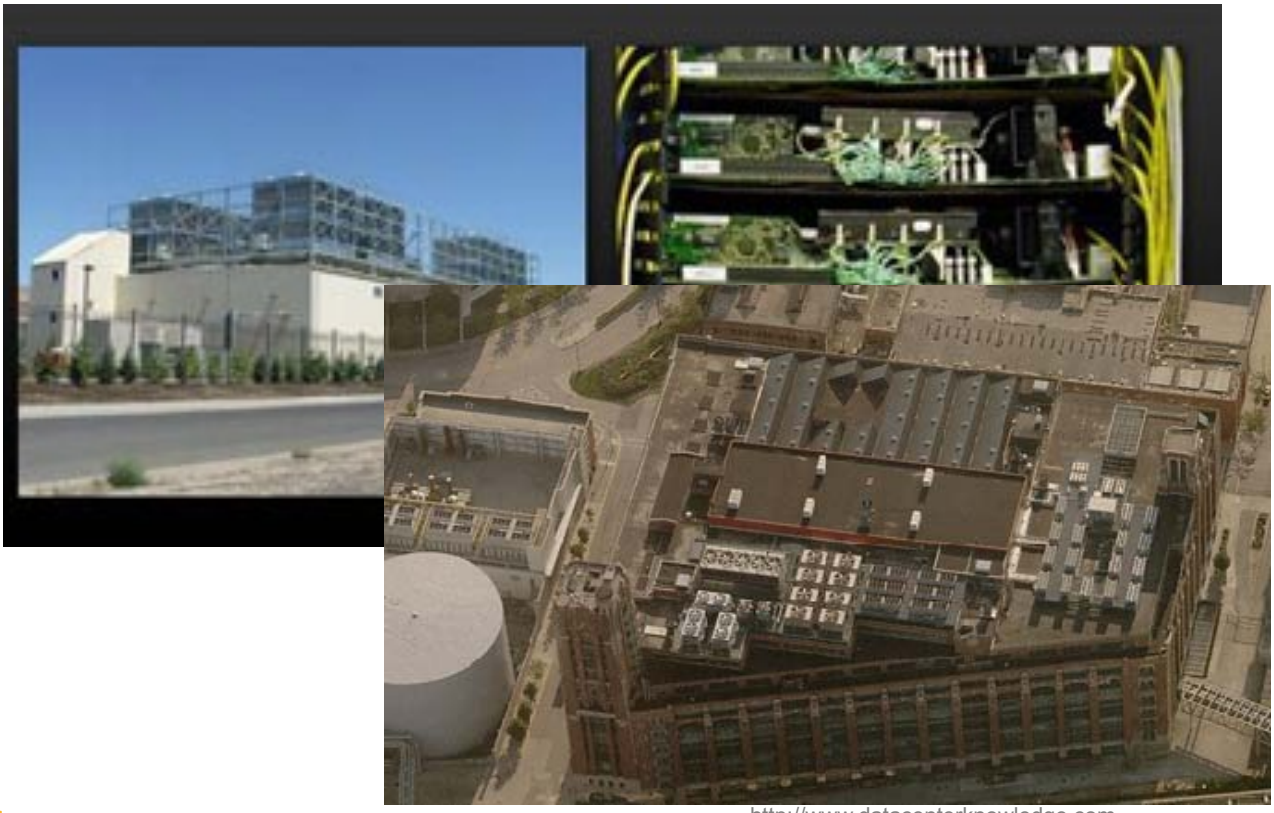
A/Prof Uwe Röhm
School of Information Technologies

This Week's Agenda

- **Data Centers: Infrastructure of Scale**
- **Virtualization Technology**
- **Multitenancy (Overview)**

Based on slides from James Hamilton (perspectives.mvdirona.com)
and from Jaehyuk Huh (KAIST).

Cloud Data Centers



http://www.datacenterknowledge.com

Data Center Building Blocks

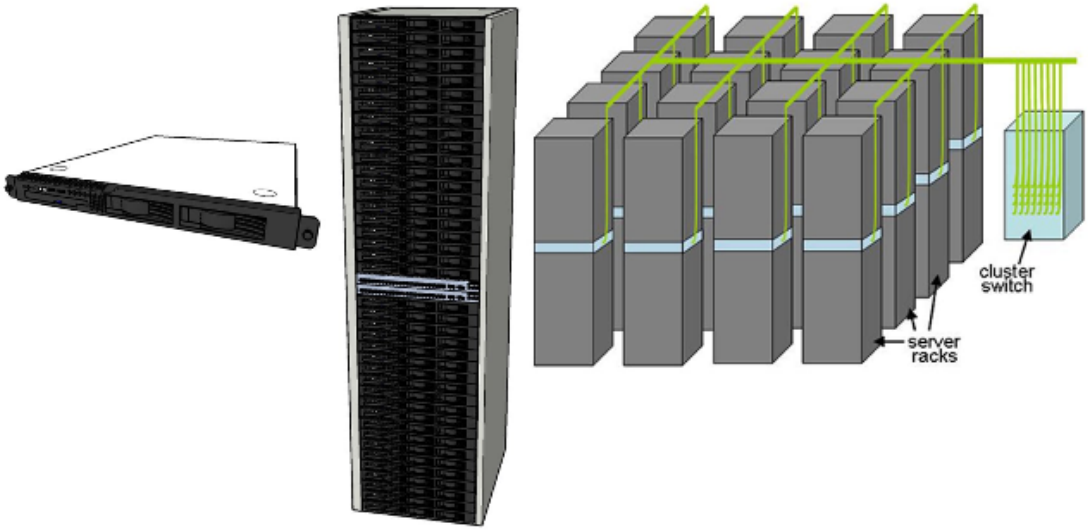


FIGURE 1.1: Typical elements in warehouse-scale systems: 1U server (left), 7′ rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/router (right).

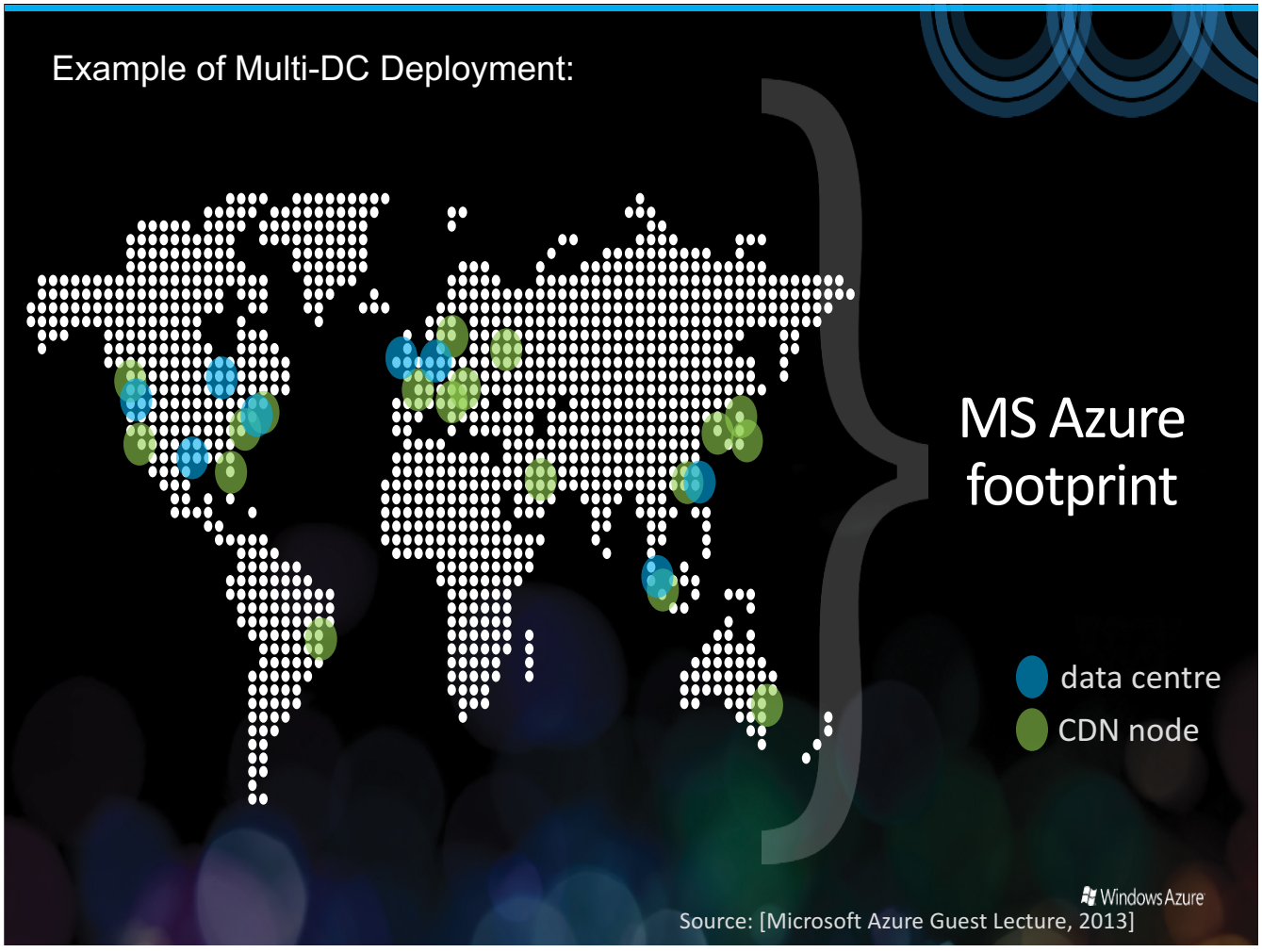
Infrastructure of Scale

- Typical ‘cloud-size’ data center will have about 40-50k servers
  - ‘commodity’ design with multiple cores
  - virtualisation to host multiple services on same shared hardware

