Memory Resource Management in VMWare ESX Server
Contents

• Background
• Motivation and Overview
• Memory Virtualization
• Reclamation Mechanism
• Sharing Memory
• Allocation Policies
• Conclusions
VMWare Infrastructure

VMWare Infrastructure

Migration with Vmotion

[VMWare DRS (Distributed Resource Scheduler)]

[VMWare HA (High Availability)]
Motivation and Overview

• Motivation
  – Higher memory utilization
    • With memory overcommitment, ESX ensures that host memory is consumed by active guest memory as much as possible
  – Higher consolidation ratio
    • With memory overcommitment, each virtual machine has a smaller footprint in host memory usage,

• Overview
  – In order to effectively support memory overcommitment,
    • Ballooning technique
    • Content-based page sharing
    • Idle memory tax
    • Hot IO/O page remapping
Memory Virtualization Basic

• Terminology
  – Virtual address = Guest virtual address
  – Physical address = Guest physical address
  – Machine address = Host physical address

• Shadow page table
  – Converts VA to MA using trap
Reclamation Mechanisms

• **Over-commitment of memory**
  – VM’s host memory usage <= VM’s guest memory size + VM’s overhead memory

  ![Memory overcommitment in ESX](image)

• **Page Replacement Issues**
  – Meta-level page replacement
    • Hypervisor choose pages for replacement
    • Conflicts with guest OS replacement policy
  – Double paging problem
    • Meta-level policy is able to select the same page that the native guest os policy
Memory ballooning

• Ballooning
  – Guest OS has best information for page reclamation
  – Inflate: allocate pinned PPNs; backing MPNs reclaimed
  – Ballooning overhead: 1.4% ~ 4.4%
  – Limitation: Balloon driver must be loaded into guest OS
    unable to reclaim memory quickly

[Memory ballooning]

[Balloon Performance]
Black bar: w/o ballooning
Gray bar: with ballooning

Throughput (MB/sec)

VM Size (MB)
128 160 192 224 256

[Memory ballooning]
Sharing Memory

- **Motivation**
  - Multiple VMs running same OS, apps
  - Collapse redundant copies of code, data, zeros

- **Transparent page sharing**
  - Map multiple PPNs to single MPN copy-on-write
  - Pioneered by Disco [Bugnion 97], but required guest OS hooks

- **New twist: content-based sharing**
  - General-purpose, no guest OS changes
  - Background activity saves memory over time
Content-Based Page Sharing

- **Page Sharing: Scan Candidate PPN**
  - Hash value summarizes a page’s content
  - Only shared page is marked COW
  - Unshared page is not marked COW, but instead tagged as a special hint entry
Content-Based Page Sharing

- **Page sharing : Successful match**
  - Uses 16-bit reference count for each shared page (Disco : backmap)
  - Fast, high-quality hash function (64-bit hash value -> collision probability < 0.01%)
  - Scans guest pages randomly
Page Sharing Performance

• **Best-case workload**
  – Identical Linux VMs (Red Hat Linux 7.2 with 40MB of "physical" memory)
  – SPEC95 benchmarks (runs during 30 minutes)
  – Lots of potential sharing

• **Metrics**
  – Shared PPNs -> 67%
  – Saved MPNs -> 60%
Page Sharing Performance

• Real-World Page Sharing

<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 10 WinNT</td>
<td>2048</td>
<td>880</td>
<td>42.9</td>
<td>673</td>
<td>32.9</td>
</tr>
<tr>
<td>B 9 Linux</td>
<td>1846</td>
<td>539</td>
<td>29.2</td>
<td>345</td>
<td>18.7</td>
</tr>
<tr>
<td>C 5 Linux</td>
<td>1658</td>
<td>165</td>
<td>10.0</td>
<td>120</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Workload A: 10 Windows NT VMs serving users at a Fortune 50 company, running database (Oracle, SQL Server), web (IIS, Websphere), development (Java, VB), and other applications.

