Xen™ and the Art of Virtualization

Ian Pratt
XenSource Inc and
University of Cambridge
Outline

- Xen today
- Benchmarks
- New features
- Roadmap
- Questions
Xen Architecture

- **VM0**: Device Manager & Control s/w
  - GuestOS (XenLinux)
  - Back-End
  - Native Device Drivers

- **VM1**: Unmodified User Software
  - GuestOS (Xen Linux)
  - Front-End Device Drivers

- **VM2**: Unmodified User Software
  - GuestOS (XenLinux)
  - SMP
  - Front-End Device Drivers

- **VM3**: Unmodified User Software
  - Unmodified GuestOS (Windows and legacy Linux)
  - Front-End Device Drivers

**Xen Hypervisor**
- Control IF
- Safe HW IF
- Event Channel
- Virtual CPU
- Virtual MMU

**Hardware (SMP, MMU, physical memory, Ethernet, SCSI/IDE)**

**VT/AMDV “HVM”**
Xen 3.0 Highlights

- i686, x86_64 and ia64 support
  - Power support ready to merge
- Leading performance
- Secure isolation and QoS control
- SMP guests
- Hotplug CPUs, memory and devices
- Guest save/restore and live relocation
- VT/AMDV support: "HVM"
  - Run unmodified guest kernels
  - Supports progressive paravirtualization
**Xen Use Cases**

- Consolidate under-utilized servers
- Utilize multicore
- Avoid downtime with VM Relocation
- Dynamically re-balance *workload* to guarantee application SLAs
- Enforce security policy
- Debugging, monitoring
Xen 3 API support

- Linux 2.6.16/17/18 and -rc/-tip
- 2.6.5 2.6.9.EL 2.4.21.EL
- Available in distros: FC4, FC5, FC6, SuSELinux10, SLES10, Ubuntu, Gentoo,...
- NetBSD 3, FreeBSD 7.0, OpenSolaris 10, Plan9, minix, ...

- Linux upstream submission process agreed at Kernel Summit
dbench

![Bar chart showing MB/s vs # processes for Native and Xen.]
Kernel build

32b PAE; Parallel make, 4 processes per CPU
HVM Architecture

Domain 0
- Linux xen64
- Control Panel (xm/xend)
- Device Models

Domain N
- Linux xen64
- Native Device Drivers
- Native Device Drivers
- Front end Virtual Drivers

Guest VM (VMX)
(32-bit)
- Unmodified OS
- Guest BIOS
- Virtual Platform

Guest VM (VMX)
(64-bit)
- Unmodified OS
- Guest BIOS
- Virtual Platform

Callback / Hypercall
- Event channel
- PIC/IOAPIC
- Emulation

Control Interface
- Processor
- Memory

Scheduler
- Event Channel
- I/O: PIT, APIC, PIC, IOAPIC

Hypercalls

Xen Hypervisor
Progressive paravirtualization

- Hypercall API available to HVM guests
- Selectively add PV extensions to optimize
  - Net and Block IO
  - XenPIC (event channels)
  - MMU operations
    - multicast TLB flush
    - PTE updates (faster than page fault)
    - Page sharing
  - Time
  - CPU and memory hotplug
PV Driver performance

![Graph showing PV Driver performance](image-url)

Measured with ttcp, 1500 byte MTU

- **rx**: blue bars
- **tx**: red bars

**y-axis**: Mb/s

**x-axis**:
- ioemu
- PV-on-HVM
- PV
IOEmu stage #2

Domain 0
Linux xen64
Control Panel (xen/xend)

Domain N
Linux xen64

Native Device Drivers

Backend Virtual driver

Front end Virtual Drivers

Callback / Hypercall

Event channel

Control Interface
Scheduler
Event Channel
Hypercalls

Processor
Memory
I/O: PIT, APIC, PIC, IOAPIC

Xen Hypervisor

Guest VM (VMX) (32-bit)
Unmodified OS
FE Virtual Drivers
Guest BIOS
Virtual Platform
VMExit
IO Emulation

