EC/CS 528: Cloud Computing

Overview of Virtualization

Instructor: Alan Liu

BU Department of Electrical & Computer Engineering

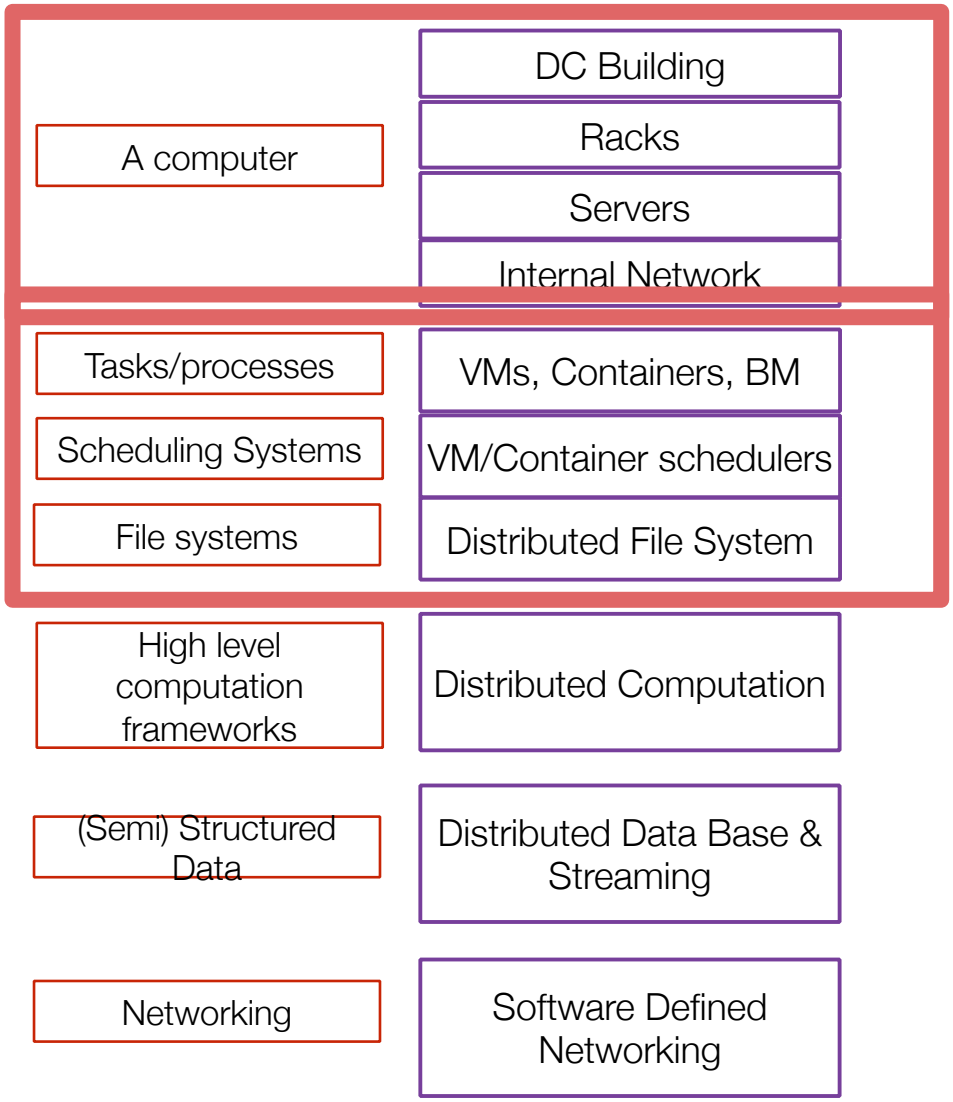
BOSTON UNIVERSITY

Q/A, Projects

- Project description
- Contact the mentor(s) and set up regular communication channels
- We will use GitHub to host open-source projects.
- We will use ZenHub w/GitHub for project management.
  - Two lectures next week. By Michael Daitzman
- We will set up cloud access via NERC
- Questions?