Workload B: Nine Linux VMs serving a large user community for a nonprofit organization, executing a mix of web (Apache), mail (Majordomo, Postfix, POP/IMAP, MailArmor), and other servers.

Workload C: Five Linux VMs providing web proxy (Squid), mail (Postfix, RAV), and remote access (ssh) services to VMware employees.
Allocation Parameters

- **Min size**
  - Guaranteed, even when overcommitted
  - Enforced by admission control

- **Max size**
  - Amount of physical memory seen by guest OS
  - Allocation when undercommitted

- **Shares**
  - Specify relative importance
  - Proportional-share fairness
Allocation Policy

• **Traditional approach**
  - Optimize aggregate system-wide metric
  - Problem: no QoS guarantees, VM importance varies

• **Share-based allocation**
  - Revoke from VM with min shares-per-page ratio
  - Problem
    - Do not incorporate any information about active memory usage or working sets
    - Idle clients with many shares can hoard memory unproductively.

• **Desired behavior**
  - VM gets full share when actively using memory
  - VM may lose pages when working set shrinks
Reclaiming Idle Memory

• **Idle memory tax**
  - Charge a client more for an idle page
  - Tax rate (τ): the maximum fraction of idle pages that may be reclaimed from a client (0 ≤ τ < 1)
    • τ = 0: pure share-based isolation
    • τ ≈ 1: all of a client’s idle memory are reclaimed
    • ESX server default tax rate: 75%
  - Adjusted shares-per-page ratio ρ
    • f: active page ratio
    • idle page cost k = 1/(1 - τ)
    
    \[
    ρ = \frac{S}{P \cdot (f + k \cdot (1 - f))}
    \]
Measuring Idle Memory

- **Statistical sampling approach**
  - A small number $n$ of VM’s physical pages are selected randomly.
  - The next guest access to a sampled page reestablish these mappings, increasing a touched page count $t$.
  - At the end of sampling period, $f$ is estimated as $f = t/n$.
  - By default, ESX server samples 100 pages for each 30 second period.
Experimental Results

• Idle Memory Tax
  – VM1(gray) runs Windows, and remains idle after booting
  – VM2(black) executes a memory-intensive Linux workload
  – Before no tax: Gets same 179MB allocation
  – After tax(75%): VM2 gets more memory
Dynamic Reallocation

- **Reallocation events**
- **Enforcing target allocations**
  - Ballooning: common-case optimization
  - Swapping: dependable fallback, try sharing first
- **Reclamation states**
  - High background sharing (6% >= free memory of system memory)
  - Soft mostly balloon (<4%)
  - Hard mostly swap (<2%)
  - Low swap and block VMs above target (<1%)
Experimental Result

• **Result**
  – Only 1 GB was available for executing VMs
  – Use a total of 1472 MB
    • 160MB overhead memory
    • Overcommitted by more than 60%
  – (a) ESX Server allocation state transitions
  – (b) Aggregate allocation metrics summed over all five VMs
  – (c) Allocation metrics for MetaFrame Server VM
  – (d) Allocation metrics for SQL server VM

[Dynamic Reallocation]
I/O Page Remapping

• DMA from high memory
  – IA-32 PAE mode supports 36-bit addressing (up to 64 GB)
  – Many 32-bit I/O devices (low 4 GB only)
  – VM memory may be located anywhere

• Copy when necessary
  – Conventional approach
  – Use temporary DMA bounce buffer

• Dynamic page remapping
  – Keep copy statistics to identify hot pages
  – Transparently remap from high to low memory
Related Work

- **Disco**
  - insert an additional layer of software between OS and HW
  - FLASH microprocessor on ccNUMA
  - Multiple copies of commodity OSes across the layer
Conclusions

• **Key features**
  – Flexible dynamic partitioning
  – Efficient support for overcommitted workloads

• **Novel mechanisms**
  – Ballooning leverages guest OS algorithms
  – Content-based page sharing
  – Statistical working-set estimation

• **Integrated policies**
  – Proportional-sharing with idle memory tax
  – Dynamic reallocation