Overview of Virtualization

Instructor: Alan Liu
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

<table>
<thead>
<tr>
<th>OS subsection</th>
<th># lines</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linux</td>
<td>XP</td>
</tr>
<tr>
<td>Architecture-independent</td>
<td>78</td>
<td>1299</td>
</tr>
<tr>
<td>Virtual network driver</td>
<td>484</td>
<td>-</td>
</tr>
<tr>
<td>Virtual block-device driver</td>
<td>1070</td>
<td>-</td>
</tr>
<tr>
<td>Xen-specific (non-driver)</td>
<td>1363</td>
<td>3321</td>
</tr>
<tr>
<td>Total</td>
<td>2995</td>
<td>4620</td>
</tr>
<tr>
<td>(Portion of total x86 code base)</td>
<td>1.36%</td>
<td>0.04%</td>
</tr>
</tbody>
</table>

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).
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/exit 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
Concurrent Virtual Machines

Multiple Apache processes in Linux vs. One Apache process in each guest OS

Similar performance to running separate 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
Scalability

Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains
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:
    • 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
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
Container-based Operating System Virtualization:

A Scalable, High-performance Alternative to Hypervisors
Context

- Operating systems provide rich sharing and poor isolation
- Virtualization provides strong isolation and poor sharing
- Types of sharing:
  - logical
  - resources
- Many scenarios require (relatively) strong isolation, but require higher efficiency for sharing resources: PlanetLab, HPC, Grid, web/game hosting.
What is the fundamental difference with virtualization?

- With containers, the interface to the container is the ABI of the OS kernel
- With virtualization, the interface is the HW
Architecture

- Primary OS for admin, similar Xen
- Each VM container owns FS
- Shared kernel
Efficiency vs. Isolation

- **Efficiency:**
  - throughput, latency
  - number of concurrent VMs

- **Isolation:**
  - fault isolation
  - resource isolation
  - security isolation
Isolates OS objects vs. HW

• Separate name spaces – contexts:
  • E.g., PID space, sockets, ptys,

• Access control - filters:
  • E.g. shared file system space, network (in their design)

• A combination:
  • Chroot then hard links to common files: unification
Advantages of containers

• Easy to exploit OS mechanisms for sharing:
  • Typical Linux servers 500MB
  • 10 unified servers only about 700MB
  • Share file system cache

• Much faster startup time - running app

• Easier to administer externally

• Direct access to network/disk; no virtualization penalty
Limitations

• Can’t load a kernel module

• Can’t run windows & linux

• Less secure: Spectre, Meltdown
### Features

<table>
<thead>
<tr>
<th>Features</th>
<th>Hypervisor</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Kernels</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Administrative power (root)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Checkpoint &amp; Resume</td>
<td>✓</td>
<td>✗ [15,23,18]</td>
</tr>
<tr>
<td>Live Migration</td>
<td>✓</td>
<td>✗ [23,18]</td>
</tr>
<tr>
<td>Live System Update</td>
<td>✓</td>
<td>✗ [18]</td>
</tr>
</tbody>
</table>
Micro-benchmark results

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Linux-UP</th>
<th>VServer-UP</th>
<th>Xen3-UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>fork process</td>
<td>86.50</td>
<td>86.90</td>
<td>271.90</td>
</tr>
<tr>
<td>exec process</td>
<td>299.80</td>
<td>302.00</td>
<td>734.70</td>
</tr>
<tr>
<td>sh process</td>
<td>968.10</td>
<td>977.70</td>
<td>1893.30</td>
</tr>
<tr>
<td>ctx (16p/64K)</td>
<td>3.38</td>
<td>3.81</td>
<td>6.02</td>
</tr>
<tr>
<td>mmap (64MB)</td>
<td>377.00</td>
<td>379.00</td>
<td>1234.60</td>
</tr>
<tr>
<td>mmap (256MB)</td>
<td>1491.70</td>
<td>1498.00</td>
<td>4847.30</td>
</tr>
<tr>
<td>page fault</td>
<td>1.03</td>
<td>1.03</td>
<td>3.21</td>
</tr>
</tbody>
</table>

- Big advantage over Xen, since no hypercall
- Goes away with modern HW
Larger performance result

• Shows improved network use at reduced CPU utilization
  • Problem need to go through separate Dom0 Xen; containers operate at native speed
  • ESX addressed by driver in hypervisor, & with modern SRIOV hardware this is going away

• Other results show:
  • 2x improved server performance, better utilization…
Where are things today

• With modern HW, the performance issues virtualization gone away
• Containers still don’t properly support migration (WIP)
• Containers intrinsically don’t allow different kernels, kernel modules, different Oses
• Most people believe virtualization more secure
• Virtualization still has problems with memory if images very similar (de-duplication partially addresses)
• Containers much faster to start up.
• Containers increasingly popular Kubernetes, OpenShift, …
It's a weird world

- Dominant compute environment increasingly containers

- In AWS and Azure, VM is the base environment
  - Customers and platform support Kubernetes/containerized environments on top

- In Google, containers/Borg is the base
  - they run VMs on top of containers to isolate tenants
  - they/tenants run kubernetes on top of those VMs
Side notes

- Work was done in PlanetLab testbed:
  - lets researchers perform planetary-scale research

- PlantLab -> Genie -> CloudLab->?

- Open Cloud Testbed
Discussion

- Any team has met with the mentor(s)?
  - How was the initial contact?

- Any teammate still “unreachable”?

- Self-Introduction