15-712: Advanced Operating Systems & Distributed Systems

Virtual Machines II

Prof. Phillip Gibbons

Spring 2020, Lecture 20
“Memory Resource Management in VMware ESX Server”

Carl A. Waldspurger 2002

- Carl Waldspurger (VMware)
  - DEC SRC, VMware, independent consultant
  - Program Chair for USENIX ATC, FAST, VEE
  - Married to professional rock concert photographer

© Paige Parsons
This paper introduced elegant and effective techniques of hypervisor memory management. Memory ballooning allows the hypervisor to reclaim memory from a virtual machine in accordance with the unmodified guest’s operating system policies. Transparent page sharing supports efficient memory use with small overhead. The combination of active memory estimation, idle memory tax, and proportional fair sharing, along with admission-controlled memory reservation, provides the basis for service level agreements and reasoned overcommitment. This paper has been highly influential; many of its techniques have been adopted by widely-used hypervisors.
Architecture of DISCO

Virtual machine monitor (VMM), aka Hypervisor
Advantages of VMM Approach

• No redesign of system software to new hardware
• Can keep running conventional OS & legacy apps; Can stage new apps
• Can have multiple OS, including application-specialized OS
• Fine-grain resource sharing: e.g., move memory between VMs
• Dynamic load balance of virtual processors on physical machines
• Simpler VM becomes unit of scalability & of fault containment
• Can hide NUMA, via dynamic page migration/replication

Server Consolidation improves server utilization
Disco Required Changes to IRIX

- Kernel code & data relocated to TLB-controlled address space
- Special device drivers that use Disco’s call interface
- Load/store on special addresses for fast interrupt enable/disable
- Monitor calls for requesting a zeroed page & for inactive page
- Changed mbuf freelist data structure to avoid page writes
- Remap function to enable transparent sharing over NFS

Goal: VMM with NO changes to the OS
VMware ESX Server: Key New Ideas

• **Ballooning**
  – Reclaims the pages considered least valuable by an OS

• **Content-Based Page Sharing & Hot I/O Page Remapping**
  – Eliminate redundancy & reduce copying overheads via transparent page remapping

• **Idle Memory Tax**
  – Efficient memory utilization with performance isolation guarantees
Disco Memory Virtualization

ESX uses same level-of-indirection approach
Ballooning

• What problem is it trying to solve?
  – Ill-informed page reclamation by VMM, incl. double paging
    (over-provisioning implies VMM needs to reclaim memory)

• How does it solve it?
  – VMM-controlled module running in guest OS
  – Allocates pages pinned to *physical* memory, sends PPNs to VMM
  – VMM reclaims machine pages for these PPNs, sets OS pmap entry to fault-on-access

• How much overhead?
  – 1.4-4.4% on *dbench* w/40 clients
Content-Based Page Sharing

• What problem is it trying to solve?
  – Disco page sharing required changes to OS & to application programming interfaces

• How does it solve it?
  – Check random pages for matches
  – If find match:
    share & mark as COW
  – If no match:
    save as hint (no COW)
  – Check page before swap

• How does it perform?
Content-Based Page Sharing
Performance

- Identical Linux VMs running SPEC95 benchmarks
- Production workloads at three companies

### Table: Guest Types and Memory Usage

<table>
<thead>
<tr>
<th>Guest Types</th>
<th>Total MB</th>
<th>Shared MB</th>
<th>%</th>
<th>Reclaimed MB</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>A WinNT</td>
<td>2048</td>
<td>880</td>
<td>42.9</td>
<td>673</td>
<td>32.9</td>
</tr>
<tr>
<td>B Linux</td>
<td>1846</td>
<td>539</td>
<td>29.2</td>
<td>345</td>
<td>18.7</td>
</tr>
<tr>
<td>C Linux</td>
<td>1658</td>
<td>165</td>
<td>10.0</td>
<td>120</td>
<td>7.2</td>
</tr>
</tbody>
</table>
Discussion: Summary Question #1

• **State the 3 most important things the paper says.** These could be some combination of their motivations, observations, interesting parts of the design, or clever parts of their implementation.
Idle Memory Tax

• What problem is it trying to solve?
  – Hoarding of idle pages by clients with many shares

• How does it solve it?
  – Makes cost(idle page) >> cost(active page)
  – Shares-per-page ratio is $\frac{S}{P(f+k(1-f))}$ for $S$ shares, $P$ pages, $f$ active fraction, $k = \frac{1}{(1 - \tau)}$ for a given tax rate $0 \leq \tau < 1$
  – Reclaim pages from client with lowest ratio
  – Default tax rate $\tau = 75$

• How does it determine $f$?
  – Active memory sampling
Active Memory Sampling

• Each sampling period, select $n$ physical pages at random from VM
  – Invalidate their TLB & virtualized MMU state mappings
  – On access, increment counter $t$ & reestablish mappings

• Estimate $f = \max(\text{slow}, \text{fast}, \text{fastest})$ moving averages of $t/n$
Idle Memory Tax Performance

360 MB available memory
• **Describe the paper's single most glaring deficiency.** Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.
Dynamic Reallocation of Memory

- High: no reclamation
- Soft: use ballooning
- Hard: use paging
- Low: paging+blocking

5 VMs running real apps, memory overcommitted by 60%
Hot I/O Page Remapping

• **What problem is it trying to solve?**
  – DMA for I/O transfers can only write to “low” memory
  – Current systems copy through a bounce buffer (high overhead)

• **How does it solve it?**
  – Count # such copies for each PPN
  – If count(PPN) > threshold: Remap page to low memory

• **How successful is it?**
  – Up to several orders of magnitude reduction in copies
VMware ESX Server: Key New Ideas

• **Ballooning**
  – Reclaims the pages considered least valuable by an OS

• **Content-Based Page Sharing & Hot I/O Page Remapping**
  – Eliminate redundancy & reduce copying overheads via transparent page remapping

• **Idle Memory Tax**
  – Efficient memory utilization w/ performance isolation guarantees
“Dynamo: Amazon’s Highly Available Key-value Store”

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, Werner Vogels 2007

Further Reading:

“Chord: A Scalable Peer-to-peer Lookup Service for Internet Application”

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan 2001