Memory Resource Management in VMware ESX Server

Carl Waldspurger

OSDI '02 Presentation

December 10, 2002

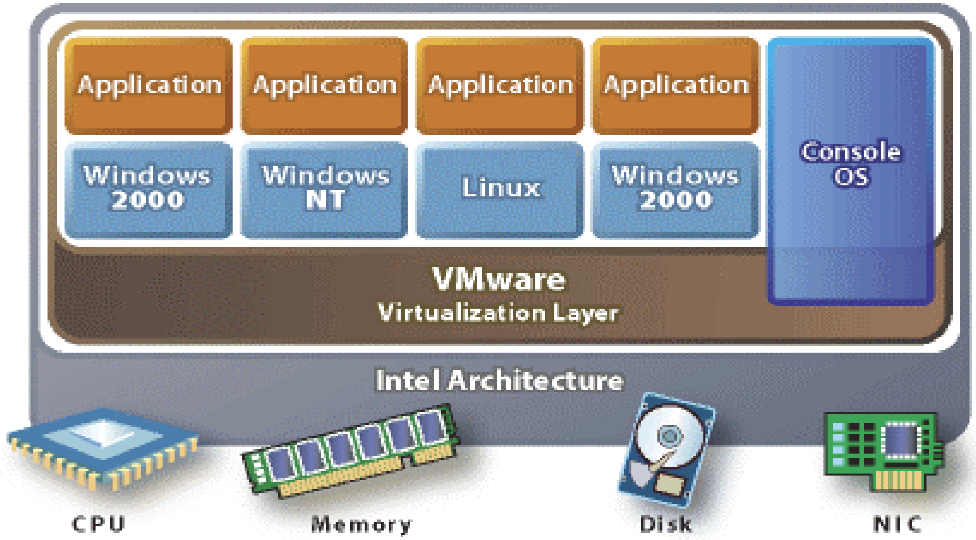
Overview

- Context
- Memory virtualization
- Reclamation
- Sharing
- Allocation policies
- Conclusions

Motivation

- Server consolidation
  - Many physical servers underutilized
  - Consolidate multiple workloads per machine
- Virtual machines
  - Illusion of dedicated physical machine
  - Encapsulate workload (OS + apps)
  - IBM VM/370 [Creasy '81], Disco [Bugnion '97], VMware [Sugerman '01]
- Resource management
  - Fairness, performance isolation
  - Efficient utilization

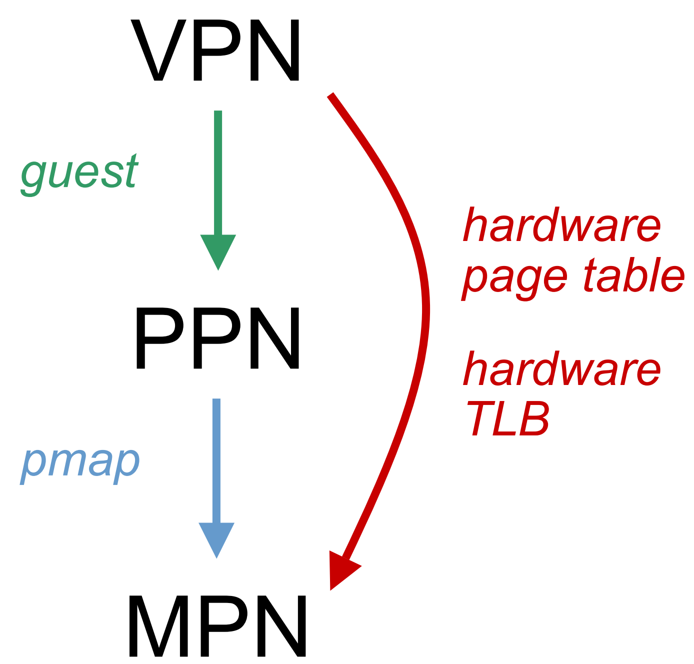
ESX Server



- Commercially-available product
- Thin kernel designed to run VMs
- Multiplex hardware resources
- High-performance I/O