BOSTON UNIVERSITY

Layers

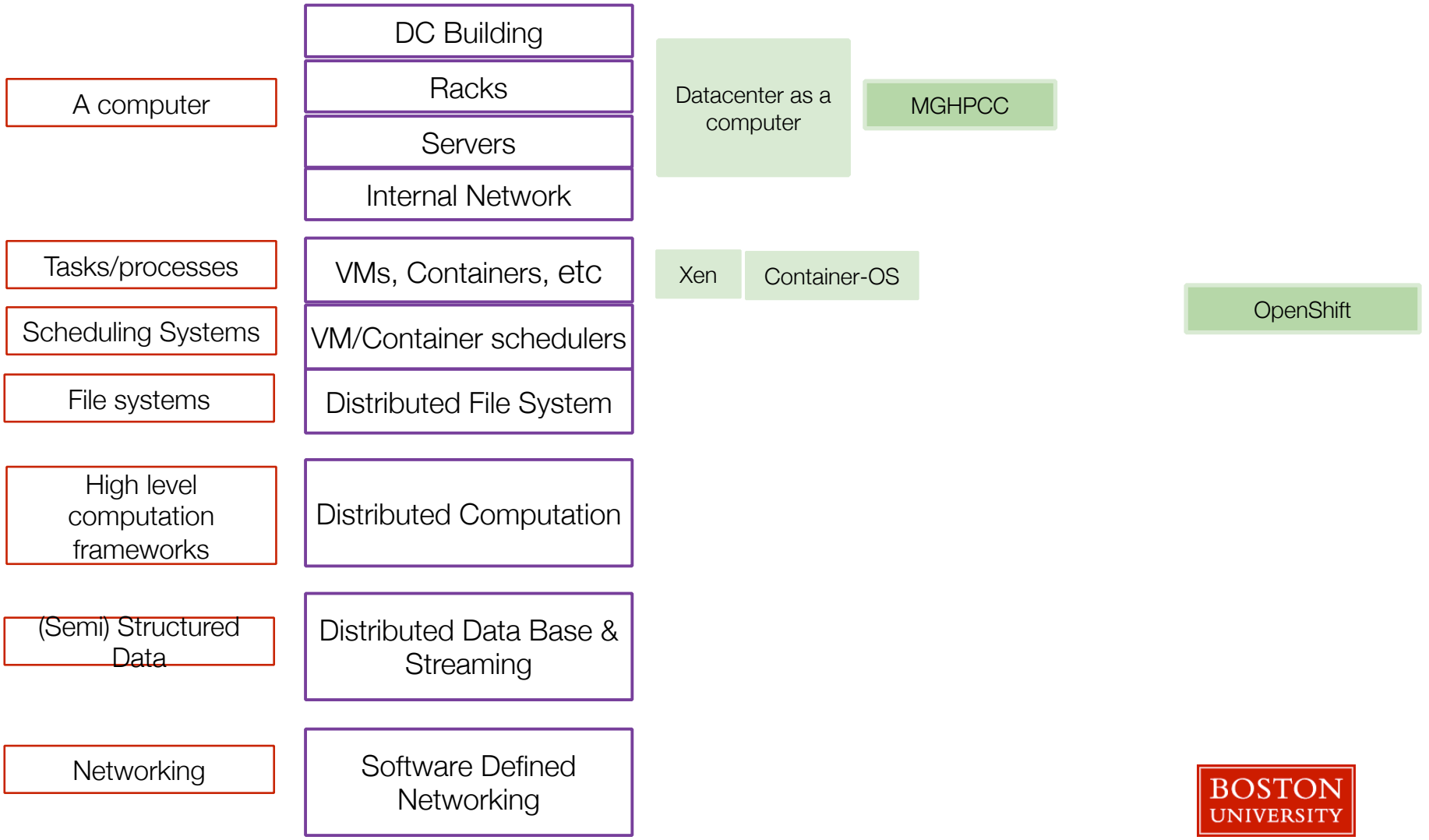


Hardware level:
How do you build cloud-scale systems?

"Operating system":
How do you manage and run cloud applications? What about file systems?

BOSTON UNIVERSITY

Top-down view of the course



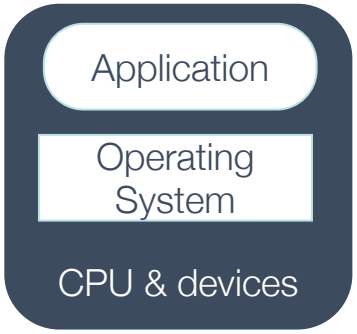
Virtualization

What is it?

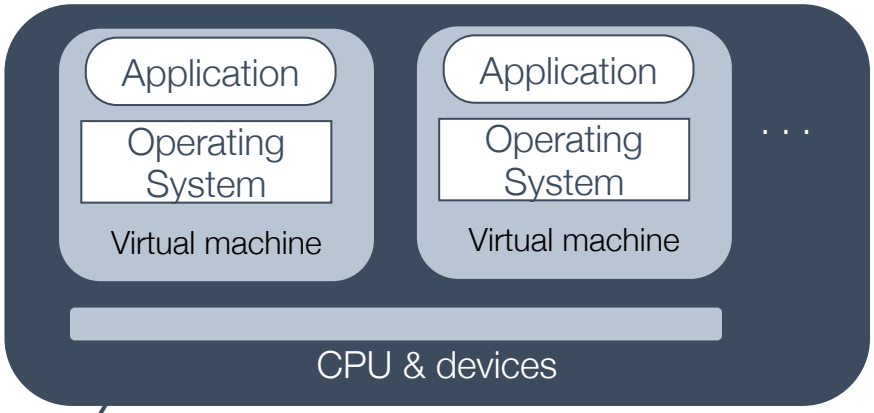
- A software-implemented *virtual machine/computer* that you can run a full OS on top of it
- Started way back in mid 60s, share expensive computer between multiple single-user Oses
- Went away (outside of mainframe) when OSes got better….
- Then… came back in the 90s with VMware

BOSTON UNIVERSITY

Simple view



"Bare Metal"



Multiple "guest" operating systems on virtual machines

"hypervisor" or "virtual machine monitor"

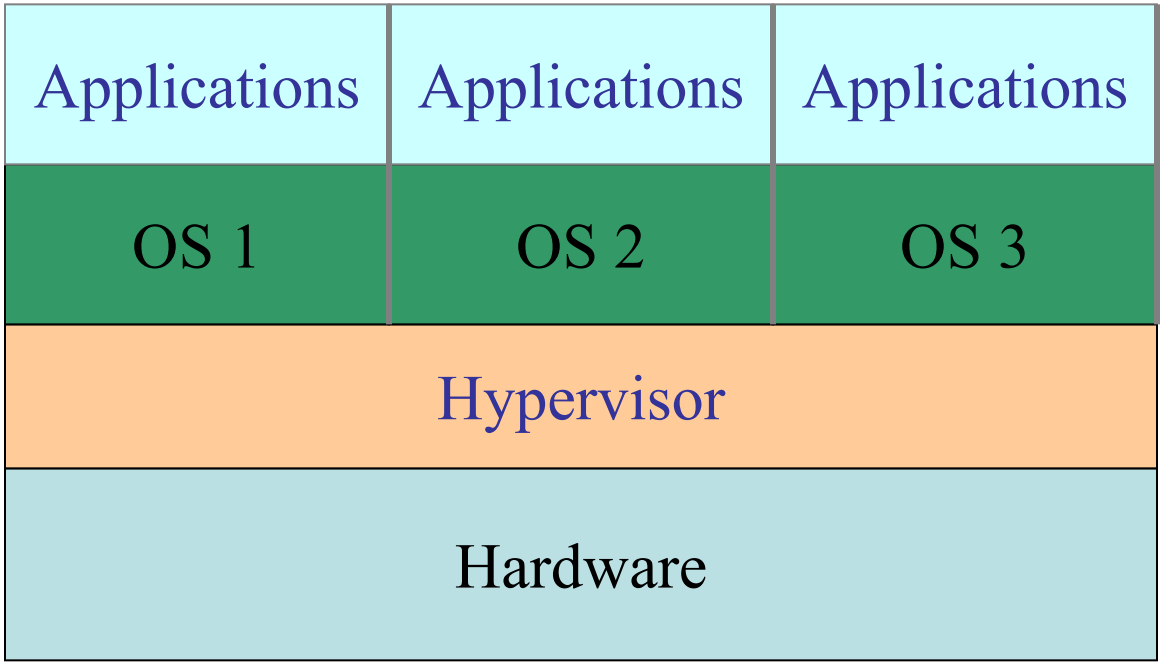
What are the advantages of virtualization?

