Topics Covered – My Cheat Sheet

- Virtualization
  - Review
  - What is virtualization
  - Definition of classical virtualization
  - Trap-and-Emulate
  - Memory Management

- x86 Virtualization
  - What are the challenges
    - Memory Tricks
  - What are the solutions
    - Binary Translation

- Approaches to Server Virtualization
  - Full Virtualization
  - Paravirtualization OS Assisted virtualization
  - Hardware-assisted virtualization
  - Charts

- Memory Management
  - Memory Tax
  - Chart
  - Ballooning
  - Content based Page Sharing
Overview

- Virtualization
- x86 Virtualization
- Approaches to Server Virtualization
- Memory Resource Management Techniques
What is Virtualization?

- Virtualization allows one computer to do the job of multiple computers, by sharing the resources of a single hardware across multiple environments.
VMWare Product Suite

- **Desktop** – runs in a host OS
  - VMWare Workstation (1999) – runs on PC
  - VMWare Fusion – runs on Mac OS X
  - VMWare Player – run, but not create images

- **Server**
  - VMWare Server (GSX Server) – hosted on Linux or Windows
  - VMWare ESX (ESX Server) – no host OS
  - VMWare ESXi (ESX 3i) – freeware (July 2008)
Virtual Machine

- abstracted isolated Operating System

Virtual Machine Monitor (VMM)

- capable of virtualizing all hardware resources, processors, memory, storage, and peripherals
- aka Hypervisor
Popek & Goldberg: Virtualization Criteria


Properties of Classical Virtualization

1. **Equivalence = Fidelity**
   - Program running under a VMM should exhibit a behavior identical to that of running on the equivalent machine

2. **Efficiency = Performance**
   - A statistically dominant fraction of machine instructions may be executed without VMM intervention

3. **Resource Control = Safety**
   - VMM is in full control of virtualized resources
De-privileging

- VMM emulates the effect on system/hardware resources of privileged instructions whose execution traps into the VMM
- aka trap-and-emulate
- Typically achieved by running GuestOS at a lower hardware priority level than the VMM
- Problematic on some architectures where privileged instructions do not trap when executed at deprivileged priority
Strategies: Memory Virtualization

Primary/Shadow structures
- Isolation/protection of Guest OS address spaces
- Avoid the two levels of translation on every access

Memory traces
- Efficient MM address translation
Popek & Goldberg: Classically Virtualizable

- According to Popek and Goldberg,
  
  "an architecture is virtualizable if the set of sensitive instructions is a subset of the set of privileged instructions."

- Is x86 Virtualizable?
  - No
Overview

- Virtualization
- x86 Virtualization
- Approaches to Server Virtualization
- Memory Resource Management Techniques
Challenges to x86 Virtualization (1)

- Lack of trap when privileged instructions run at user-level
  - **Classic Example**: `popf` instruction
    - Same instruction behaves differently depending on execution mode
    - User Mode: changes ALU flags
    - Kernel Mode: changes ALU and system flags
    - Does not generate a trap in user mode
Challenges to x86 Virtualization (2)

- Visibility of privileged state
  - Sensitive register instructions: read or change sensitive registers and/or memory locations such as a clock register or interrupt registers:
  - Protection system instructions: reference the storage protection system, memory or address relocation system:
Binary Translation

Characteristics

- **Binary** – input is machine-level code
- **Dynamic** – occurs at runtime
- **On demand** – code translated when needed for execution
- **System level** – makes no assumption about guest code
- **Subsetting** – translates from full instruction set to safe subset
- **Adaptive** – adjust code based on guest behavior to achieve efficiency
Binary Translation

TC: translation cache
TU: translation unit (usually a basic block)
CCF: compiled code fragment

Hash Table

Translation Cache

Guest Code

PC → [x] → TU → Binary Translator → CCF → [y] → execute

Running time

% translation

<table>
<thead>
<tr>
<th>Few cache hits</th>
<th>Working set captured</th>
</tr>
</thead>
</table>

continuation

Continued...
Eliminating faults/traps

- **Process**
  - **Privileged instructions** – eliminated by simple binary translation (BT)
  - **Non-privileged instructions** – eliminated by adaptive BT
    - (a) detect a CCF containing an instruction that trap frequently
    - (b) generate a new translation of the CCF to avoid the trap (perhaps inserting a call-out to an interpreter), and patch the original translation to execute the new translation
Binary Translation - Performance Advantages

