Xen and the Art of Virtualization

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield

Presented by Thomas DuBuisson
Outline

- Motivation
- Design – What is the goal?
- Xen Drawing – What does it look like?
- Design Details – How?
- Performance – How Well?
- Beyond the Paper
Motivation

- server consolidation
- co-located hosting facilities
- distributed web services
- secure computing platforms
- mobility

Requiring

- Numerous simultaneous OSes
- Isolation of:
  - IO demands
  - Computation
  - Memory
- Minimal Overhead
Design – Basic Concepts

● Unlike micro-kernels of the 90s, hypervisors expose an idealized hardware interface to the higher level processes.

● Unlike typical VMMs, the OS must be modified to cooperate with the hypervisor. Not just for performance, but for basic operation.
Design – Paravirtualization

- Emulation is expensive – minor modifications* to the guest OS, making it more virtualization friendly and has huge performance payoffs.
- Application source code bases are huge – ABI must remain compatible.
- OSes altered in this way are called Paravirtual Machines (PVMs)

* 3000 lines of the Linux kernel sufficed
** The Xen patch to the Linux kernel is now over 14000 line
Typical View of Xen

- Interface Provides
  - CPUs
  - Memory
  - IO Devices
  - Management
Now for details!

- Memory Management
- CPU Virtualization
- IPC
  - IO Rings
  - Grant Tables
- Device Access
Memory Management

- Xen was originally x86 specific
  - No tagged TLB
  - No software managed TLB
- Guests are provided with a portion of machine memory from which they can allocate to their processes.
  - No over-committing of physical memory
  - Instead of modifying the page tables, PVMs perform hypercalls requesting Xen make the desired modification.
Memory Management

- All Page Table and Segment Descriptor updates are validated by Xen.
- On page faults, Xen copies the faulting address from CR2 into a stack frame and sends an event to the corresponding guest.
CPU Virtualization

- Xen runs in privileged mode (ring 0)
- Guest OS kernels run unprivileged (exact mode varies by architecture)
- Protected operations instead are performed by hypercalls from the guest to Xen.
CPU Virtualization

- Exceptions are propagated to the guest from Xen via event channels.
- Exceptions from system calls call directly from application into the guest OS.
  - Guest OS registers a 'fast' exception handler.
  - Xen validates the address is part of the guest address space and installs the handler.
Inter VM Communication (IVC)

- Hypercalls
  - Like syscalls, but occur from PVM to Xen

- Events
  - Xen sets a bitmask to specify which event(s) occurred then calls a previously registered handler belonging to the relevant domain.
Advanced IVC: IO Rings and Grant Tables

- IO Rings allow high bandwidth two-way communication:
  - Each cell of the ring contains a grant reference
  - Grant references indicate the ability to map memory or completely transfer a page of memory
IO Devices

- Network cards
  - Implemented as two rings (transmit, receive)
  - Rings allow communication between a 'front-end' and 'back-end' network driver.

- Guests are assigned virtual block devices
  - Access to devices is governed by a round robbin scheduler.
Control

- Start new domains? Domain 0
- Drivers for physical devices? Domain 0
- Run device back-ends? Domain 0
- What is Domain 0? Linux
Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).
Performance (Net)

- Near native performance for typical MTU
- Note the lack of CPU usage data during network activity
- Comparison with VMWare Workstation rather unfair

<table>
<thead>
<tr>
<th></th>
<th>TCP MTU 1500 TX</th>
<th>TCP MTU 1500 RX</th>
<th>TCP MTU 500 TX</th>
<th>TCP MTU 500 RX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>897</td>
<td>897</td>
<td>602</td>
<td>544</td>
</tr>
<tr>
<td>Xen</td>
<td>897 (-0%)</td>
<td>897 (-0%)</td>
<td>516 (-14%)</td>
<td>467 (-14%)</td>
</tr>
<tr>
<td>VMW</td>
<td>291 (-68%)</td>
<td>615 (-31%)</td>
<td>101 (-83%)</td>
<td>137 (-75%)</td>
</tr>
<tr>
<td>UML</td>
<td>165 (-82%)</td>
<td>203 (-77%)</td>
<td>61.1 (-90%)</td>
<td>91.4 (-83%)</td>
</tr>
</tbody>
</table>

Table 6: ttcp: Bandwidth in Mb/s
Performance (Micro)

- Typically better than Linux with SMP support.
- Even when significantly worse than Native, its much better than UML or VMW.
Performance (Multi-Guest)

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.
Performance (Scalability)

- Up to 192 domains... then crash

**Figure 6:** Normalized aggregate performance of a subset of SPEC CINT2000 running concurrently on 1-128 domains
How Xen Should Look

- Security wise it is similar to single server micro kernels
- Perhaps IO-MMU will help change that.
IO-Rings

• Not a Xen primitive!
  – Use the initial communication path with Dom0 to setups a grant-ref with the other domain via XenStore or similar. Yes, this is a hack.
  – Mapping in the grant-ref results in a shared page between the two domains desiring to communication. The IO-Ring may reside here.
  – The IO-Ring is just a fast way to communicate more grant references!
Non-Standard PVMs

- Linux, *BSD, and Windows XP can all be ran as PVMs to some extent.
- Traditionally, these take up significant memory (which we can't over-allocate!)

- Smaller, special, OSes exist:
  - Mini-OS: 'C' based for generic use.
    - 12000 lines
    - 4MB needed when running
  - StubDom: Based on Mini-OS
    - 4MB needed
    - Richer (standard C libraries included)
    - Intended for driver domains
  - HaLVM: Can compile Haskell code to run on bare Xen.
    - 6MB needed
    - Extremely rich, high level language
    - Experimental and not yet released to the public
IVC Issues to Consider

• For intra-PC network communication too many context switches are needed
  - A builds gntref (hypercall to signal B)
  - B receives ref (hypercall to transfer page)
  - B builds response, A receives response

• To setup Guest to Guest communication, XenStore is a free for all security blood bath.
Questions? Comments?