BOSTON UNIVERSITY

What can you do with virt'n?

- Run one OS on another (e.g., Linux on Win/Mac)
  - cool, but not a big use case
- Run lots of machines on a single machine
- Migrate, pause/resume, snapshot, AMI files...

Type 1 Hypervisor



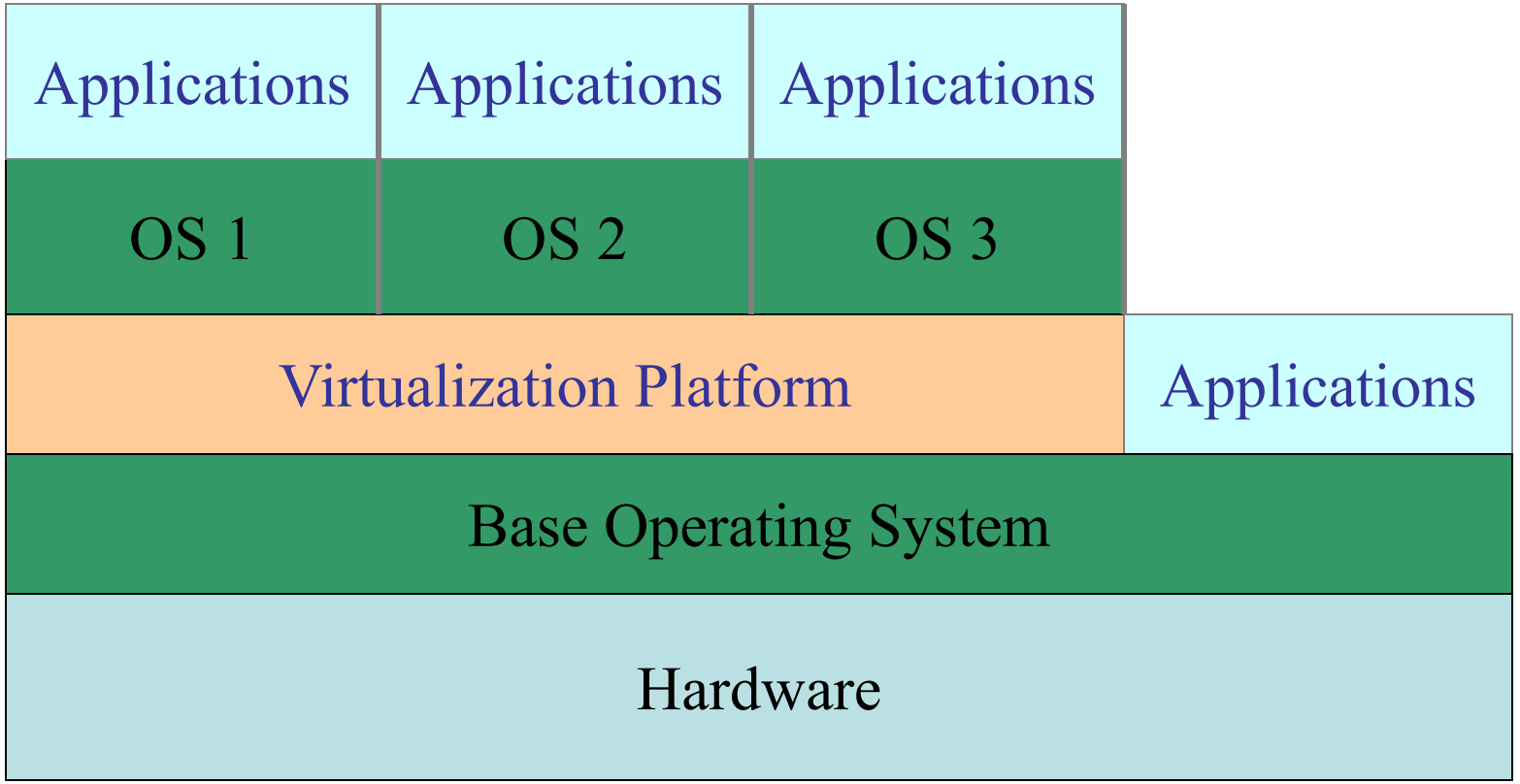
Microsoft Hyper-V

VMware vSphere 5.0

Xen Server™

BOSTON UNIVERSITY

Type 2 Hypervisor



What makes it **hard**?