- Avoid privilege instruction traps
  - Pentium privileged instruction (rdtsc) Trap-and-emulate: 2030 cycles
    - Callout-and-emulate: 1254 cycles
    - BT emulation: 216 cycles (but TSC value is stale)
Overview

- Virtualization
- x86 Virtualization
- Approaches to Server Virtualization
- Memory Resource Management Techniques
Approaches to Server Virtualization

- **1st Generation: Full virtualization (Binary translation)**
  - Software Based
  - VMware and Microsoft

- **2nd Generation: Paravirtualization**
  - Cooperative virtualization
  - Modified guest
  - VMware, Xen

- **3rd Generation: Silicon-based (Hardware-assisted) virtualization**
  - Unmodified guest
  - VMware and Xen on virtualization-aware hardware platforms
1st Generation: Full Virtualization

- Ring 3: User Apps
- Ring 2: 
- Ring 1: Guest OS
- Ring 0: VMM

Direct Execution of User Requests
Binary Translation of OS Requests
Full Virtualization - Drawbacks

• Hardware emulation comes with a performance price

• In traditional x86 architectures, OS kernels expect to run privileged code in Ring 0
  – However, because Ring 0 is controlled by the host OS, VMs are forced to execute at Ring 1/3, which requires the VMM to trap and emulate instructions

• Due to these performance limitations, paravirtualization and hardware-assisted virtualization were developed
2\textsuperscript{nd} Generation: Paravirtualization

- Ring 3: User Apps
- Ring 2
- Ring 1
- Ring 0: Paravirtualized Guest OS

Virtualization Layer

Host Computer System Hardware

Direct Execution of User Requests

‘Hypercalls’ to the Virtualization Layer replace Non-virtualizable OS Instructions
Paravirtualization Challenges

- Guest OS and hypervisor tightly coupled
  - Relies on separate kernel for native and in virtual machine
  - Tight coupling inhibits compatibility
  - Changes to the guest OS are invasive
  - Inhibits maintainability and supportability
  - Guest kernel must be recompiled when hypervisor is updated
Hardware Support for Virtualization

- **Non-root Mode Privilege Levels**
  - Ring 3: User Apps
  - Ring 2
  - Ring 1
  - Ring 0: Guest OS

- **Root Mode Privilege Levels**
  - VMM

- **Direct Execution of User Requests**
- **OS Requests Trap to VMM without Binary Translation or Paravirtualization**

Host Computer System Hardware
Software vs Hardware

- Hardware extensions allow classical virtualization on the x86.
- The overhead comes with exits – it no exits, then native speed
- Hardware Advantages:
  - Code density is preserved – no translation
  - Precise exceptions – BT performs extra work to recover guest state for faults and interrupts in non-IDENT code
  - System calls run without VMM intervention
- Software Advantages:
  - Trap elimination – replaced with callouts which are usually faster
  - Emulation speed – callouts provide emulation routine whereas hardware must fetch and decode the trapping instruction, then emulate
  - Callout avoidance: BT can avoid a lot of callouts by using in-TC emulation
## Summary

<table>
<thead>
<tr>
<th></th>
<th>Binary Translation</th>
<th>Hardware Assist</th>
<th>Para-virtualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatibility</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Poor</td>
</tr>
<tr>
<td>Performance</td>
<td>Good</td>
<td>Average</td>
<td>Excellent</td>
</tr>
<tr>
<td>VMM sophistication</td>
<td>High</td>
<td>Average</td>
<td>Average</td>
</tr>
</tbody>
</table>
Overview

- Virtualization
- x86 Virtualization
- Approaches to Server Virtualization
- Memory Resource Management Techniques
Memory resource management

- VMM (meta-level) memory management
  - **Must identify both VM and pages within VM to replace**
  - **VMM replacement decisions may have unintended interactions with GuestOS page replacement policy**
  - **Worst-case scenario: double paging**

