Xen as a test environment

Thomas L. Kula

University of Michigan

AFS & Kerberos Best Practices Workshop 2007

http://kula.tproa.net/talks/afsbpw2007/

What is Xen?

- Xen is a virtualization technology developed at the University of Cambridge Computer Laboratory

- Developed as part of a larger project called XenoServers, a public distributed computing utility project

- Development is sponsored by various academic and industry sources

- XenSource is a company founded by several of the Xen team members, offering various commercial products based on Xen

- The Xen hypervisor is released under the GPL

- Is available bundled with several commercial offerings, through XenSource and others

Okay, so what Xen?

Xen is virtualization

What is virtualization?

- Don't run an OS directly on hardware
- Run it on top of a *hypervisor*

Hyper*whatsit*?

- The hypervisor sits between the guest OSes and the physical hardware

- It is responsible for making each OS *think* it is running on a dedicated machine, while keeping them from stepping on each other

Isn't that slow?

- It typically is

- The hypervisor basically has to run the OS and watch for anything that would change the physical hardware state, trap it, and do something else with it

How does Xen get around this?

Xen uses para-virtualization

Para*whatsit*?

- Instead of running an unmodified OS, replace certain low level bits of the OS with calls to the hypervisor

- The *guest* OS and the hypervisor send messages to each other, to, for example, change MMU state or indicate an interrupt has happened

- The general idea is that modifying guest OSes a small amount so that they know they are running on a hypervisor yields significant performance gains

- Xen also presents to the guest OS virtual disk and ethernet devices
- The *host* OS provides these services on the real hardware
- Mapping virtual disks to files or partitions available to it
- Bridging virtual ethernet devices to physical network hardware, or otherwise providing networking for guests

- This para-virtualization is relatively small (about 50K lines of code)

- Allows for close cooperation between hypervisor and guest OSes

- For fully para-virtualized OSes, the overhead seems to be about $5 - 10$ percent

Other Xen features

- With proper CPU extensions (Intel's VT-x or AMD's Pacifica technologies) stock, unmodified OSes can run as guests
- Devices (disk, network, screen, keyboard, etc) must be emulated by Xen and the host OS
- Doesn't work for all OSes