- Isn't VirtualBox just another program?
- No.
  User programs can only run non-privileged instructions

```
$ cat foo.c
main()
{
    __asm__("movl    %eax, %cr3"::);
}
$ gcc foo.c
$ ./a.out
Segmentation fault (core dumped)
```

Some background

- Virtual memory / Address translation
  - each process has separate address space
  - can't access:
    - other process memory
    - operating system memory

- User/Supervisor mode
  - enter on interrupt / page fault / system call
  - do protected stuff
  - exit back to user program

How to run an OS on top of another

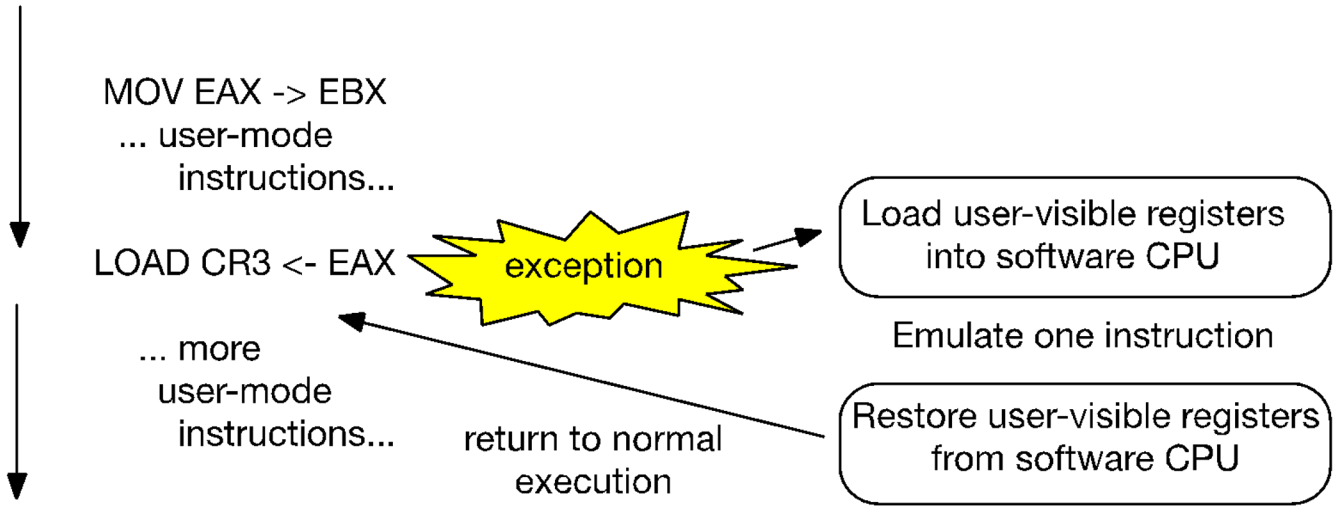
- Attempt 1 - Emulate every instruction

```
char memory[EMULATED_MEM_SIZE];
int R1, R2, R3, ...;
int PC, SP, CR1, CR2, CR3, ...;
bool S; /* supervisor mode */

while (true):
   instr = memory[PC];
   switch (instr):
     case "MOV R1 -> R2":
       R2 = R1; break;
     case "JMP":
       PC = … ; break;
     case "STORE Rx, <addr>":
            <paddr> = TLB[<addr>]
       if <paddr> is real memory:
         memory[paddr] = Rx
       else
         simulate_IO_access(paddr, Rx)
     .... Etc. (for ~1000 more instructions)
```

Trap and emulate ("classic" virt'n)

- Since you're emulating the same CPU…
- Run everything in **user mode**
- When privileged instruction traps, load the software emulator, run for one step, load results back into CPU and continue direct execution



But it doesn't work ☹

- Worked great on IBM machines from 1969 onwards
- But… x86 and ARM are "unvirtualizable" CPUs
- Some of the privileged instructions don't trap when you run them in user mode
  - Some are no-ops
  - Others do some but not all of what the privileged version does

Solution 1: Binary translation

- Guest user mode – direct execution
- When it tries to trap into the kernel, run all kernel code in emulation using BT
- That's what got VMware where they are today.

Faster emulation through BT

- Binary translation = JIT (Just-In-Time) compilation
- Translate code fragment X into code that does **what the emulator would do executing X**
- Typically expands the number of instructions executed (e.g., have to emulate MMU)
- Think of it as **eliminating the loop and switch overhead in the emulator**, plus you get to run an optimizer on the translated code.
- What Apple did for PPC->Intel switch (and 68k->PPC), [Rosetta]
- ~3x-10x slower for good implementations

BOSTON UNIVERSITY

Example

```
ADD   R1+R2 -> R2
ADD   R2+R3 -> R3
MUL   2,R3  -> R3
```

```
LOAD Rx <- &emulated_R1
LOAD Ry <- &emulated_R2
LOAD Rz <- &emulated_R3
ADD Rx,Ry -> Ry
ADD Ry,Rz -> Rz
MUL 2,Rz  -> Rz
STORE Ry -> &emulated_R2
STORE Rz -> &emulated_R3
RET
```

BOSTON UNIVERSITY

Virtualized Memory

- Problem – need 2 levels of translation
  - guest virtual -> “fake physical”
  - “fake physical” -> real physical address
- But the CPU only has one level of translation
- Solution: fake page tables

Shadow Page Tables

- Emulated CPU (what the guest sees) has CR3 pointing to “fake” page tables
- Real CPU has CR3 pointing to real page tables
- On a real page fault, first check to see if there’s a fake page table entry (guest virt -> fake phys)
  - If yes: calculate fake phys->real phys
    install guest virt -> real phys in real page table
  - Otherwise: pass the page fault to guest OS

I/O devices

- Device registers are just addresses in the physical address space
- So take page faults on them, and emulate what the real I/O device would do
- Can implement fake devices; e.g., disk in a file