- Strategies
  - **Eliminating duplicate pages – even identical pages across different GuestOSs.**
    - VMM has sufficient perspective
    - Clear savings when running numerous copies of same GuestOS
  - “ballooning” –
    - add memory demands on GuestOS so that the GuestOS decides which pages to replace
    - Also used in Xen
  - **Allocation algorithm**
    - Balances memory utilization vs. performance isolation guarantees
    - “taxes” idle memory
Content-based page sharing

- A hash table contains entries for shared pages already marked "copy-on-write"
- A key for a candidate page is generated from a hash value of the page's contents
- A full comparison is made between the candidate page and a page with a matching key value
- Pages that match are shared – the page table entries for their VMMs point to the same machine page
- If no match is found, a "hint" frame is added to the hash table for possible future matches
- Writing to a shared page causes a page fault which causes a separate copy to be created for the writing GuestOS
Page sharing performance

- Identical Linux systems running same benchmark
- “best case” scenario
- Large fraction (67%) of memory sharable
- Considerable amount and percent of memory reclaimed
- Aggregate system throughput essentially unaffected
Ballooning: Inflate

- Inflating the balloon
  - Balloon requests additional “pinned” pages from GuestOS
  - Inflating the balloon causes GuestOS to select pages to be replaced using GuestOS page replacement policy
  - Balloon informs VMM of which physical page frames it has been allocated
  - VMM frees the machine page frames corresponding to the physical page frames allocated to the balloon (thus freeing machine memory to allocate to other GuestOSs)
Ballooning: Deflate

- Deflating the balloon
  - VMM reclaims machine page frames
  - VMM communicates to balloon
  - Balloon unpins/frees physical page frames corresponding to new machine page frames
  - GuestOS uses its page replacement policy to page in needed pages
Measuring Cross-VM memory usage

- Each GuestOS is given a number of shares, $S$, against the total available machine memory.
- The shares-per-page represents the “price” that a GuestOS is willing to pay for a page of memory.
- The price is determined as follows:

$$\rho = \frac{S}{P \cdot (f + k \cdot (1 - f))}$$

- The idle page cost is $k = 1/(1-\tau)$ where $0 \leq \tau < 1$ is the “tax rate” that defaults to 0.75
- The fractional usage, $f$, is determined by sampling (what fraction of 100 randomly selected pages are accesses in each 30 second period) and smoothing (using three different weights)
Memory tax experiment

- Initially, VM1 and VM2 converge to same memory allocation with \( \tau = 0 \) (no idle memory tax) despite greater need for memory by VM2
- When idle memory tax applied at default level (75%), VM1 relinquishes memory to VM2 which improves performance of VM2 by over 30%
References and Sources

- A Comparison of Software and Hardware Techniques for x86 Virtualization
  Keith Adams & Ole Agesen
- A Comparison of Software and Hardware Techniques for x86 Virtualization
  Mike Marty
- A Comparison of Software and Hardware Techniques for x86 Virtualization
  Jordan and Justin Ehrlich
- A Survey on Virtualization Technologies  Susanta K Nanda
- Disco: Running Commodity Operating Systems on Scalable Multiprocessors
  Divya Parekh
- Hardware Support for Efficient Virtualization  John Fisher-Ogden
- Memory Resource Management in VMware ESX Server  Carl A. Waldspurger
- Memory Resource Management in VMware ESX Server  VMware
- Resource Management  Carl A. Waldspurger
- Understanding Intel® Virtualization Technology (VT)  Dr. Michael L. Collard
References and Sources

- Understanding *Memory Resource Management in VMware ESX™ Server* VMware
- Understanding Full Virtualization, Paravirtualization and Hardware Assist VMware
- Virtualization *Intel and Argentina Software Pathfinding and Innovation*
- VMWare and CPU Virtualization Technology *Jack Lo*
- VMWare Virtualization of Oracle and Java *Scott Drummonds & Tim Harris*
- Lecture on Vmware *Dr. Dennis Kafura*
- What is Virtualization *Scott Devine*