Memory Virtualization

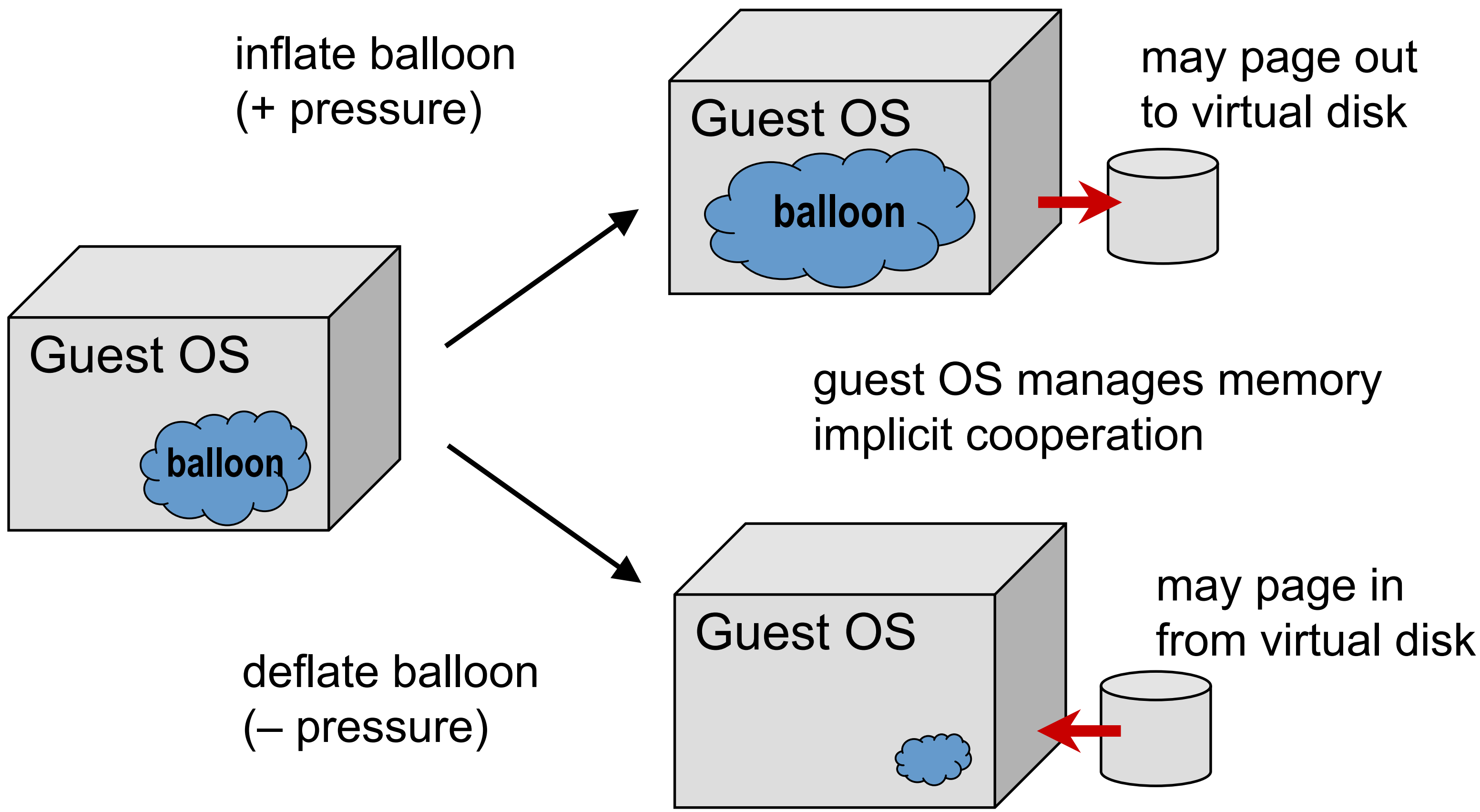
- Traditional VMM approach: extra level of indirection
- virtual → "physical"
  guest maps VPN to PPN
- "physical" → machine
  pmap maps PPN to MPN
- Ordinary memory refs:
  hardware maps VPN to MPN