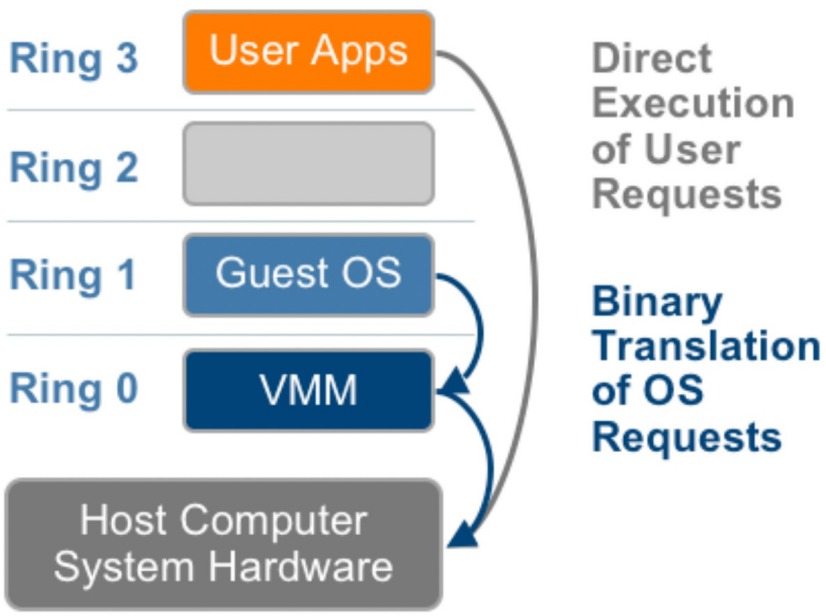
Solution 2: Paravirtualization (vs. Full virtualization)

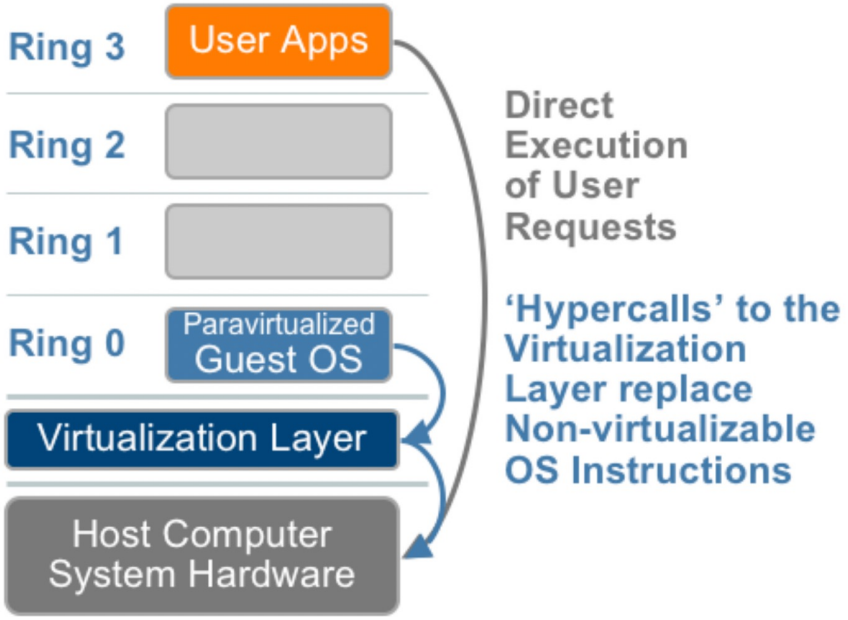
- When you don't really need full hardware virtualization
- Write an OS for running other operating systems
  - System calls are things like "add virtual memory mapping"
  - Modifying guest OS is relatively easy vs. sophisticated BT
- Modify guest OS to run on it
  - Linux/arch/x86/xen/*

Paravirtualized I/O devices

- Everyone uses them
- No need to rewrite the OS – just load a device driver.
- Because taking page faults, disassembling instructions and grabbing values out of registers isn't the most efficient API

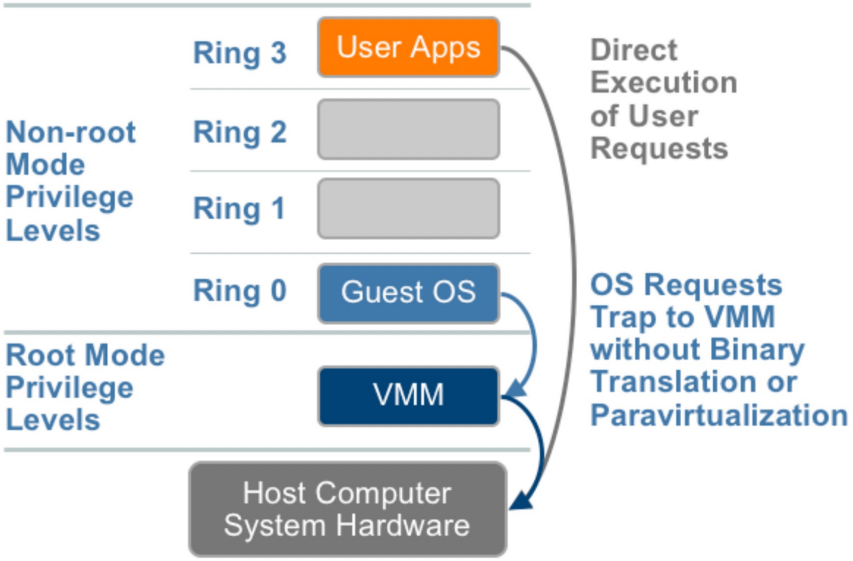
BT-based Full Virtualization vs Paravirtualization





Solution 3: New hardware-assisted virtualization

- 3 privilege modes:
  - User mode
  - Supervisor mode
  - Hypervisor mode
- All sorts of settings for when to trap into hypervisor mode.