  => Plunging cost of computing

- Multiple datacenters
  - At scale, multiple datacenters can be used
    - Close to customer
    - Cross data center data redundancy
    - Address international markets efficiently

- Remote administration is crucial
  - Machines and centers fully monitored; administered over network not only OS level, also network and infrastructure (e.g. even IP-PDU)



Performance Is Everything?

- Traditionally: Computer Systems optimized for Performance (e.g. throughput)
- Nowadays, power is an increasingly important measure too
  - Costs of data center (DC) are dominated by **power** and **cooling** infrastructure
  - Carbon trading schemes will effect this even more
  - Hence DCs to optimize for *work done*/*Watt*

How is DC Energy Efficiency Measured?

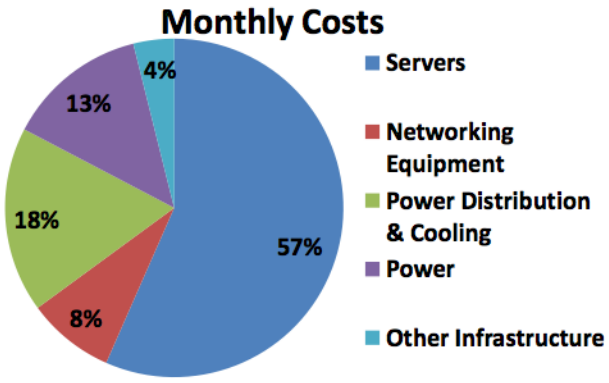
- Commonly used metric for assessing data centers (DCs):
- **Power Usage Efficiency (PUE)**

$$\text{PUE} = \frac{\text{Total Facility Power}}{\text{IT Equipment Power}}$$

- The average data center in the US has a PUE of 2.0 (Source: EPA - U.S. Environmental Protection Agency)
- According to James Hamilton, many have even PUE > 3.0
- High scale cloud services in the 1.2 to 1.5 range

Data Center Costs

- **Assumptions:**
  - Facility: ~$88M for 8MW critical power
  - Servers: 46,000 @ $1.45k each
  - Commercial Power: ~$0.07/kWhr
  - Power Usage Effectiveness: 1.45

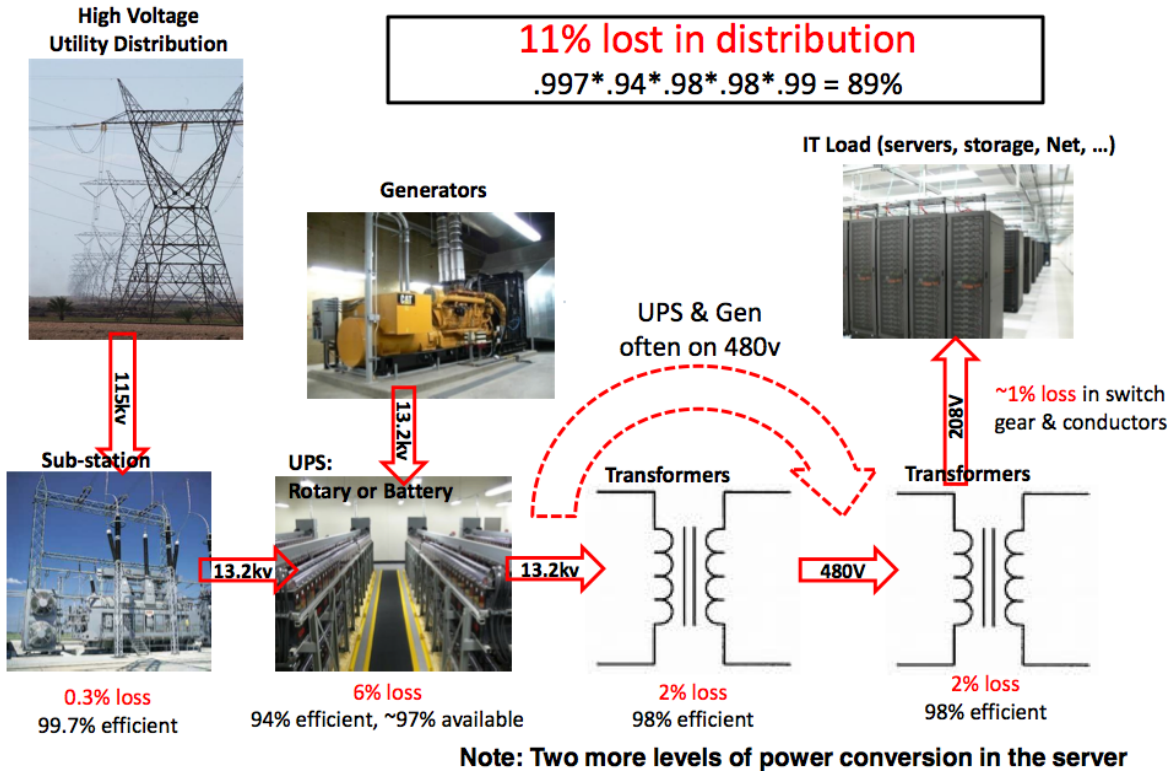


3yr server & 10 yr infrastructure amortization

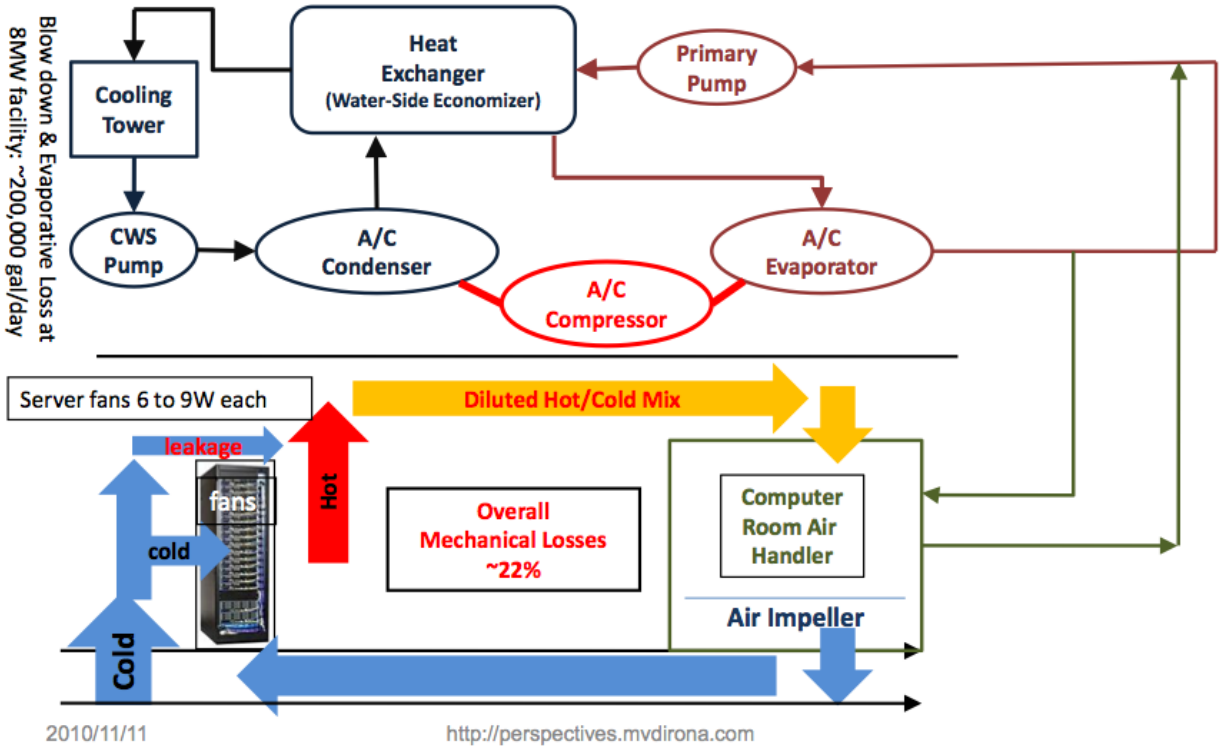
- **Observations:**
  - 31% costs functionally related to power (trending up while server costs down)
  - Networking high at 8% of costs & 19% of total server cost (many pay more)

