The Xen hypervisor

virtualizing a machine near you

Muli Ben-Yehuda

mulix@mulix.org

IBM Haifa Research Labs

TOC

- intro to Virtualization
- the Xen hypervisor
- CPU virt.
- MMU virt.
- IO virt.
- status and future work

Introduction to Virtualization

- we’re used to a simple equation, one physical machine runs one OS at any given time. By virtualizing the machine, we are able to run several operating systems (and all of their applications) at the same time. Magic? almost.
- virtualization isn’t new... its been around since the 60’s and 70’s in IBM. Some of you may be also familiar with VMWare, which is a virtualization offering for x86. In this talk we will discuss the Xen Virtual Machine Manitor (I prefer the term hypervisor), an open source project which brings free software virtualization to a machine near you.

Why Virtualize?

- machine consolidation.
- testing.
- debugging.
- OS research.
- live updates (a.k.a. 24x7 uptime).
- diversity, a.k.a using the best tool for the job.

Quick Overview of Xen

- provides secure isolation
- provides resource control and QoS
- requires **minimal** operating systems changes, and no userspace changes
- supports x86, x86-64, ia64 and PPC in varying degrees of maturity
- supports Linux 2.4 and 2.6, NetBSD, FreeBSD, ...
- close to native performance!
- supports live migration of VMs
- widespread hardware support, including direct device access

Quick Overview of Xen cont'

- developed and maintained at the Cambridge University Systems Research Group, by Steven Hand, Ian Pratt, Keir Fraser, lots of others.
- contributions from Intel, AMD, HP, IBM, others.
- commercial backing available from `http://www.xensource.com/`

These slides are based on the Xen papers, presentations and code. You can find all of them at `http://www.cl.cam.ac.uk/Research/SRG/netos/xen/` and are encouraged to do so.

Full Virtualization

- full virtualiztion refers to running an unmodified OS on a virtual machine (e.g. VMWare). There are many ways to do this - for example binary rewriting of the running OS image.
- hardware support makes full virtualiztion much easier.
- x86 is notoriously difficult to virtualize because some priviledged instructions simply fail silently, rather than raising a trap.

Paravirtualization

Paravirtualization refers to modifying the OS to make virtualization faster - modifying the OS to run on the virtualized environment rather than on bare metal.

- easy to do when you have the source
- 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 priviledged instructions (e.g. cli, hlt, write to cr3) with hypervisor calls.
- core concept: modify the OS to the virtualized environment, but expose some details of the hardware for optimization.

CPU Virtualization

(All details refer to x86-32 - other archs are done differently)

- x86 CPUs have 4 modes of operation, known as the 4 rings. Ring 0 is the most priviledged and ring 3 the least.
- Xen runs in ring 0, where the kernel normally runs.
- The kernel is moved to ring 1 or 2. No OS has used either of these since OS2. This is the part that requires the operating system changes.
- Userspace runs in ring3 - just like it does today - no userspace changes necessary.
- Xen is mapped in every OS's address space at the top 64 MB of memory, to save a tlb flush.

CPU Virtualization cont'

- how to protect xen from malicious guests? we could use page tables switching, but that would be too slow. Instead segmentation is used.
- to perform a priviledged operation, the guest uses a hypercall that jumps to Xen
- it is possible to let system calls go from user-space to guest kernel directly, by verfiying the trap handler before it is installed.

virtualizing the MMU

Virtualizing the MMU is the most difficult task a hypervisor implementor faces. We will discuss:

- shadow page tables
- direct page table access

Shadow Page Tables

- when using shadow page tables, the 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 and is repsonsible for validating them and propagating changes to the hardware page tables and back.

Direct Page Tables access

Xen, being a paravirtualized architecture, can do this much more efficiently. Rather than keeping distinct page tables for itself and for the OS, 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 (cr3)
- 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.

Segmentation Support

- segementation support is required by the thread libraries for TLS (thread local storage)
- Xen provides virtualized GDT and LDT
- since Xen itself is protected by a segment, there can be no other segments that overlap with it
- NPT TLS uses uses segments in a way that conflicts with Xen - Xen has to use emulation and binary rewriting to deal with it.

IO handling

- when Xen boots up, it launches dom0, the first priviledged domain
- dom0 is a priviledged domain that can touch all hardware in the system (long term goal is to move all hardware handling to dom0)
- dom0 exports some subset of the the devices in the system to the other domains
- devices are exported via device channels - an asynch shared memory transport coupled with event rings for interrupts

IO handling cont'

- dom0 runs the backend of the device, which is exported to each domain via a frontend
  - netback, netfront
  - blockback, blockfront
  - there's a PCI pass through for other kinds of devices (e.g. sound)
- backends and frontends communicate via a high level device abstraction - block class, network class, etc
- domains other than dom0 may be granted physical device access, securely [as secure as the architecture allows, anyway]
- virtual PCI configuration space and virtual interrupts

Performance

Xen provides near native performance. What else is there to say? ;-)

Scalability

- Xen includes a balloon driver to reclain unused guest memory and give it to other guests.
- PAE36 work in progress, as is large SMP support (2-4 way SMP support works for xen and virtual CPUs for guests already work)
- multiple scheduling algorithms supported, multiple backend block drivers, multiple NICs, etc.

Status

- stable (2.0.5), testing and unstable (3.0) branches.
- 3.0 expected in July 2005.
- unstable really means unstable, progressing at a rapid pace.
- the management tools are being constantly rewritten ;-)
- x86 works very well, x86-64, IA64 and PPC work in progress.
- already includes suppot for Intel's VT/VX and AMD's Pacifica, which should allow running unmodified OS's - including Windows.

Future Work

- larger SMP support
- stability, robustness and performance - making Xen enterprise quality
- cluster load balancing
- system debugging, pause/replay, VM forking
- secure Xen (sHype)
- your project here :-)