But I/O still sucks…

- Paravirtualization still requires trap to hypervisor
- **Lets throw more HW at the problem:**
  - IOMMU – keep device from DMAing to wrong VM's memory
  - Single Root I/O Virtualization (SR-IOV)
    - Device exposes multiple set of registers mapped directly into VM – especially used for NIC that can support 100s
  - Share enough information that, if OS running, interrupt goes directly to it.
- This is just getting common today
- Note not used in today's clouds; mostly for accounting reasons

Live Migration/vMotion

- Moving a VM from one computer to another
- Standard approach:
  - 1 copy pages while running, detect all modified pages
  - continue to copy until working set small
  - pause and finish copy
  - same as what we learned for fork
- Typically assumes network mounted storage
- Exploits standard networking techniques to advertise IP new location

BOSTON UNIVERSITY

Use cases

- Server consolidation - if load imbalanced, can move VMs around
- Dealing with server failures
- Move VMs and power off servers
- Key feature: Resource pools
  - shares/limits/reservation
  - enables administrator to control resource use across many VMs
  - key to increasing utilization of data center
- Not used in most of today's clouds

BOSTON UNIVERSITY

Concluding remarks

- Virtualization has transformed data center
  - Server consolidation, management, high availability, …
- Pervasiveness of virtualization resulted in hardware changing to support it.
- Today, with the most modern HW, virtualization is easy and there is little cost for virtualization.

BOSTON UNIVERSITY

Lessons learned

1. A level of indirection solves a huge set of problems; VMware became a $30Billion dollar company
2. Mendel/VMware had no idea what they where unleashing; released one product at a time

3. Huge problem for large companies to eat their own children

4. Get it working, then HW will catch up…

BOSTON UNIVERSITY

Xen and the Art of Virtualization

Philosophy

- Support user applications unmodified
- Minor changes to OS kernels to reduce complexity & increase performance: paravirtualization
- Goal: support 100s of VMs on a single server
  - Strong performance isolation
    - bug, fork bomb….

Experience

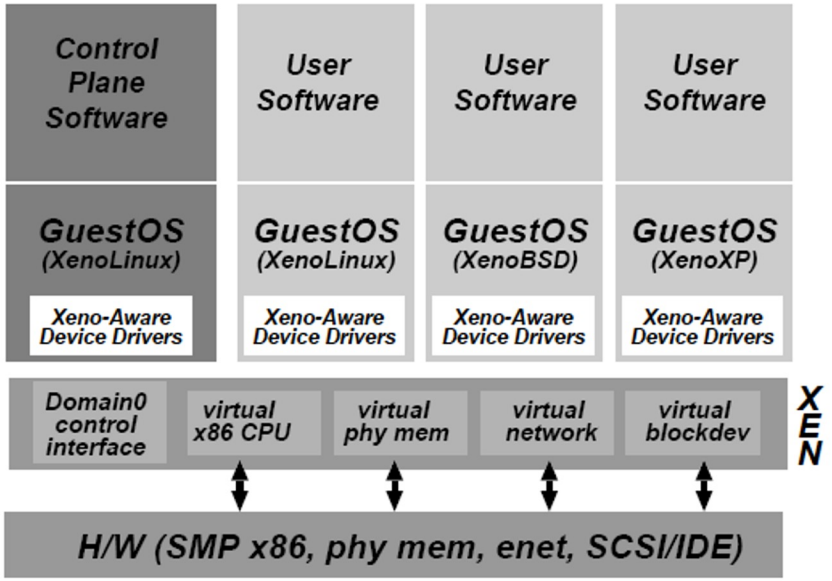
OS subsection	# lines	
	Linux	XP
Architecture-independent	78	1299
Virtual network driver	484	–
Virtual block-device driver	1070	–
Xen-specific (non-driver)	1363	3321
Total	**2995**	**4620**
(Portion of total x86 code base	**1.36%**	**0.04%)**

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).

BOSTON UNIVERSITY

Architecture

- Very simple base hypervisor
- Domain0 hosts the application-level management software & I/O control



Memory Management tricks

- Xen exists at the top 64MB of every address space
  - Avoid TLB flushing when an guest OS enter/exist Xen
- OS creates page tables, sends to Xen, has read access; no shadow page tables
- Writes are validated by Xen, changes can be batched

CPU tricks

- Guest runs at lower level of privilege: ring 1, while hypervisor is in ring 0 on x86; guest OS cannot directly execute privileged instructions
  - privileged instructions paravirtualized, OS needs to call Xen to install page table or yield processor
- System-call and page-fault handlers registered to Xen
- “fast handlers” for system calls, Xen isn’t involved
  - goes to ring 1 bypassing ring 0; validated by Xen when installed in hardware table

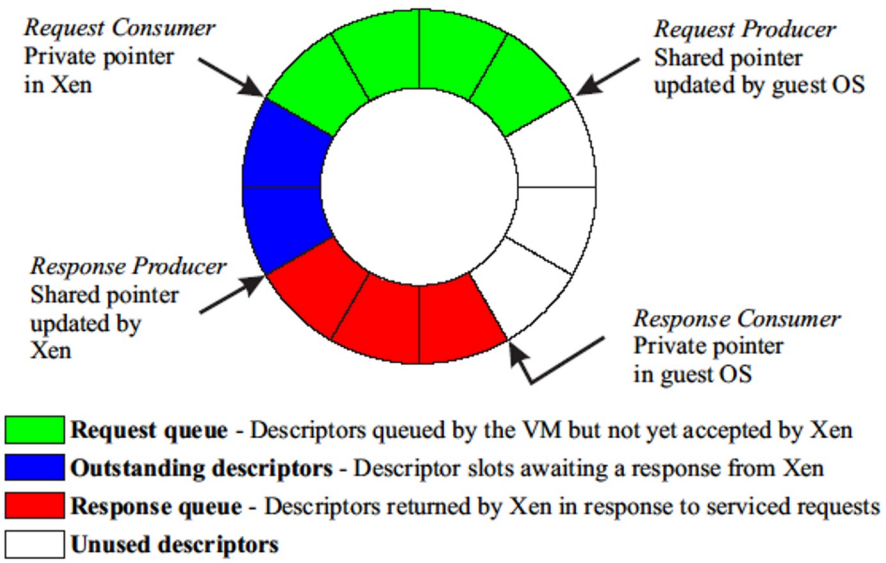
Time and Timers

- Xen provides each guest OS with
  - Real time (since machine boot)
  - Virtual time (time spent for execution)
  - Wall-clock time