From: http://perspectives.mvdirona.com/2010/09/18/OverallDataCenterCosts.aspx
http://perspectives.mvdirona.com

Power Distribution



Mechanical Systems



Hot Aisle / Cold Aisle Containment

WriteLine

Intel

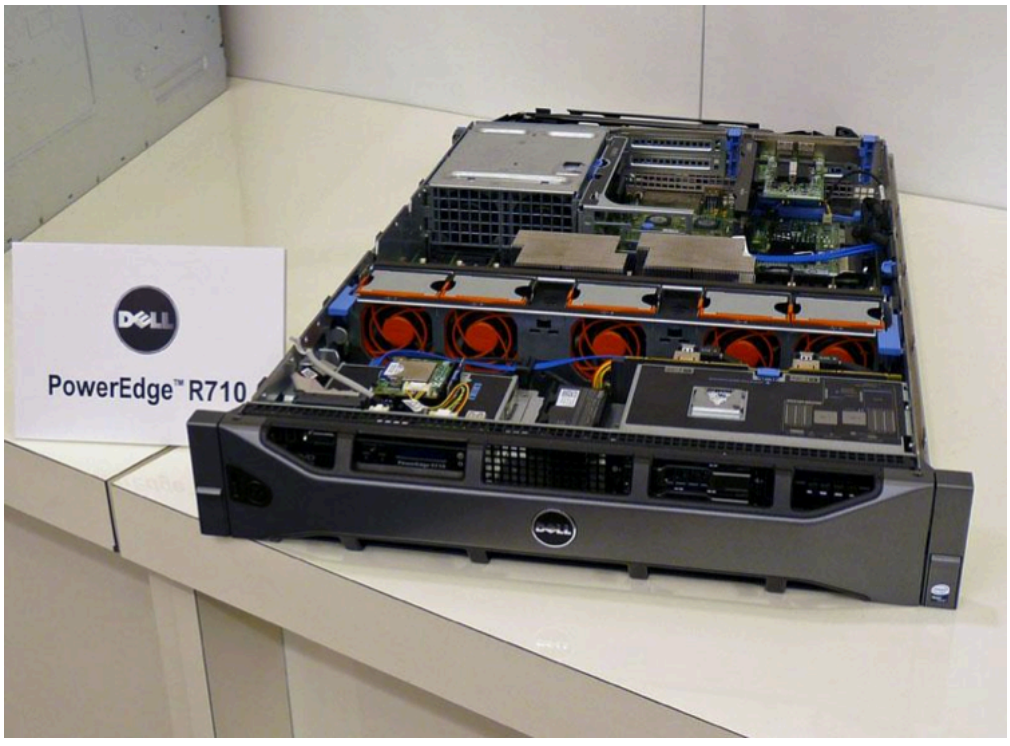
Intel

2010/11/11 http://perspectives.mvdirona.com

Data Center Commissioning

Improvised DC commissioning at Rhyad data center

Commodity Rack Servers



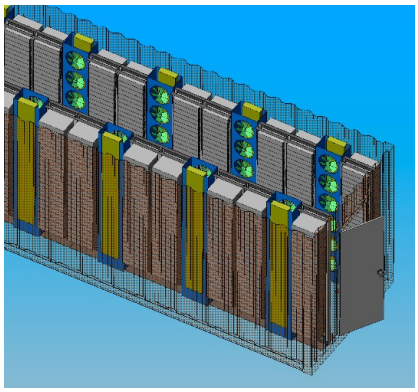
Copyright cloud.watch.impress.co.jp

How much 'Commodity' is possible?

Nortel Steel Enclosure
Containerized telecom equipment

Sun Black Box (242 systems in 20')

Rackable Systems (1,152 Systems in 40')



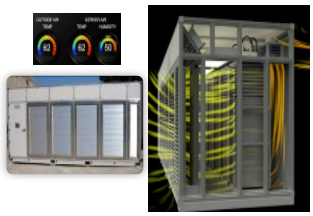
Rackable Systems Container Cooling Model

Source: [CIDR2009 keynote talk by J.Hamilton]

Unit of Data Center Growth

- One at a time:
  - 1 system
  - Racking & networking: 14 hrs ($1,330)
- Rack at a time:
  - ~40 systems
  - Install & networking: .75 hrs ($60)
- Container at a time:
  - ~1,000 systems
  - No packaging to remove
  - No floor space required
  - Power, network, & cooling only
- Weatherproof & easy to transport
- Data center construction takes 24+ months
  - Both new build & DC expansion require regulatory approval

Source: [http://perspectives.mvdirona.com]



IT PAC
(Pre-Assembled Components)

Server
Capacity

Rack

Containers

Challenges

- Consumer Services
  - Google estimated at ½ million systems in 30 data centers
- Basic observation:
  - No single system can reliably reach five 9's
    (need redundant H/W with resultant S/W complexity)
  - With S/W redundancy, most economic H/W solution is large numbers of commodity systems

Source: [http://perspectives.mvdirona.com]

Data Center in a Container

This Week's Agenda

- Data Centers & Computing Units
- **Virtualization Technology**
- Multitenancy (Overview)

Based on slides from Jaehyuk Huh (KAIST),
and the SOSP'03 paper by P. Braham et al.: *Xen and the Art of Virtualization*.

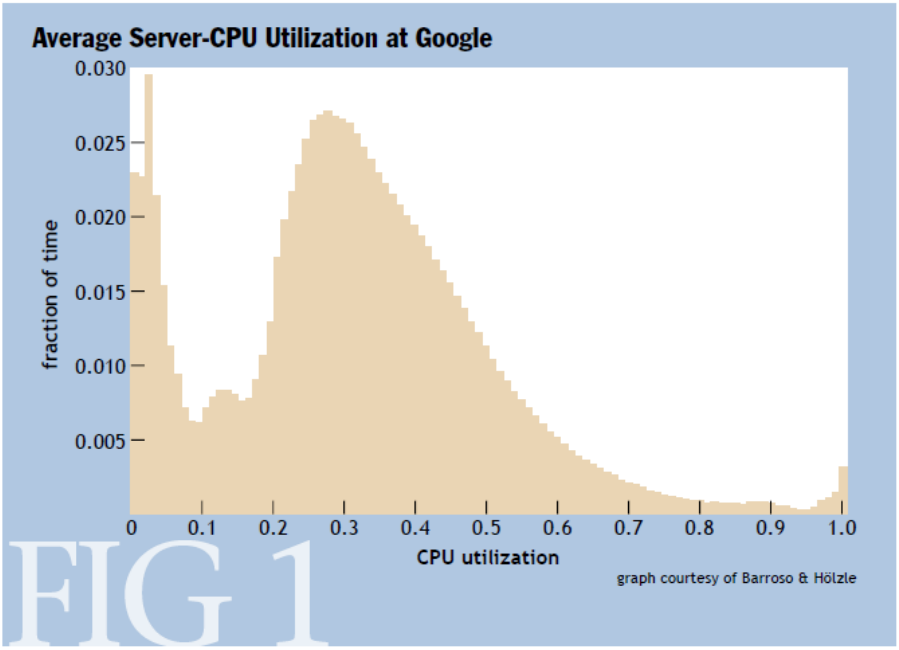
Well-Known Utilization Problem: Server Sprawl

- *Server Sprawl*
  (large number of underutilized servers)
  a major problem in many IT departments
- Main causes:
  - Requirement by vendors to run their applications in isolation
  - Operating system heterogeneity
    - Mail server may require Windows server; a database may best run on Linux or Solaris.
  - Mergers and other integration projects
- "81 percent of CIOs were using virtualization technologies to drive consolidation, according to a recent survey by CIO" (2008 survey)

Increasing Utilization is Hard!

The problem only magnifies on a data center scale:

Average CPU utilization of 5000+ servers at Google during a six- month period



Beyond Server Consolidation. Werner Vogels. ACM Queue, Jan/Feb 2008.