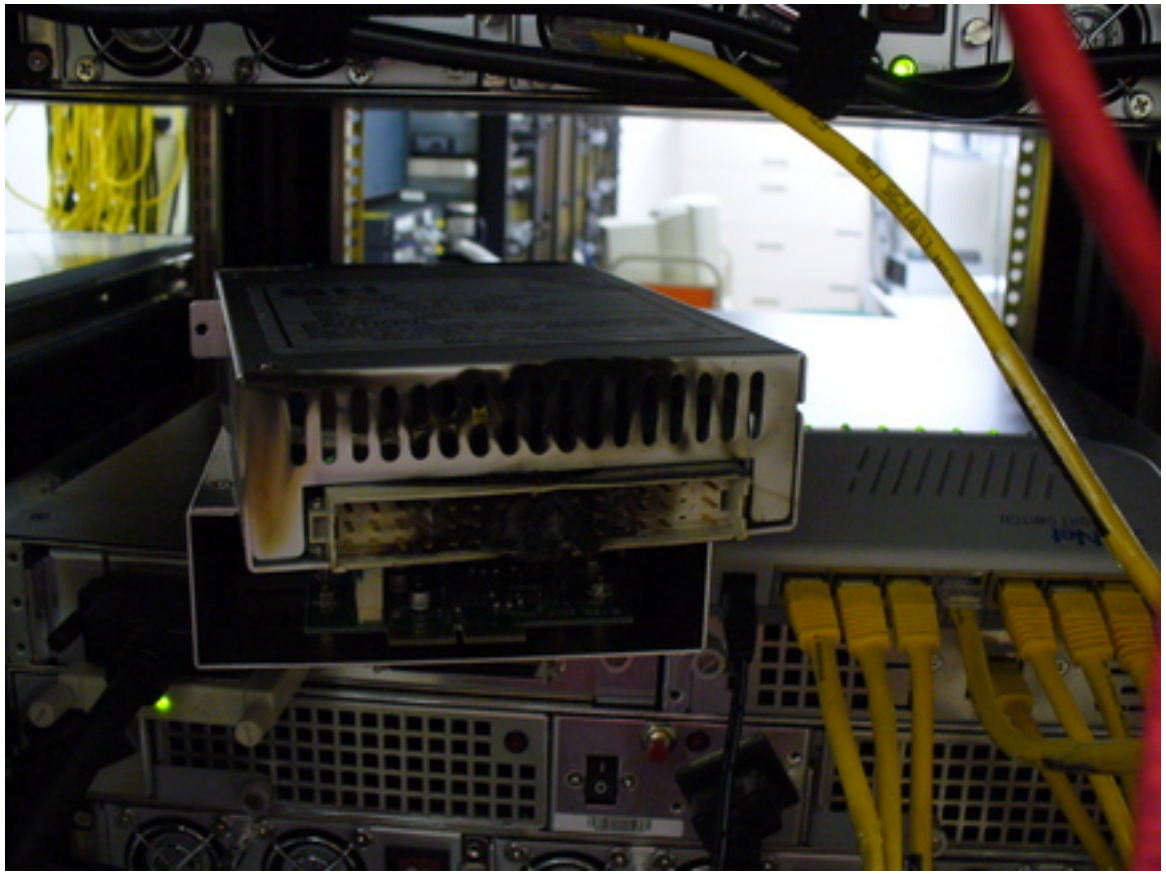
Other Xen features

- Xen also supports live migration from one host to another
- Various caveats apply here: host network devices must be on same broadcast domain, host storage for guest OS disks must be available to both host machines (e.g. SAN, dual-homed SCSI, etc.)

What does Xen run on?

Under Xen 3

- Host OSes: Linux 2.6 series, NetBSD 4.0_BETA, _BETA2 and -current
- Guest OSes: Linux 2.6 series, NetBSD 3.1, 4.0_BETA, _BETA2, -current, Solaris 10
- Various unmodified OSes with CPU support
- (Xen 2 supports others, like Linux 2.4, FreeBSD 5, NetBSD 2.0, Plan 9)



This has nothing to do with the talk, it's just a picture of a RAID we had catch on fire

What advantage does Xen offer someone tasked with running an AFS and/or Kerberos infrastructure?

- Ease in which one physical machine can be turned into multiple virtual machines

- Advantageous in test environments

- And when talking about AFS or Kerberos, testing quickly evolves into "several machines".

So how well does AFS/Kerberos run under Xen?

- On the most part, fairly well

- There have been some bumps

Example 1

- **Host:** NetBSD 4.0_BETA, Xen 3.0.2
- **Guests:** NetBSD 3.1
- **Kerberos:** Heimdal 0.72
- **AFS:** Arla 0.43 (client), OpenAFS 1.4.2
- The `tproa.net` cell, a whopping dozen users

```
kernel =
"/local/xen/service-m1.tproa.net/netbsd-XEN3_DOMU-nodiag.gz"
memory = 256
name = "service-m1"
vif = [ 'mac=00:16:3e:10:11:c4, bridge=bridge3' ]
disk = [
'file:/local/xen/service-m1.tproa.net/wd0.img,0x01,w',
'phy:/dev/wd2d,0x02,w', 'phy:/dev/wd3e,0x03,w',
'phy:/dev/wd3f,0x04,w' ]
root = "/dev/wd0d"
```

```
kraken# xm create -c service-m1
Using config file "service-m1".
Started domain service-m1
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005
The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
The Regents of the University of California.  All rights reserved.
NetBSD 3.1 (XEN3_DOMU-nodebug) #0:  Thu Nov 9 20:52:31 CST 2006
root@waffle-1.tproa.net:/usr/src/sys/arch/i386/compile/XEN3_DOMU-nodebug
total memory = 252 MB
avail memory = 246 MB
mainbus0 (root)
cpu0 at mainbus0:  (uniprocessor)
cpu0:  Intel (686-class), 2999.99 MHz, id 0xf62
cpu0:  features
bfebfbff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR>
cpu0:  features bfebfbff<PGE,MCA,CMOV,PAT,PSE36,CFLUSH,DS,ACPI,MMX>
cpu0:  features bfebfbff<FXSR,SSE,SSE2,SS,HTT,TM,SBF>
cpu0:  I-cache 12K uOp cache 8-way
cpu0:  ITLB 4K/4M: 128 entries
cpu0:  DTLB 4K/4M: 64 entries
hypervisor0 at mainbus0
```