- Each guest OS can program a pair of alarm timers
  - Real time
  - Virtual time

Data Transfer: I/O Rings

- Each request has id
- Response has same id, so can handle out of order
- Queue descriptors with pointers to data enables zero-copy



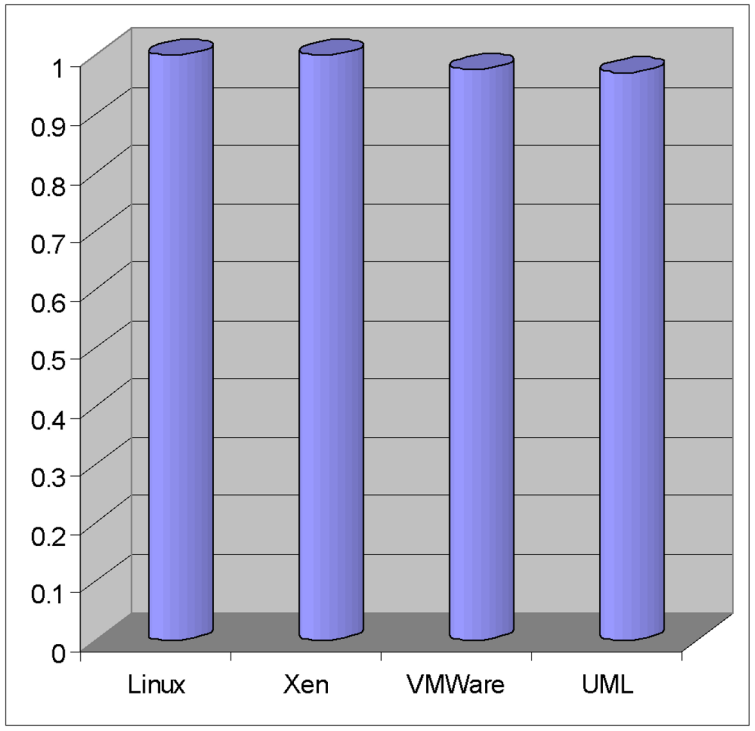
Network

- Xen provides simple virtual firewall-router in hypervisor
  - Dom0 controls network filters and routing rules
- Each domain has network interface attached to the router - two I/O rings: 1) transmit, 2) receive
- To send a packet, enqueue a buffer descriptor into the transmit ring
- Use scatter-gather DMA (no packet copying)

Disk

- Only Domain0 has direct access to disks
- Other domains need to use virtual block devices
  - List of extents on disk - translation table provided by Dom0
  - on disk request, xen translates, and enqueues the corresponding request
  - Disk DMAs directly into guest pages