100% Utilization is not the Goal

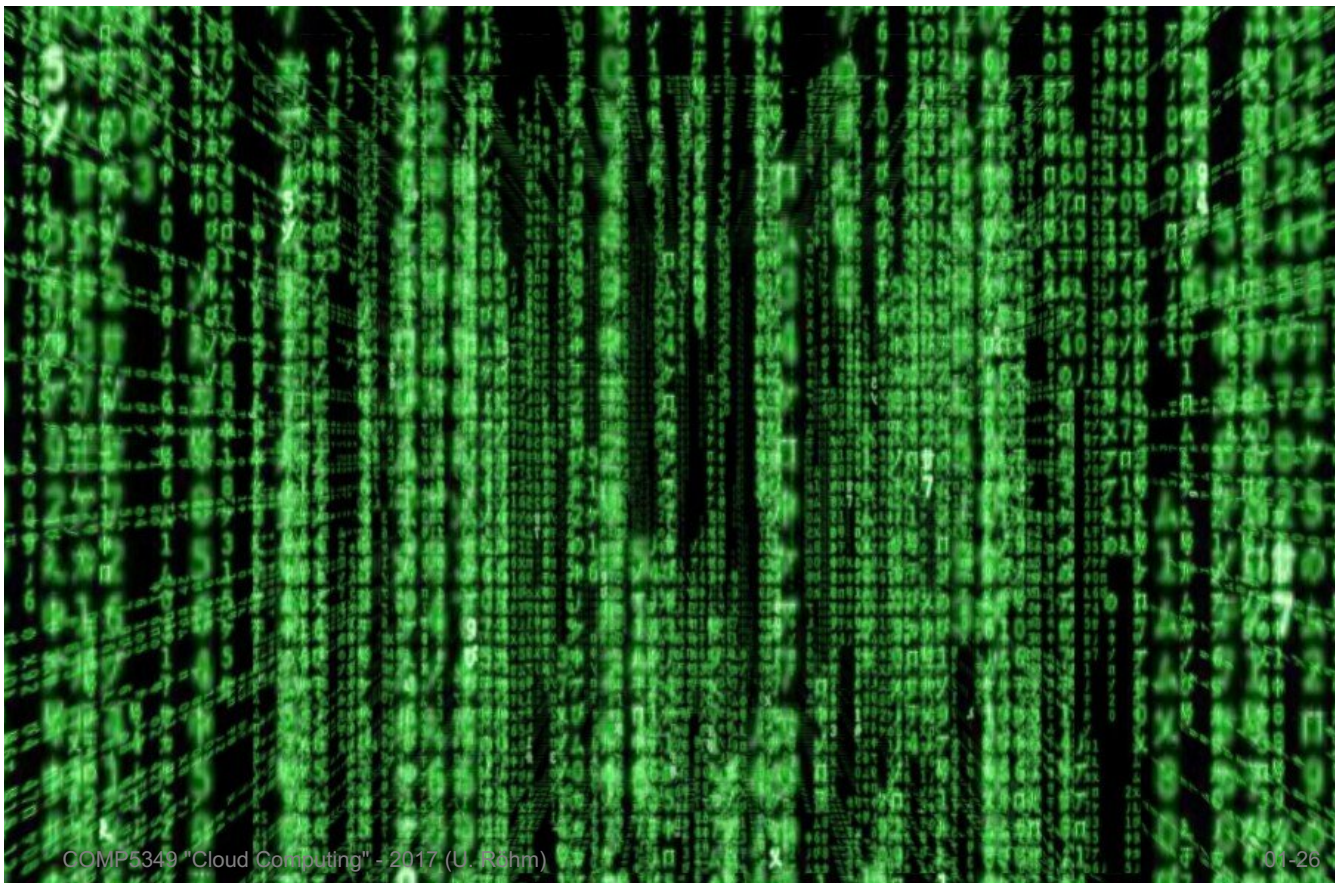
- Workload in the enterprise are heterogeneous
- Demand is uncertain and often occurs in spikes
- Operating System starts to behave unpredictable under high CPU and IO load
- for pure CPU-bound environments, 70 percent seems to be achievable for highly tuned applications; for environments with mixed workloads, 40 percent is a major success, and 50 percent has become the Holy Grail.
- Real world estimates of server utilization in datacenters range from 5% to 20%!
  - 30% is seen as a very good value already…

Server Consolidation

- Solution: Pool heterogeneous services together
  - To do so, run several **virtual machines** on the same shared hardware, coordinated by a **hypervisor**
    - (also known as: virtual machine manager/monitor (VMM))
    - Hypervisor abstracts / hides physical computing platform
  - Allows to share commodity hardware without sacrificing security and performance

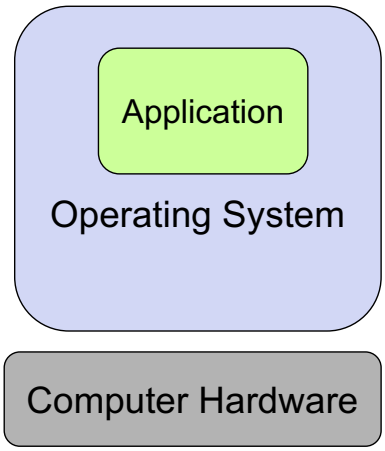
- Benefits:
  - Server consolidation which gives better resource utilization
  - Fault tolerance (able to migrate VMs on faults)
  - Portability (can use any OS on same HW)
  - Manageability

The Idea: Build the Matrix!

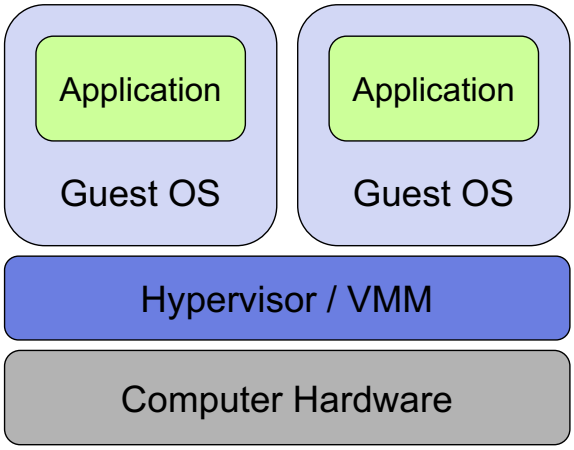


Virtualization Concept

Before Virtualization

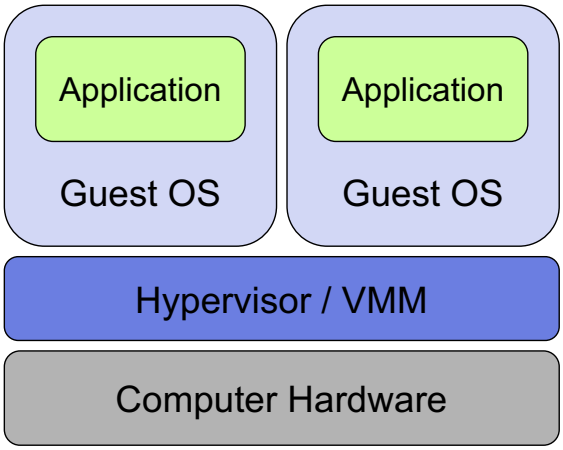


With Virtualization

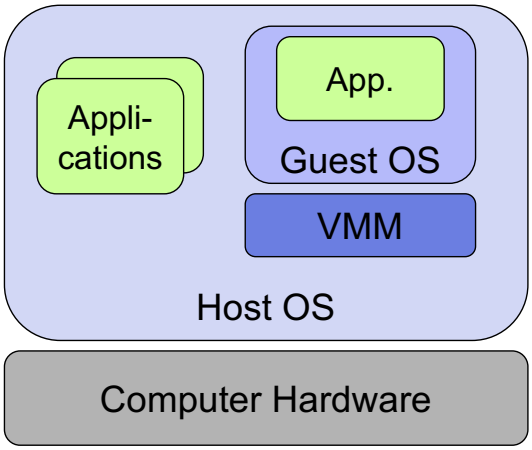


Virtualization Approaches

Type I: Bare-metal VM



Type II: Hosted VM



Challenges for Virtualization

- Fidelity: Software on the VMM executes identically to its execution on hardware, barring timing effects
  - Supporting multiple operating systems
  - Support for unmodified applications
- Safety: The VMM manages all hardware resources
  - Some hardware is 'uncooperative'
    - Missing hardware features
    - Especially true for the original(!) pre-dominant x86 architecture
- Performance
  - an overwhelming majority of guest instructions to be executed by the hardware without the intervention of the VMM

Virtualization Techniques

- **Full Virtualization**
  - No Guest OS modification
  - Either Software Approach
    - Binary translation (on-the-fly modification of code)
    - Examples: VMware, Parallels
  - Or Hardware-assisted Virtualization
    - New privilege mode
    - Examples: VT-x, VT-i, VT-d, AMD-V
- **Paravirtualization**
  - Guest OS modification
  - Software approach
    - Hypercalls
    - Example: Xen

