

Xen and the Art of Virtualization

Barham, et al (Univ of Cambridge), SOSP03

Context: System Structure

Introduction

Returning to a old theme—the idea of virtualization.

New technology of powerful machines that allow it to happen.

Question: how to do it?

This approach: Xen, a virtual machine monitor (VMM). Supports concurrent execution of multiple commodity OSes.

Challenges

Challenges for partitioning a machine to accommodate multiple virtual machines.

- isolation—no adverse performance effects by one for another
- need to support a variety of OSes to accommodate a variety of applications
- performance overhead should be small

Xen Approach

Hosts multiple OSes, but requires some source code modification.

Linux port done, Windows XP in progress, some progress on NetBSD.

Checking current work shows ports for Linux 2.4 and 2.6, NetBSD, Plan 9 and FreeBSD. No Windows XP! (due to license restrictions). Up to version 3.0 of Xen.

Used to host multiple servers each in their own virtual machine (vs. using multiple machines to support multiple applications).

Target: up to 100 hosted OSes.

More on Xen Approach

Did not take the approach of *full virtualization*—exposing the entire physical hardware to each machine.

Support concept of *paravirtualization*—improved performance, but requires changes to the guest operating system. Means that virtual systems sometimes see real resources.

Does support the same *application binary interface (ABI)* so no changes needed to guest applications.

Design Principles

1. support unmodified application binaries is essential
2. support for multi-application operating systems is important
3. paravirtualization is necessary
4. do not completely hide effects of resource virtualization

Similarities with Denali project (U. Washington), but not as many virtual machines.

Xen Virtual Machine Interface

Summarized in Table 1. Lots of details on the Xen hypervisor.

Highlights:

- Memory mgmt is the hardest. Software-managed TLB helps. A tagged TLB is better. x86 does not have either! Operating systems must go through Xen to register and update page tables.
- 4 levels of privilege supported by the x86 architecture. Rings (highest to lowest priority). Normally Operating System is in ring 0 and application in ring 3, but for Xen, the virtual OS is in ring 1. Privileged instructions are handled and validated by Xen in ring 0. System calls handled by a registered *fast* exception handler that runs in ring 1. However, page faults must be handled by Xen in ring 0.
- Xen supports a lightweight event-delivery mechanism for sending asynchronous notifications to a domain. Used for delivering device I/O interrupts.

Porting Costs

Porting costs summarized in Table 2.

~3000 lines in Linux or 1.4%.

How much is reusable for the next port?

Control and Management

Design goal: separate policy and mechanism.

System structure shown in Figure 1 with XenLinux, et al running on top of the Xen hypervisor.

Domain0 runs control software in Xen. This is a special privileged domain created at boot time that can subsequently create other domains to run operating systems.

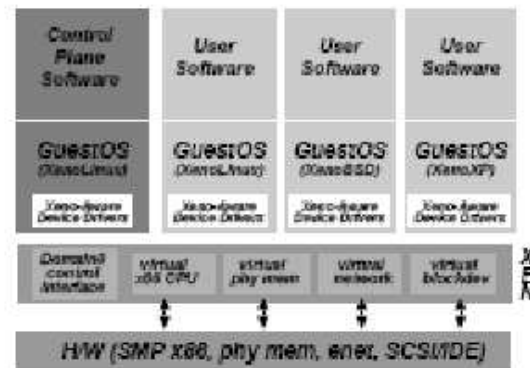


Figure 1: The structure of a machine running the Xen hypervisor, hosting a number of different guest operating systems, including *Domain0* running control software in a XenLinux environment.

Detailed Design

Highlights:

- synchronous *hypercall* from domain to Xen similar to a system call in a traditional operating system. Asynchronous event mechanism for Xen to domain communication.
- Try to minimize overhead in moving data vertically through rings.
- Scheduling of domains done using Borrowed Virtual Time (BVT) algorithm. Supports fast dispatch of events to domains.
- Time (Virtual and real timers for each domain):
 1. real—nanoseconds since boot time
 2. virtual—only when domain is executing
 3. wall-clock—offset added to real time.
- Xen maintains a shadow page table. Capability for domains to batch queue updates for page tables to minimize hypercalls.
- physical memory, network, disk, ...

Evaluation

Compare:

1. native Linux (L)
2. XenLinux (X)
3. VMware 3.2 (V)
4. User-Mode Linux (U)

Figure 3 shows performance for different application-level benchmarks. Xen is close to native performance.

Table 5 shows microbenchmarks using *lmbench*. Always the problem of trying to map microbenchmarks to application-level performance.

Table 6 for network-level benchmark measuring bandwidth.

Design goal of Xen was to avoid performance overhead—seems to have largely succeeded, although not the latest version of VMware.

Concurrent Application Performance

SPEC WEB99 over Xen vs running multiple copies on Linux. Xen actually did better for 4 and 8 copies while Linux did better for 1, 2 and 16 copies.

Disruptive applications caused more performance issues on Linux than separate copies using Xen (Figure 6).

Related Work

- Old idea—IBM VM/370
- VMware and Connectix—virtualize commodity PC hardware—provide full virtualization, but some performance cost
- Disco-ccNUMA machines—some operating system changes required
- IBM and zSeries uses paravirtualization
- Denali
- PlanetLab—distributed infrastructure

Summary

Current project:

<http://www.cl.cam.ac.uk/research/srg/netos/xen/>

Tradeoff between full virtualization and performance.

Not taking full virtualization approach affords better performance.