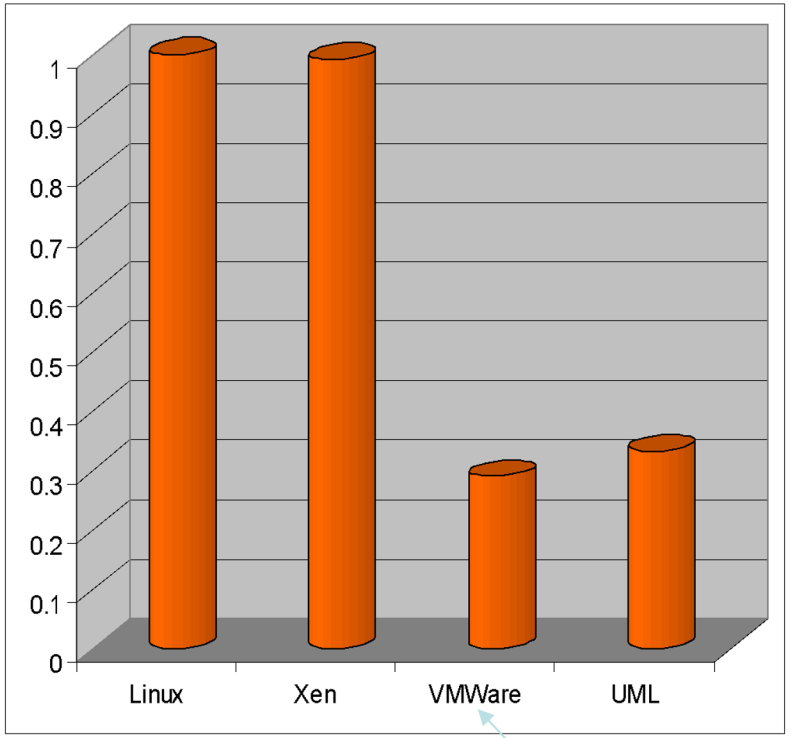
Relative Performance



SPEC INT2000 score

CPU Intensive

Little I/O and OS interaction



SPEC WEB99

network and disk intensive

workstation

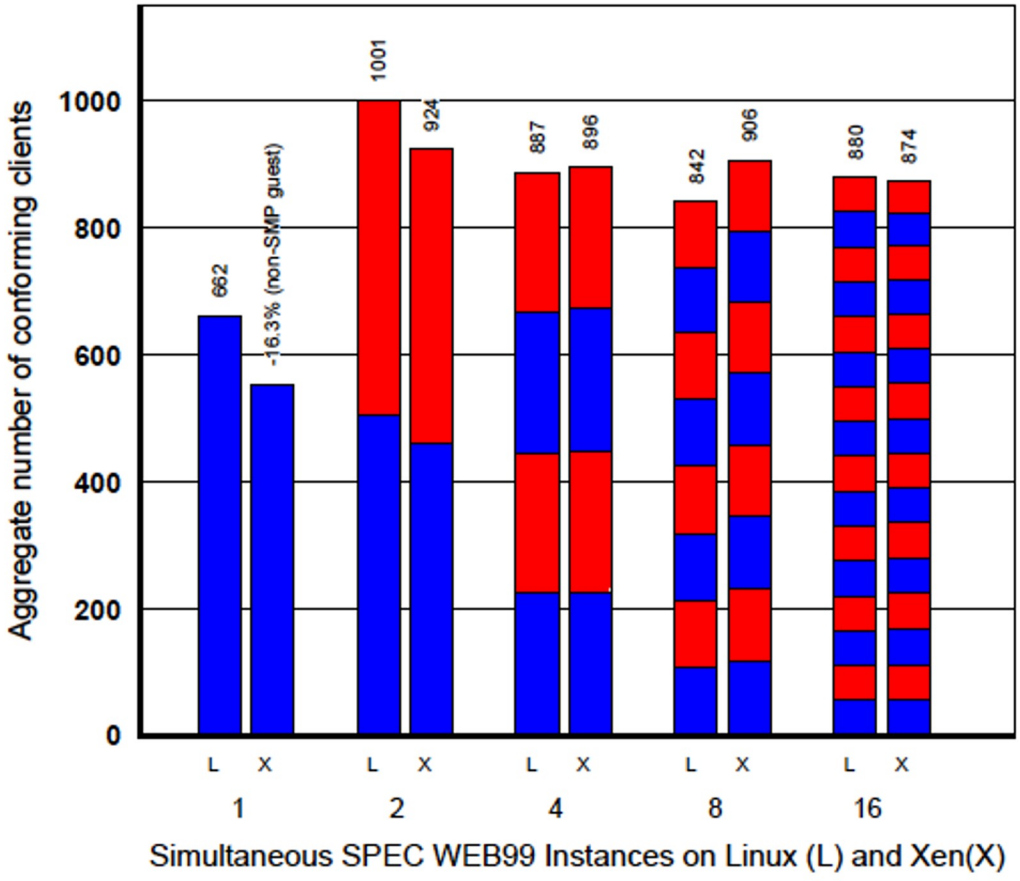
Concurrent Virtual Machines

Multiple Apache processes in Linux

vs.

One Apache process in each guest OS

Similar performance to running seperate processes

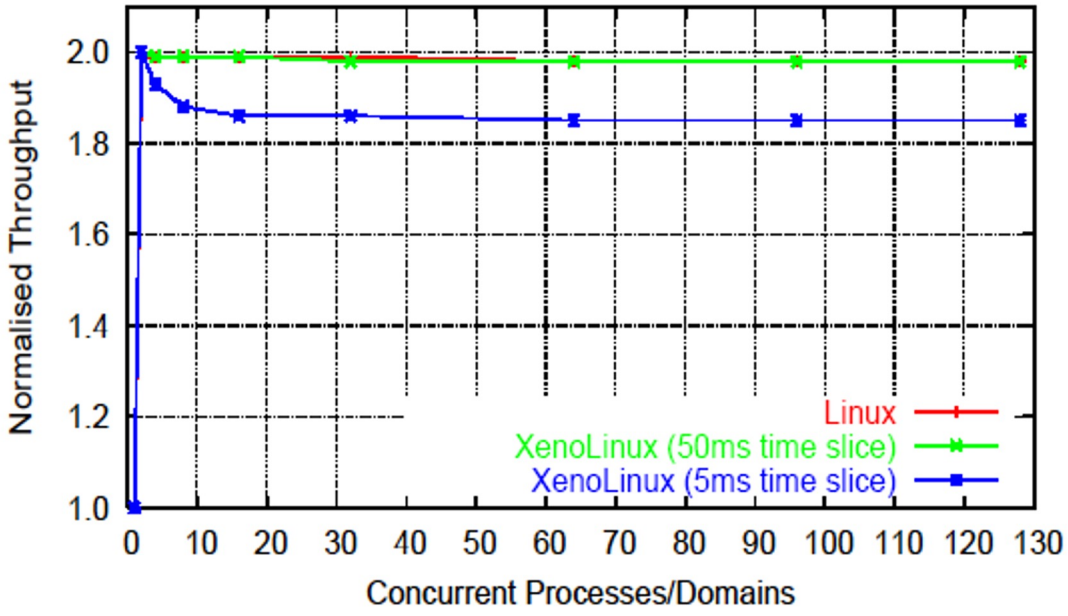


Performance Isolation

- 4 Domains configured with equal resources
- 2 running benchmarks
- 1 running dd - disk bandwidth hog
- 1 running a fork bomb in the background
- the 2 antisocial domains contributed only 4% performance degradation
- Under native linux huge degradation

BOSTON UNIVERSITY

Scalability



Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains

BOSTON UNIVERSITY

Comments

- Demonstrated good performance for I/O intensive, direct access through kernel
- Can support lots of concurrent virtual machines
- Good performance isolation
- Can run lots of CPU intensive applications

Results

- Released Open Source, company to support
- After 2 years trying to release rhype, Xen came up (not only IBM researchers got it).
  - IBM got permission to release rhype:
    - http://www.techrepublic.com/article/ibm-hypervisor-software-makes-stealth-debut/
  - If they agreed to work on Xen ☹, and didn't accept patches
- Amazon created EC2 based on Xen – open source
- Citrix acquired Xen for $500 Million

BOSTON UNIVERSITY

Concluding remarks & lessons

- In paper Domain0 just control:
  - eventually host back end drivers… simply hypervisor
- All the cute tricks of Xen largely irrelevant
  - Shared address space – HW support
  - Page flipping went away
- Xen was the right project at the right time:
  - OpenSource alternative to VMware, enabled first IaaS cloud
- Open Source community critical:
  - Xen eventually failed… poor support for community, took too long to get into Linux, …
  - KVM is eating its lunch now (Type 2)
- Stick to your guns:
  - Will never know what would have happened if I had just released rhype

BOSTON UNIVERSITY

Q&A