Paravirtualization

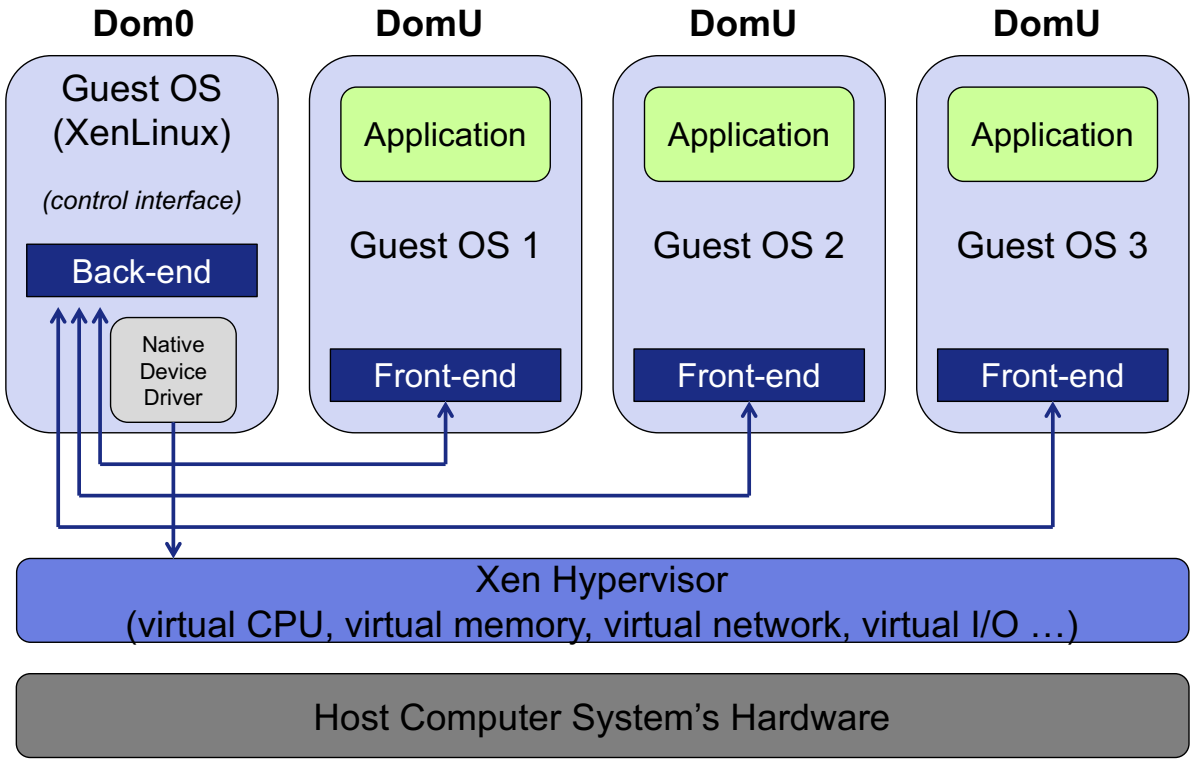
- ***Paravirtualization*** refers to modifying the OS to make virtualization faster
- Can be combined with full virtualization techniques
  - paravirtualize where you can
  - use full-virtualization techniques where you can't avoid it.
- XenoLinux - a port of Linux to run under the Xen hypervisor.
- The bulk of the work is replacing privileged instructions (e.g. cli, hlt, write to cr3) with ***hypercalls***.
- Core concept:
  modify the OS to the virtualized environment, but expose some details of the hardware for optimization.

Xen Hypervisor

- Original published in SOSP 2003:
  Barham et al: "Xen and the Art of Virtualization"

- Open-source hypervisor using Paravirtualization
- Machine gets sub-divided into 'domains'
  - Complete running virtual environments
  - Dom0
    - elevated privilege
    - device drivers
    - creates and manages user domains; User interface
  - DumU
    - unprivileged domains
    - applications; different guest OSs

Xen Architecture



Barham et al: "Xen and the Art of Virtualization"

Virtualizing an Operating System

- CPU Management / Instructions
- Memory Management
- Device I/O

General VM Techniques

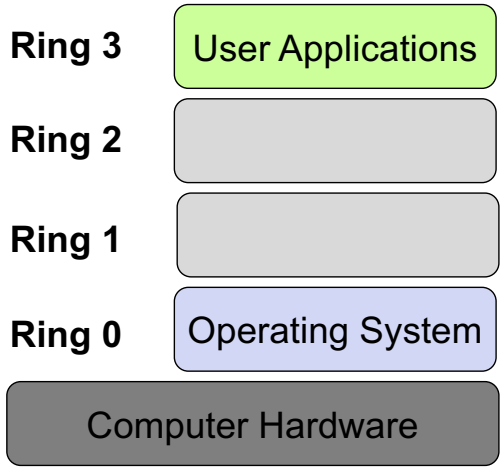
- De-privileging
  - Executing guest operations systems at a reduced privilege level
  - Trap and emulate the privileged instruction against the virtual machine state
- Primary and shadow structures
  - Privileged state of a virtual system differs from that of the underlying system
  - VMM maintains an image of the guest CPU states
  - Shadow page table

Xen Virtualization

- Hypervisor needs control on how much CPU cycles are allocated to which guest OS
  - And needs control on actual hardware
- BUT: OS's designed to run with highest privileges on CPU
  - De-Privileging
    - Guest OS must run at lower privilege.
      Since ring 1-2 seldom used, run guest OS in ring 1.
  - Exceptions
    - Guest OSs must register handlers with Xen. Generally identical to original.
    - Safety is done by making sure it doesn't execute in ring 0.
  - System Calls
    - "Fast" handlers may be registered to avoid going through ring 0.
      Instead go from ring 3 to ring 1.
  - Page Fault
    - Page fault handler must be modified, fault addr in a priv reg.
    - Technique is for Xen to write to a location in the stack frame.

X86 CPU Virtualization

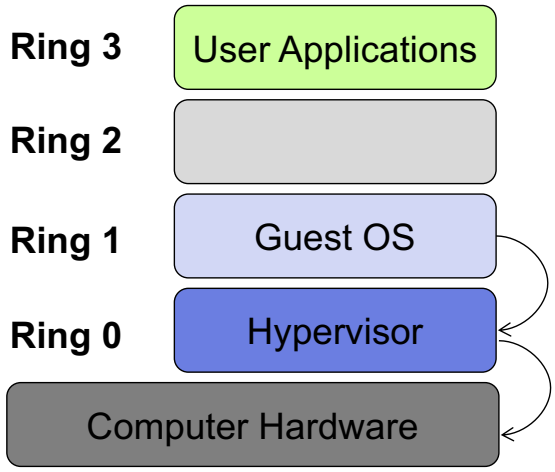
- On x86, HW access organised in 'rings' (levels of privileges):
  - Ring 0: Operating system
  - Ring1-3: applications, where apps typically run in ring 3
  - Only Ring 0 can access certain privileged CPU commands



- Where sits the hypervisor?

The Approach taken by Xen

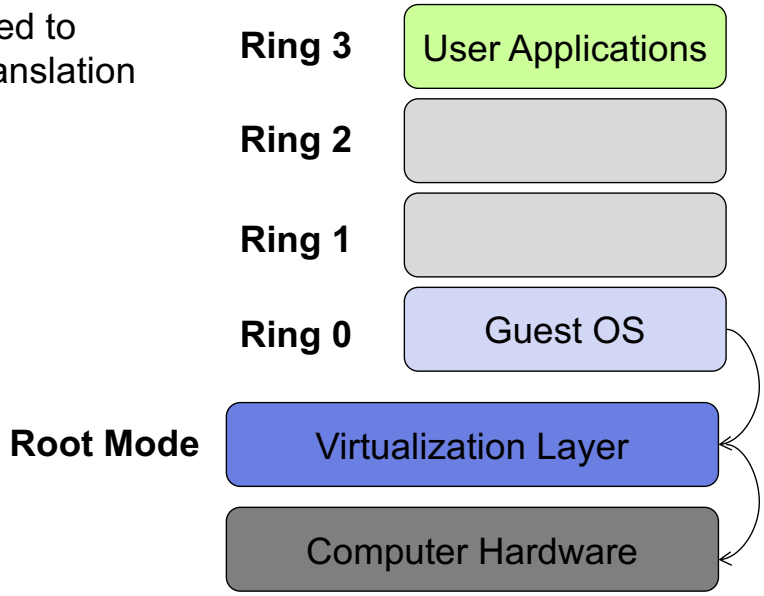
- Privileged instructions are *paravirtualized*
  - Xen runs in ring 0, where the kernel normally runs.
  - The kernel is moved to ring 1 or 2. This requires changes to the operating system changes.
  - User apps. runs in ring 3 - just like it does today – no userspace changes necessary.