Reclaiming Memory

- Traditional: add transparent swap layer
  - Requires meta-level page replacement decisions
  - Best data to guide decisions known only by guest OS
  - Guest and meta-level policies may clash
  - Example: “double paging” anomaly
- Alternative: implicit cooperation
  - Coax guest into doing page replacement
  - Avoid meta-level policy decisions

Ballooning


inflate balloon
(+ pressure)
Guest OS
balloon
Guest OS
balloon
may page out
to virtual disk
guest OS manages memory
implicit cooperation
deflate balloon
(– pressure)
Guest OS
may page in
from virtual disk

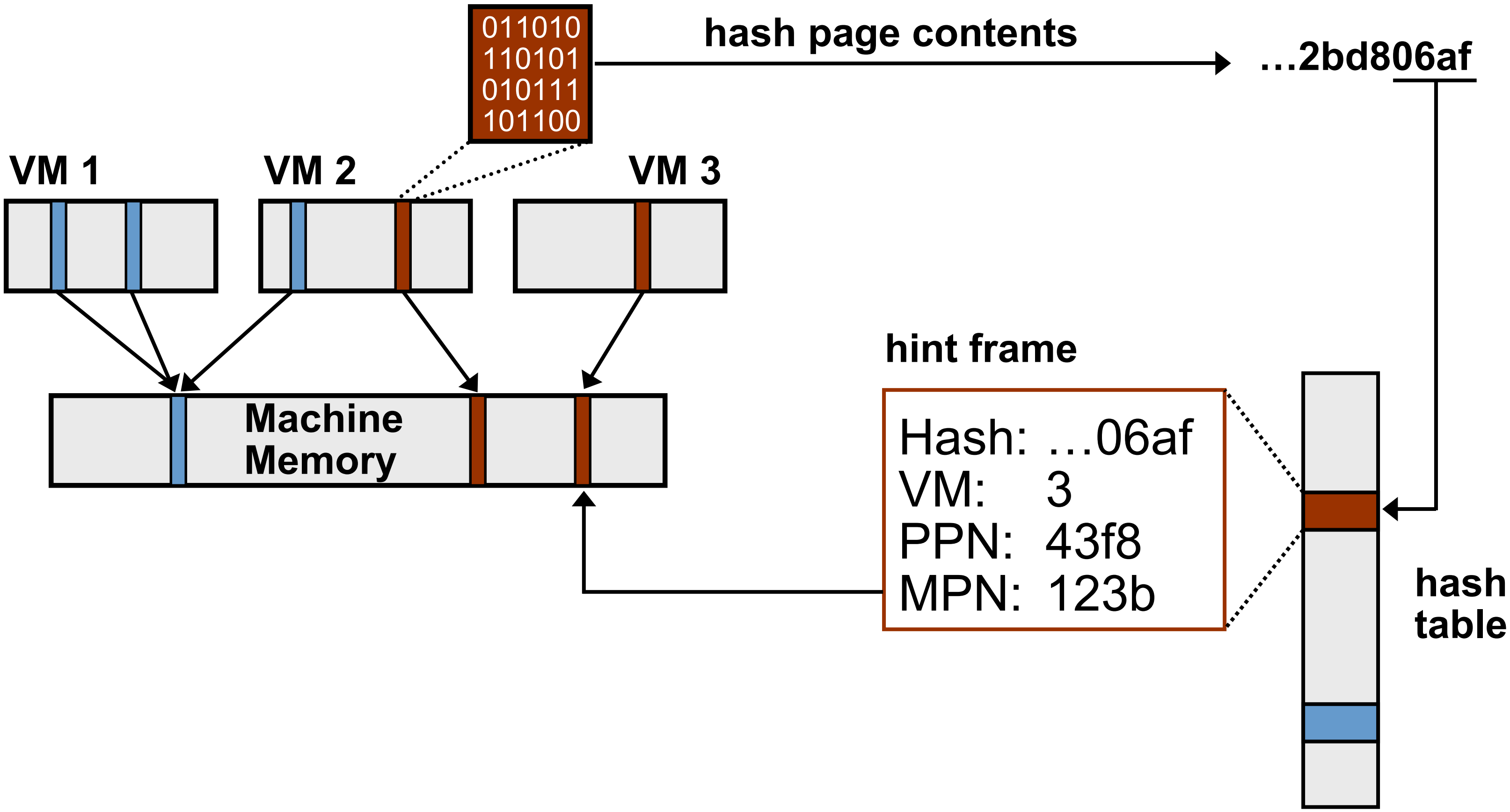
Ballooning Details

- Guest drivers
  - Inflate: allocate pinned PPNs; backing MPNs reclaimed
  - Use standard Windows/Linux/BSD kernel APIs
  - Related: Nemesis “self-paging” [Hand ’99], Collective [Sapuntzakis ’02]
- Performance benchmark
  - Linux VM, memory-intensive dbench workload
  - Compare 256 MB with balloon sizes 32 – 128 MB vs. static VMs
  - Overhead 1.4% – 4.4%
- Some limitations