Guest VM (VMX) (64-bit)
Unmodified OS
FE Virtual Drivers
Guest BIOS
Virtual Platform
VMExit
IO Emulation

I/O Emulation

FE Virtual Drivers

Virtual Platform
VMExit

IO Emulation

 FE Virtual Drivers

Virtual Platform
VMExit

IO Emulation
HVM Performance

- Very application dependent
  - 5% SPECJBB (0.5% for fully PV)
  - OS-intensive applications suffer rather more
  - Performance certainly in-line with existing products
  - Hardware support gets better every new CPU

- More optimizations forthcoming
  - “V2E” for trap-intensive code sequences
  - New shadow pagetable code
Shadow Pagetables

- Guest reads
- Guest writes
- Accessed & dirty bits
- Updates
- Virtual $\rightarrow$ Pseudo-physical
- Virtual $\rightarrow$ Machine
- VMM
- Hardware
**Virtual Disk Storage**

- Most admins use LVM LV’s to store guest images
  - Not as intuitive as using files
  - Using ‘loop’ driver is dangerous, as is dm-snap for CoW
- “Blktap” provides an alternative:
  - Allows all block requests to be serviced in user-space using zero-copy AIO approach
Blktap and tapdisk plug-ins

- Char device mapped by user-space driver
- Request/completion queues and data areas
- Grant table mapping for zero-copy to/from guest
- Flat files and qcow
- Sparse allocation, CoW, encryption, compression
- Correct metadata update safety
- Optimized qcow format
Blktap IO performance
Time API

- Shared info page contains per VCPU time records
  - “at TSC X the time was Y, and the current TSC frequency has been measured as Z”
  - gettime: Read current TSC and extrapolate
- When VCPUs migrate, update record for new physical CPU
- Periodic calibration of TSC's against pit/hpet
  - Works even on systems with un-synced TSCs
  - Update record after CPU frequency changes (p-state)
  - Also, resynchronize records after CPU halt
  - Only issue is thermal throttling
# Current Xen Status

<table>
<thead>
<tr>
<th></th>
<th>x86_32</th>
<th>x86_32p</th>
<th>x86_64</th>
<th>IA64</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privileged Domains</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guest Domains</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMP Guests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save/Restore/Migrate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;4GB memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Progressive PV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Driver Domains</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Xen Development Roadmap

- Performance tuning and optimization
  - Particularly for HVM and x86_64
- Enhanced control stack
- More automated system tuning
- Scalability and NUMA optimizations
- Better laptop/desktop support
  - OpenGL virtualization, power management
- Network optimizations
IO Virtualization

- IO virtualization in s/w incurs overhead
  - Latency vs. overhead tradeoff
    - More of an issue for network than storage
  - Can burn 10-30% more CPU than native
- Direct h/w access from VMs
  - Multiplexing and protection in h/w
  - Xen infiniband support
  - Smart NICs / HCAs
Xen Research Projects

- Whole-system pervasive debugging
  - Lightweight checkpointing and replay
  - Cluster/distributed system debugging
- Software implemented h/w fault tolerance
  - Exploit deterministic replay
  - Explore possibilities for replay on SMP systems
- Multi-level secure systems with Xen
  - XenSE/OpenTC : Cambridge, Intel, GCHQ, HP, ...
- VM forking
  - Lightweight service replication, isolation
  - UCSD Potemkin honeyfarm project
Conclusions

- Xen is a complete and robust hypervisor
- Outstanding performance
- Excellent resource control and protection
- Vibrant development community
- Strong vendor support

- Try the Xen demo CD to find out more! (or Fedora, SuSE, SLES etc)

- http://xensource.com/community
Thanks!

- If you’re interested in working on Xen we’re looking for developers to work in the University of Cambridge, and also XenSource’s UK and US offices.

- ian@xensource.com