  - Guest OS is modified to call through Xen ('*hypercalls*')
  - Xen validates first whether allowed
  - Then executes on behalf of the guest OS

Hardware-Assisted Virtualization

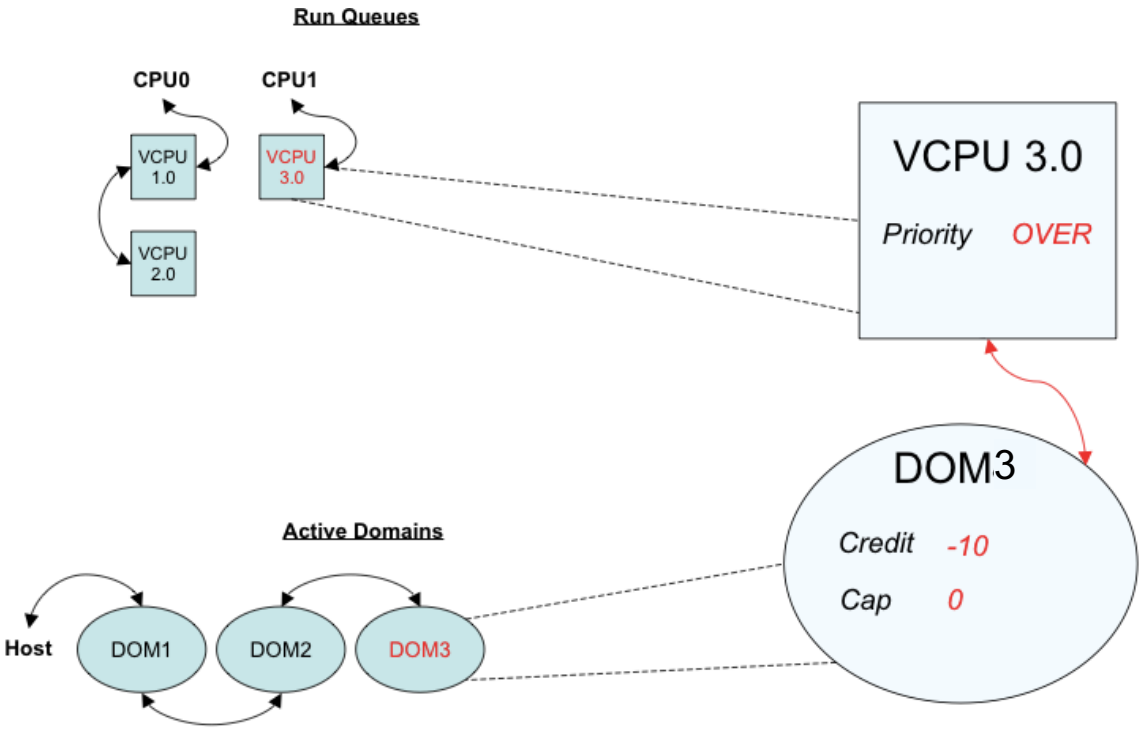
- Introduction of new privilege level directly in CPU
  - OS requests are trapped to VMM without binary translation or paravirtualization



Xen: Scheduling

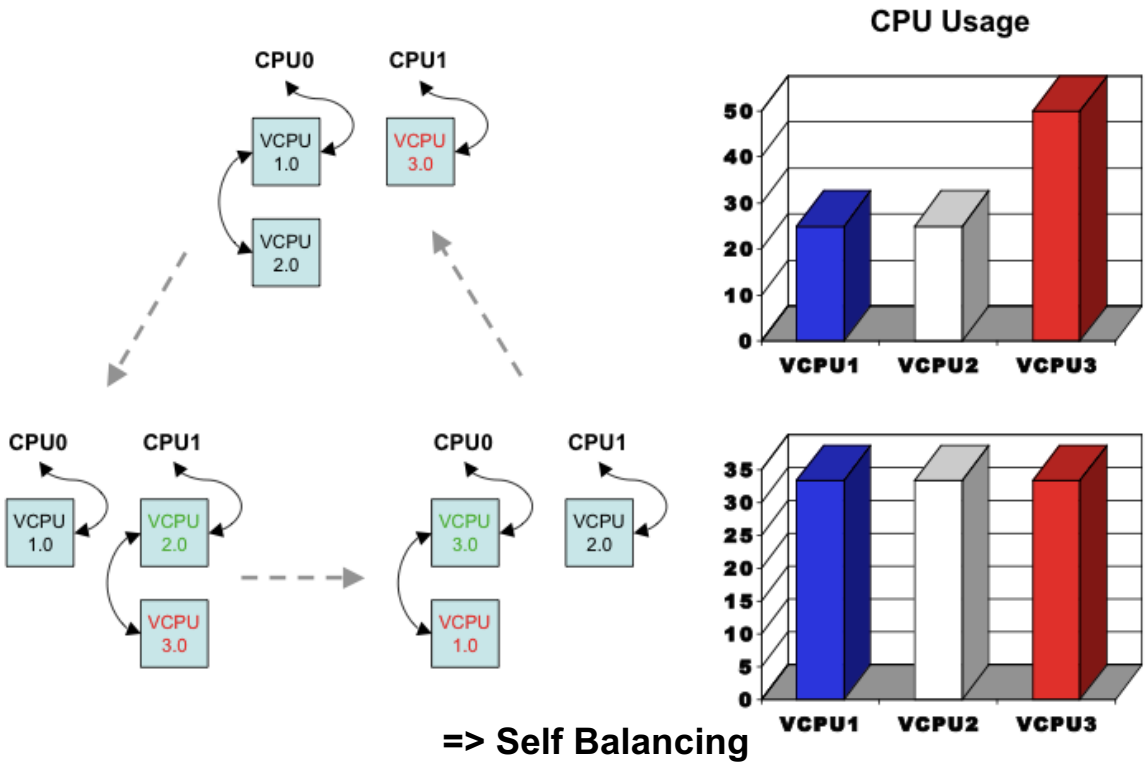
- Goal: Simple, fast and **fair** scheduler
- Credit Scheduler
  - virtual CPU (VCPU) assign to each VM (unit of scheduling)
  - VCPU's priority: two possible values
    - ***under*** (below fair share usage) or ***over*** (*above* fair share usage)
  - Every 30msec, credit is assigned to all VMs
  - As a VCPU runs, it consumes credits
    - Negative credits imply a priority is *over*
    - Until a VCPU consumes its allotted credits, its priority is *under*
  - Scheduling decision
    - End of time slice or VCPU blocks
    - Common case: run next local VCPU under fair share
    - Otherwise: pick queued remote (if more than one core) under fair share VCPU, local over VCPU, or remote over VCPU (in that order)

Credit Scheduler Example



[E. Ackaouy: "The Xen Credit CPU Scheduler"]

Credit Scheduler Example (cont'd)



=> Self Balancing

[E. Ackaouy: "The Xen Credit CPU Scheduler"]

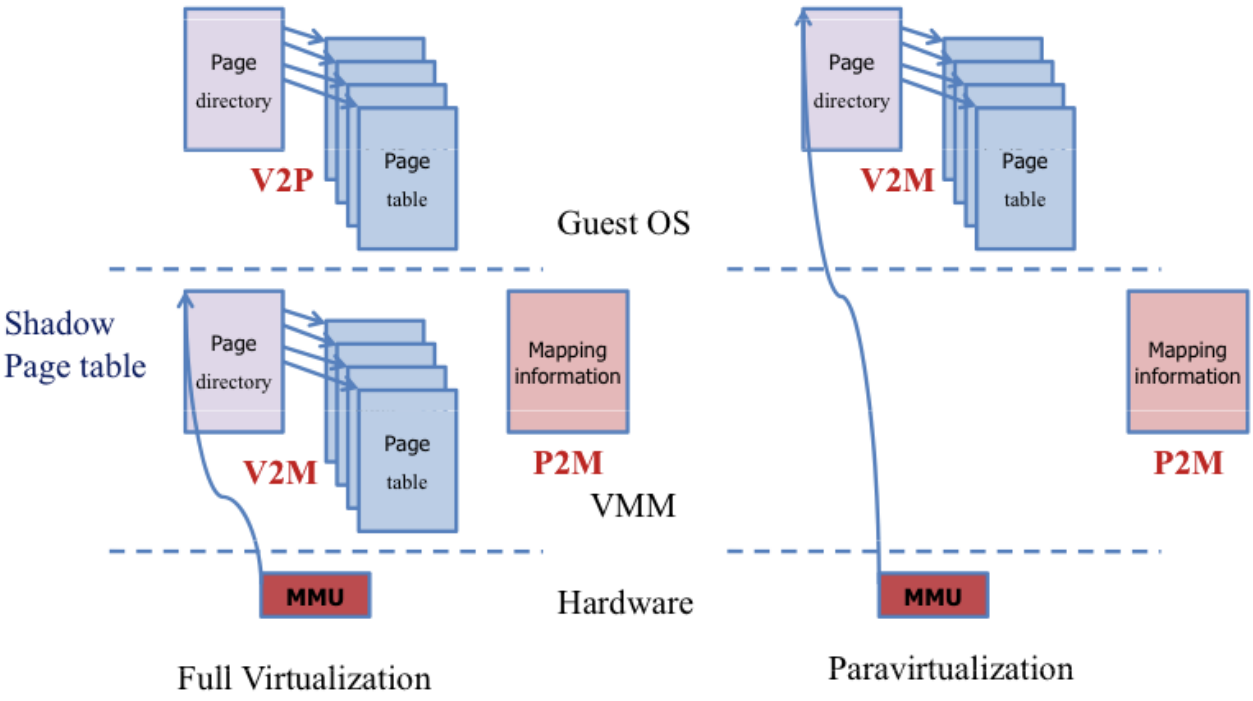
Virtualizing Memory Management

- Memory Management
  - The virtual memory management of an OS now has to cooperate with the virtual memories of other guest OS
  - Challenge 1: Protecting the Hyper-visor
  - Challenge 2: Performance
  - Challenge 3: VMM commands are privileged instructions

