Virtualization and BSD

- Approaches to Virtualization
- Benefits of Virtualization
- Para-virtualization in depth
- Para-virtualization on x86 and sparc64
Virtualization Approaches

- **Process Virtualization**
  - Jails
  - Vservers (Linux)
  - Zones (Solaris)

- **OS Virtualization (para-virtualization)**
  - Xen (Cambridge, UK)
  - T1 hypervisor (Sun)
  - VMI (VMware)
  - Phype (IBM)

- **CPU Virtualization (emulation)**
  - VMware
  - Virtual PC (Microsoft)
OS Virtualization Benefits for Development

- MMU interface can be decoupled from the OS
  - Provides future sun4v processors with the backward compatibility (on x86 by design)
- Debugging / fault isolating drivers by running them in their own domain
  - Faulty devices / drivers don't bring down the whole system
Virtualization Benefits for Administration

❖ Server Consolidation
  ● Servers typically average 10-20% utilization
  ● Large potential impact on power, space, and network ports

❖ Simplified Provisioning
  ● Allocating a new system can be just allocating an IP address and storage
OS Virtualization Benefits for Administration

- Reduction in planned downtime
  - Running virtual machines can be migrated across the network – actual “down time” determined by the size of the writable working set
- Consolidating different operating systems or different versions of the same operating system
- “Reboot” virtual machines independently of each other
- Natural interfaces for QoS across all resources
Paravirtualization In-Depth

- Memory Partitioning
- CPU Time
- CPU Privilege
- Devices / Interrupts
- Virtual Memory
Memory Allocation

- **Real / Physical**
  - Each guest is given a subset of the memory on the machine
  - Sun uses the traditional terminology for the distinction -- real addresses are addresses that guest can access and physical are the actual hardware addresses
- **Guests**
- **Physical / Machine**
  - Xen makes the same distinction but uses the terms from the earlier “Disco” papers
- **Balloon driver**
CPU Time

- Xen schedulers
  - Guest operating systems are multiplexed on the CPUs by a scheduler
  - A number of different schedulers exist
  - Difficult to get right, awkward interaction with driver domains, to work well with SMP requires some form of gang scheduling

- Sun4v strands
  - The T1 exposes 32 “strands”, the strands can be dynamically added to/removed from running domains
Xen 3.0 Architecture

VM0
- Device Manager & Control s/w
- GuestOS (XenLinux)
- Native Device Driver

VM1
- Unmodified User Software
- GuestOS (XenLinux)
- Native Device Driver

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

VM3
- Unmodified User Software
- Unmodified GuestOS (WinXP)
- Front-End Device Drivers

Xen Virtual Machine Monitor

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

Control IF
Safe HW IF
Event Channel
Virtual CPU
Virtual MMU

32/64bit
AGP
ACPI
PCI
Virtual Machine for SPARC

- Thin software layer between OS and platform hardware
- Para-virtualised OS
- Hypervisor + sun4v interface
  - Virtualises machine HW and isolates OS from register-level
  - Delivered with platform not OS
  - Not itself an OS
Sun4v Logical Domains Architecture

- Partitioning capability
  - Create virtual machines each with subset of resources
  - Protection & Isolation using HW+firmware combination
CPU Privilege

- **Xen ring/segment usage**
  - i386: Xen runs in ring 0, guest OS runs in ring 1
  - x86_64: most models lack segment checks in long mode – Xen runs in ring 0, guest OS runs in ring 3 with a different page directory from guest

- **Sun4v adds a hyperprivileged mode**
  - Sun4v adds a hpstate register (hyperprivileged pstate) some events that would previously cause a switch to privileged mode now switch to hyper-privileged
  - all guest state lives in the guest allowing the HV to be updated in place
Privileged mode constrained

- Close derivative of legacy privileged mode, but:
  - No access to diagnostic registers
  - No access to MMU control registers
  - No access to interrupt control registers
  - No access to I/O-MMU control registers
  - All replaced by Hypervisor API calls

- UltraSPARCness remains with minor changes
  - timer tick registers
  - softint registers etc.
  - trap-levels & global registers etc.
  - register window spill/fill
Legacy SPARC execution mode

 Existing sun4u chips
New SPARC Execution mode

- User Mode
- Privileged Mode
- Hyper-Privileged Mode

Interruption and error handling:
- User Mode to Privileged Mode
- Privileged Mode to Hyper-Privileged Mode
- Hyper-Privileged Mode to User Mode

System calls:
- Privileged Mode to System calls

Hyper-visor calls:
- Privileged Mode to Hyper-visor calls
- Hyper-visor calls to Privileged Mode

Retry:
- User Mode to Retry
- Privileged Mode to Retry
- Hyper-Privileged Mode to Retry
New SPARC Execution mode

- Privileged Mode
  - System calls
  - Hyper-Privileged Mode
  - Interrupts & errors
  - Retry