Example 1

Caveats

- The stock NetBSD Xen domU kernel comes compiled with the options `DEBUG,DIAGNOSTIC`
- The stock Arla nnpfs kernel module is not compiled this way
- Solution: Recompile the kernel disabling those options

Example 1

Caveats

- Some isolated reports of a domU kernel panic with Arla
- Doesn't seem frequent (I've never experienced it, with some decent load)

Example 2

- **Host:** NetBSD 4.0_BETA, Xen 3.0.2
- **Guests:** Debian Etch (Kernel 2.6.16-xen)
- **Kerberos:** MIT Kerberos 1.44
- **AFS:** OpenAFS 1.4.2

```
kernel =
"/local/xen/shea.awesmoe.org/vmlinuz-2.6.16-xen.tproa"
memory = 128
name = "shea"
vif = [ 'mac=00:16:3e:10:11:38, bridge=bridge4' ]
disk =
['file:/local/xen/shea.awesmoe.org/debian.4-0.img,0x301,w']
root = "/dev/hda1 ro"
```

```
kraken# xm create -c shea
Using config file "shea".
Started domain shea
Linux version 2.6.16-xen (kula@shea.awesmoe.org) (gcc version 4.1.2
20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Thu May 3 16:40:12 EDT
2007
BIOS-provided physical RAM map:
Xen:  0000000000000000 - 0000000008000000 (usable)
136MB LOWMEM available.
ACPI in unprivileged domain disabled
IRQ lockup detection disabled
Built 1 zonelists
Kernel command line:  root=/dev/hda1 ro
Enabling fast FPU save and restore...  done.
Enabling unmasked SIMD FPU exception support...  done.
Initializing CPU#0
PID hash table entries:  1024 (order:  10, 16384 bytes)
Xen reported:  2999.994 MHz processor.
```

Example 2

Caveats

- Let's Do The Unknown Symbol Hustle!

```
Apr 27 16:55:23 shea kernel:  openafs:  module license
'http://www.openafs.org/dl/license10.html' taints kernel.
Apr 27 16:55:23 shea kernel:  openafs:  Unknown symbol
force_evtchn_callback
Apr 27 16:55:23 shea kernel:  openafs:  Unknown symbol
xen_features
```