Virtualizing the MMU – 2 Approaches

- Approach 1: Shadow page table
  - Guest OS keeps its own set of page tables, distinct from the set of page tables that are shared with the hardware.
  - the hypervisor traps page table updates, validates them and propagates changes to the hardware page tables and back.
- Approach 2: Direct Page Tables access
  - the OS is allowed read only access to the real page tables.
  - Page tables updates must still go through the hypervisor rather than as direct memory writes.
  - guest OSes allocate and manage their own PTs - use a hypercall to change the PT base
  - updates go through the hypervisor, which validates them - the OS must not give itself unrestricted PT access, access to hypervisor space, or access to other VMs.

Shadow Mode vs. Direct Mode



Approach taken by Xen

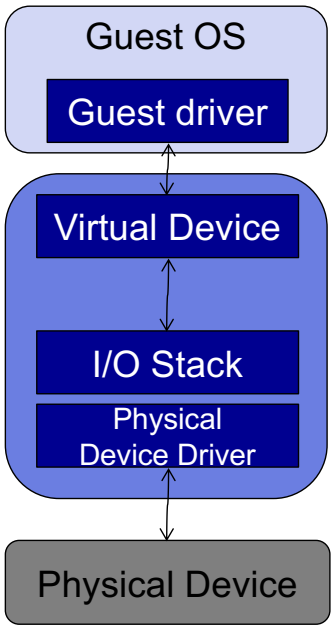
- Physical memory
  - Reserved for each guest OS instance at time of creation.
  - Provides strong isolation
  - But no sharing.
  - top 64MB of either address space is reserved for hypervisor
    - => no swapping of TLB needed when entering hypervisor -> fast
- Virtual address translation in Direct Mode
  - Handled by Xen, changes must be validated by Xen
  - For performance: batched updates possible

I/O Virtualization

- Each Guest OS needs to read/write disks, the network etc.
  - But this is now shared access across all OS'
  - Presence of hypervisor is another layer, so imperative to minimize overhead
    - Hypervisor could emulate the HW, but this is slow…
- Xen exposes abstractions which the guest OS' have to use
  - Network, Disk, etc. replaced with special, buffer-based event mechanism.
  - Lightweight event infrastructure to demultiplex data
  - Memory committed to I/O comes from relevant domains
- Goals:
  - resource accountability

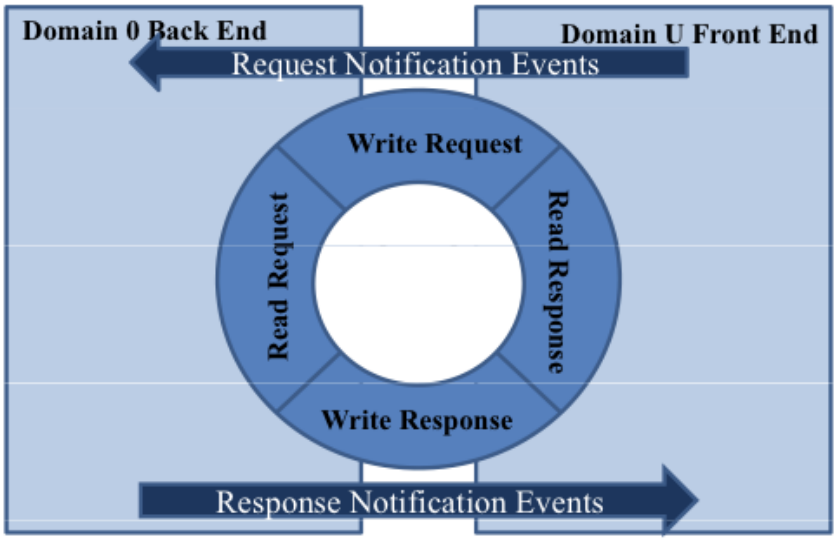
I/O Virtualization Architecture

- Guest driver
- Virtual device
- Communication mechanism
  - Virtual device -> Virtualization stack
- Virtualization I/O stack
- Physical device driver
- Real physical device



I/O Virtualization in Xen

- Split Driver Model
  - Front-End in DomU, Back-End in Dom0
- Shared memory Ring Buffers
  - Asynchronous communication



Porting Costs (Org paper from 2003)

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).

Barham et al: "Xen and the Art of Virtualization"

Evaluation (2003): Overhead of Xen?

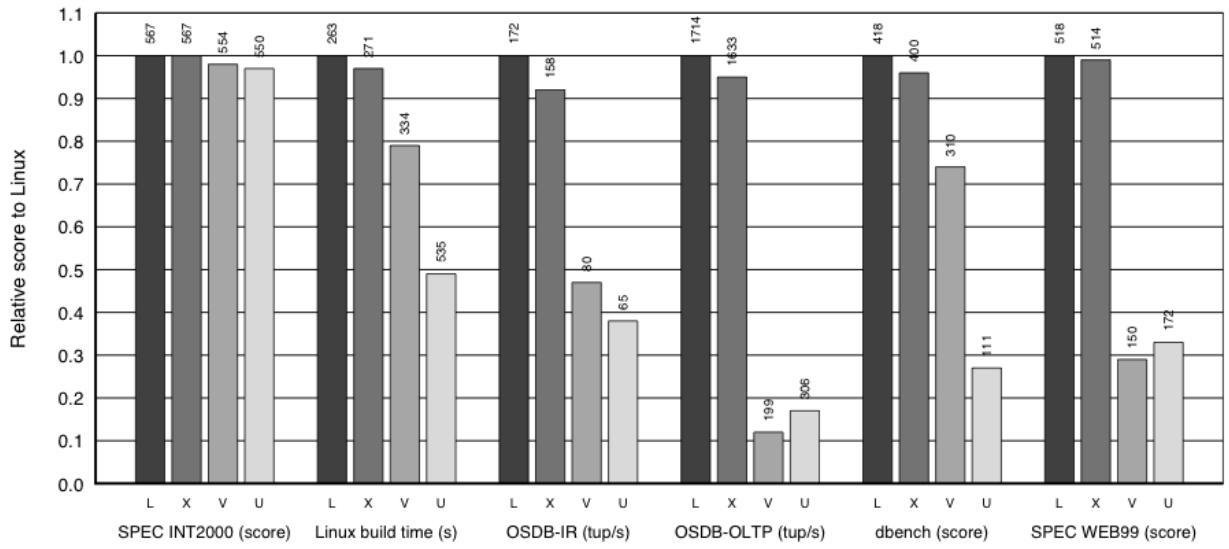


Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

Barham et al: “Xen and the Art of Virtualization”

Evaluation (2003): Scalability of Xen

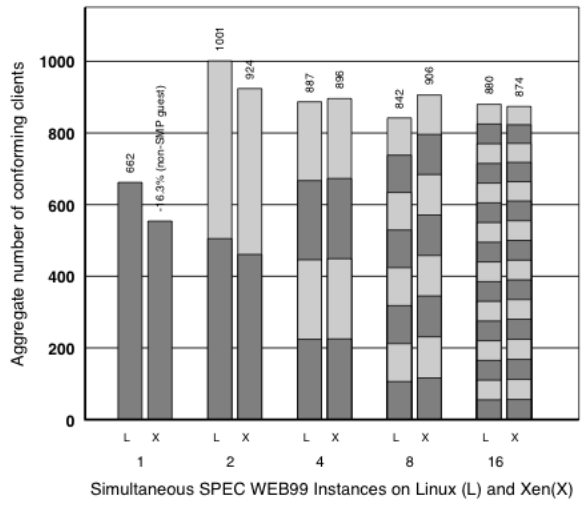


Figure 4: SPEC WEB99 for 1, 2, 4, 8 and 16 concurrent Apache servers: higher values are better.