- User Mode
  - System calls

- Virtual Machine Environment
Devices / Interrupts

- DOM0 driver domains
- Allocating PCI-e nexus
- Virtual network and block devices
- Interrupt handling
Xen I/O Architecture

- Xen IO-Spaces delegate guest OSes protected access to specified h/w devices
  - Virtual PCI configuration space
  - Virtual interrupts

- Devices are virtualised and exported to other VMs via Device Channels
  - Safe asynchronous shared memory transport
  - ‘Backend’ drivers export to ‘frontend’ drivers
  - Net: use normal bridging, routing, iptables
  - Block: export any blk dev e.g. sda4,loop0,vg3
Sun4v direct I/O model

For FreeBSD existing drivers continue to work
Virtual Memory x86

Architected Page Tables
- Difficult to abstract – pages are stateful can't allow guest to update directly to prevent guest from mapping other guest's memory
  - (L3 vs. L2 etc. page tables)
  - Other global resources that can't be manipulated directly (GDT, LDT, etc.)
- Xen directly exposes page tables to the guest
  - Upside – relatively few changes to MD VM
  - Downside – large amounts of state required for tracking type of each page, exposing super-pages is more difficult, batching updates requires writable page tables which frequently don't work outside of Linux
Virtual Memory x86 II

Page table updates are all made via hyper-calls

- Setting cr3
- Writes to page directories and page tables
- Page table updates can be batched by means of "writable page tables", but their use precludes the use of linear page tables.
Virtual Memory Sparc

Software loaded TLB
- Sparc v8 and v9 relied on a TSB (translation storage buffer) as a direct mapped secondary TLB
  - Benefits of TSB over page tables:
    - allows for arbitrary page sizes
    - Single memory access for lookup
    - Lookups can be done in parallel for set associative TSBs
    - Flat structure avoids typing issues
  - Happily this also makes for a good interface with the hypervisor
  - Guest now registers TSB with the hypervisor on context context switch – hypervisor services TLB misses from the TSB
Live VM Relocation

✔ Why is VM relocation useful?
  ● Managing a pool of VMs running on a cluster
  ● Taking nodes down for maintenance
  ● Load balancing VMs across the cluster

✔ Why is it a challenge?
  ● VMs have lots of state
  ● Some VMs will have soft real-time requirements
    ● E.g. web servers, databases, game servers
  ● Can only commit limited resources to migration
## VM Relocation Strategy

<table>
<thead>
<tr>
<th>Stage 0: Pre-Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active VM on Host A</td>
</tr>
<tr>
<td>Alternate physical host may be preselected for migration</td>
</tr>
<tr>
<td>Block devices mirrored and free resources maintained</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 1: Reservation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize a container on the target host</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 2: Iterative Pre-copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable shadow paging</td>
</tr>
<tr>
<td>Copy dirty pages in successive rounds.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 3: Stop and copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspend VM on host A</td>
</tr>
<tr>
<td>Generate ARP to redirect traffic to Host B</td>
</tr>
<tr>
<td>Synchronize all remaining VM state to Host B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 4: Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM state on Host A is released</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage 5: Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM starts on Host B</td>
</tr>
<tr>
<td>Connects to local devices</td>
</tr>
<tr>
<td>Resumes normal operation</td>
</tr>
</tbody>
</table>
Writeable Working Set

Tracking the Writeable Working Set of SPEC CINT2000

Number of pages

Elapsed time (secs)
Xen and the BSDs

- NetBSD had full dom0 support for Xen 2 – full support for Xen3 a work in progress
- OpenBSD has seen no effort on Xen support to date – but Chris Jones has proposed it as an SoC project
- FreeBSD 7.0 has 3.x support for unprivileged guests and some extra “driver domain” functionality
- Some work has been done on dom0 – but has been neglected for several months – an SoC project is being pushed
Xen and FreeBSD

- Large amounts of work required to integrate the Xen toolset into the FreeBSD environment and make Xen usable for the average user.
- Xen appears to have no thought given to incremental versioning – FreeBSD will likely support point versions.
- Less compelling now that VMware Server is free and VMI is available.
  - If performance difference is less than 10%, VMware's polish and ease of use wins out.
Sun4v and FreeBSD

- Sun's hypervisor has a thoroughly documented API and an established versioning interface
- The challenge is more general and lies largely in FreeBSD's scaling bottlenecks and the lack of a maintainer for sparc64
VMI and FreeBSD

- Not as Linux-centric
- FreeBSD will ultimately seek a unified interface for Xen and VMI
- Objective of VMI is to have a non-disruptive P2V by having the same binary support both native and virtual
What's next?

- **Xen**
  - DomU support (unprivileged guest)
    - Complete balloon driver
    - Make sleep work prior to interrupts being enabled for xenbus
  - Dom0 support (initial domain)
    - Stabilization
    - Drivers to support domU (netback, blkback)
- **VMI**
  - Need further investigation
- **Sun4v**
  - Dom0 support
    - Pmap stabilization
  - DomU support
    - Virtual drivers