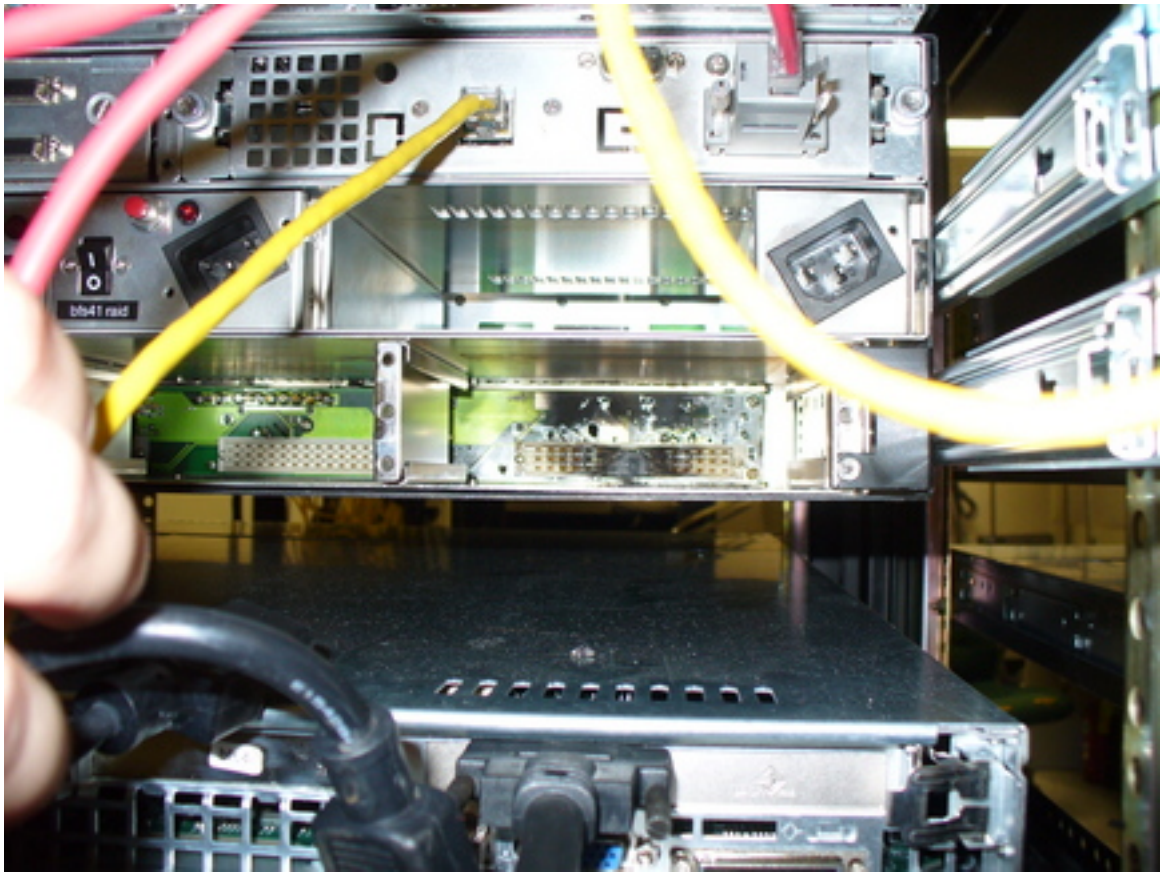
Example 2

Caveats

- This is fixed in Xen 3.0.3 and later
- The symbols become `EXPORT_SYMBOL` instead of `EXPORT_SYMBOL_GPL`

In General

- Things perform as if they are running on bare hardware



Mmm, fire

Performance?

- Didn't do any benchmarks
- Host environment is also running other services
- Don't have equivalent hardware to run bare-metal tests
- XenSource has a couple of papers, one they did, one an outside party did, comparing Xen performance to bare hardware

- Anecdotally, seems just as fast as bare hardware to me

So, could I use Xen in other situations?

Other Example Uses

- Co-host a KDC? An AFS-db server?
- Other specialized cell usage with requirements that can be fit on virtualized systems (small cell?)
- Don't share a physical host with other guests, but utilize live migration between hardware as part of a reliability plan?

Links

- Xen at the Cambridge Computer Laboratory:
  `http://www.cl.cam.ac.uk/research/srg/netos/xen/`

- XenSource: `http://www.xensource.com/`

- The Xen Wiki: `http://wiki.xensource.com/xenwiki/`

Thanks

- The Ugly Mug Cafe in Ypsilanti, Michigan, for high-quality caffeine and free wireless

Xen as a test environment

Thomas L. Kula

University of Michigan

AFS & Kerberos Best Practices Workshop 2007

http://kula.tproa.net/talks/afsbpw2007/