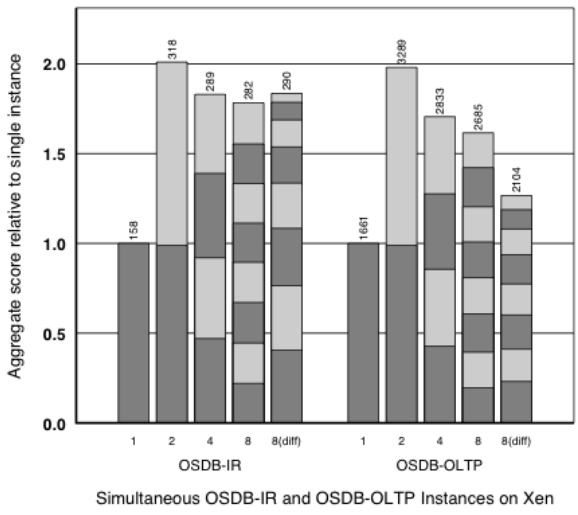


Figure 5: Performance of multiple instances of PostgreSQL running OSDB in separate Xen domains. 8(diff) bars show performance variation with different scheduler weights.

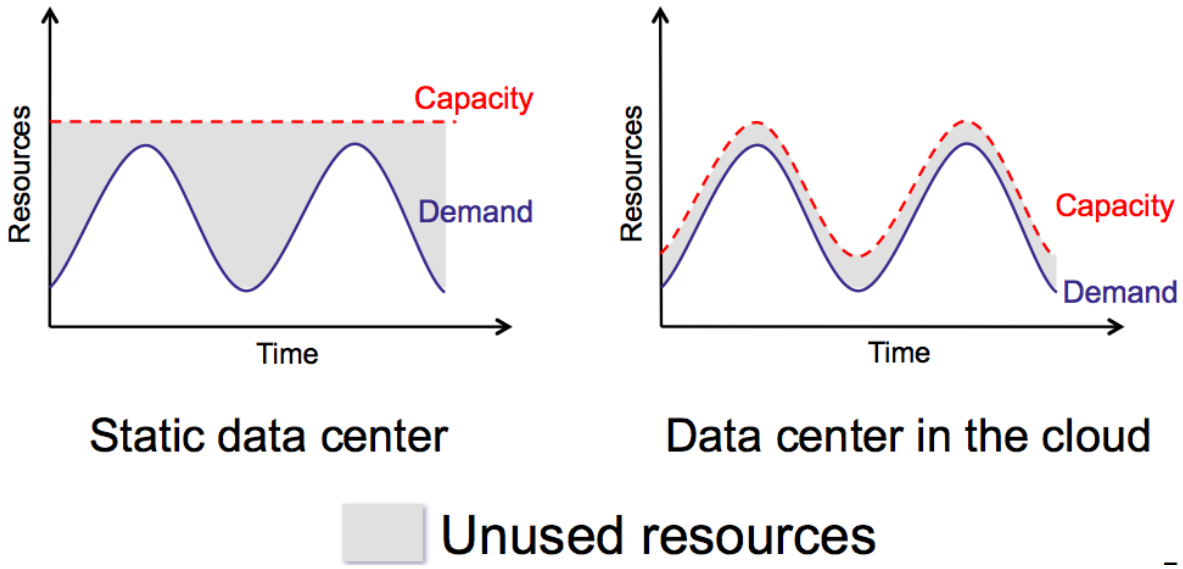
Barham et al: “Xen and the Art of Virtualization”

Evaluation (2003): Scheduling Isolation

- Isolation Test:
  - 4 Domains:
    1. PostgreSQL with OSDB-IR
    2. SPEC-Web 99
    3. disk-I/O hog (sustained dd)
    4. 'fork bomb' + VM intensive application
  - Both database and web domains were still responsive and just about 2-4% slower than measured before
  - On native Linux with same 4 processes in one OS -> machine hangs
- Scalability Test:
  - Started 128 domains on the same machine, each running a CPU intensive task (subset of SPEC CINT 2000)
  - 7.5% slower than Linux – which can be improved by opt. time slice
  - Machine still responsive when starting a 129th domain

What does this enable us to do?

- Pay by use instead of provisioning for peak



This Week's Agenda

- Data Centers & Computing Units
- Virtualization Technology
- **Multitenancy**

Based on slides from Jaehyuk Huh (KAIST),
and the SOSP'03 paper by P. Braham et al.: *Xen and the Art of Virtualization*.

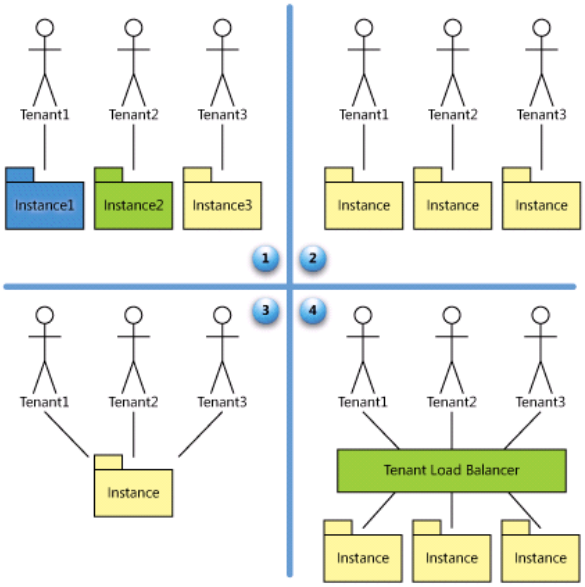
Multitenancy

- Virtualization enables efficient sharing of hardware resources
  - Allows to run separate instances of the software stack, starting from the operating system, to run isolated on the same server
- What about the application software?
  - For SaaS, the 'shared resource' is the application software!

=> SaaS (and some PaaS) are following the multitenancy principle.

- **Multitenancy**
  - single software instance serves multiple client organizations (*tenants*)
  - Software application: "designed to virtually partition its data and configuration, and each client organization works with a customized virtual application instance." [Wikipedia]
  - metadata based configuration instead of code based customization
    - note the difference between multitenancy and multi-users…

The Software as a Service Maturity Model



SaaS Maturity Levels

(1) Level I: Ad Hoc/Custom

- The first level of maturity is similar to the traditional application service provider (ASP) model of software delivery, dating back to the 1990s. *Customized instance*.

(2) Level II: Configurable

- At the second level of maturity, the vendor hosts a separate instance of the application for each customer (or *tenant*). Same instance, configured to suit different customers

(3) Level III: Configurable, Multitenant-Efficient

- Only has the “scale up” option

(4) Level IV: Scalable, Configurable, Multitenant-Efficient

- Multiple instances; not 1:1 mapping; easy to scale out

Summary

- **Commodity Data Center**
  - minimizing Power Consumption is important
  - Need to increase Resource Utilization
- **Virtualization Technology**
  - Sharing of Hardware Resources for Iaas / PaaS
  - Full Virtualization vs. Paravirtualization
  - CPU, Memory and Device virtualization; isolation of guest OS
  - Xen Hypervisor
- **Multitenancy**
  - Sharing of software applications for SaaS between larger organisations
  - configurable on software level
  - data access isolation

Where do You Want to Compute Today?

References

- James Hamilton's Blog: http://perspectives.mvdirona.com
- James Hamilton: CIDR'09 Keynote, 2009.
- James Hamilton: SIGMOD 2011 Keynote, 2011.
- http://www.datacenterknowledge.com
- Werner Vogels, *Beyond Server Consolidation*. ACM Queue, Jan/Feb 2008.
- *Xen and the Art of Virtualization*. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. In Proceedings of the Nineteenth ACM Symposium on Operating systems principles (SOSP '03), 2003.
- Jaehyuk Huh: CS530 Operating Systems lecture slides, KAIST, 2010.
- Emmanuel Ackaouy: *The Xen Credit CPU Scheduler.* Xen Source 2006.
- Clark, Fraser et al.: *Live Migration of Virtual Machines*. In NSDI 2005.
- Sudipto Das, Shoji Nishimura, Divyakant Agrawal, Amr El Abbadi, "Albatross: Lightweight Elasticity in Shared Storage Databases for the Cloud using Live Data Migration", In the 37th Int'l Conf. on Very Large Data Bases (VLDB) 2011.
- Aaron Elmore, Sudipto Das, D. Agrawal, Amr El Abbadi, "Zephyr: Live Migration in Shared Nothing Databases for Elastic Cloud Platforms", SIGMOD 2011.

Glossary

- ATS – Automated Transfer Switch
  - Automated electric switching gear that can sustain power to a load from either the electric grid or a generator
- IP4PDU – IP4enabled Power Distribution Unit
  - Switched power distribution units (PDUs) that support remote administration, configuration and monitoring
- PSU – Power Supply Unit
  - The power supply device within a server that converts the mains AC input to the internally required DC voltage
- SOL – Serial Over LAN
  - A mechanism that allows HW controllers of some managed system to redirect the input/output of a serial port over IP