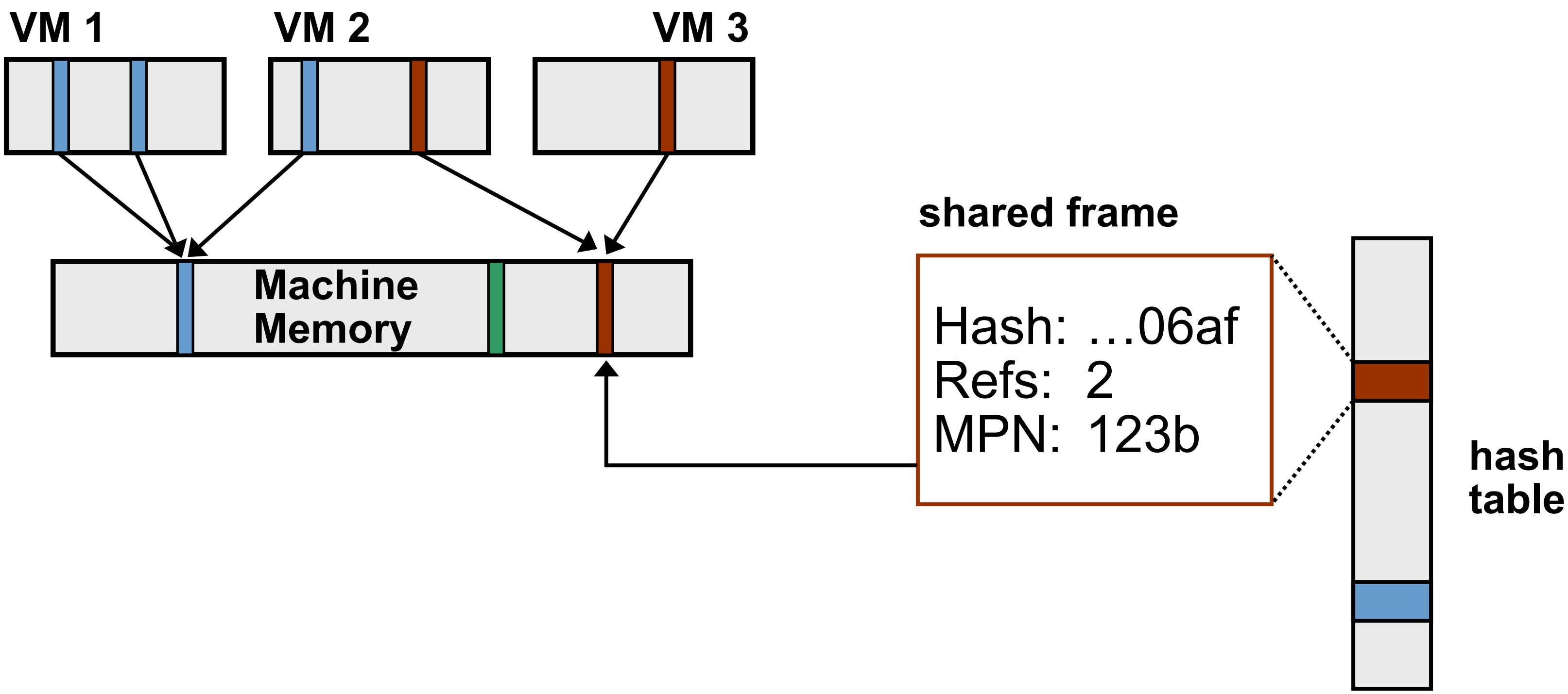
Sharing Memory

- 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

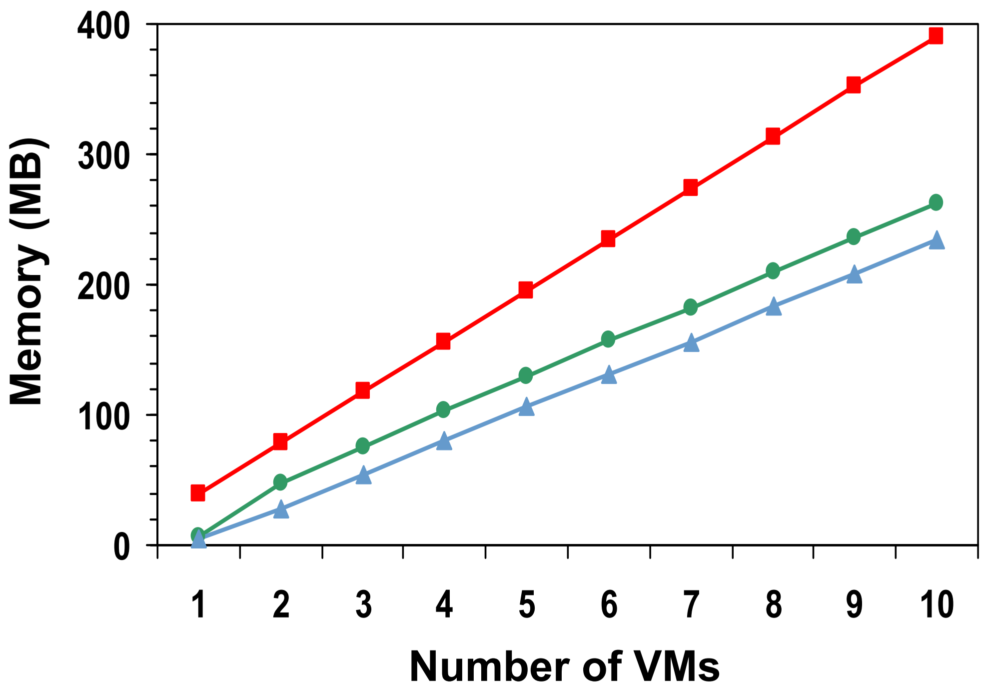
Page Sharing: Scan Candidate PPN



Page Sharing: Successful Match



Page Sharing Performance



- "Best-case" workload
  - Identical Linux VMs
  - SPEC95 benchmarks
  - Lots of potential sharing
- Metrics
  - Total guest PPNs
  - Shared PPNs → 67%
  - Saved MPNs → 60%
- Effective sharing
- Negligible overhead

Real-World Page Sharing

		Total	Saved	
Workload	Guest Types	MB	MB	%
Corporate IT	10 Windows	2048	673	32.9
Nonprofit Org	9 Linux	1846	345	18.7
VMware	5 Linux	1658	120	7.2

Corporate IT – database, web, development servers (Oracle, Websphere, IIS, Java, etc.)

Nonprofit Org – web, mail, anti-virus, other servers (Apache, Majordomo, MailArmor, etc.)

VMware – web proxy, mail, remote access (Squid, Postfix, RAV, ssh, etc.)

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

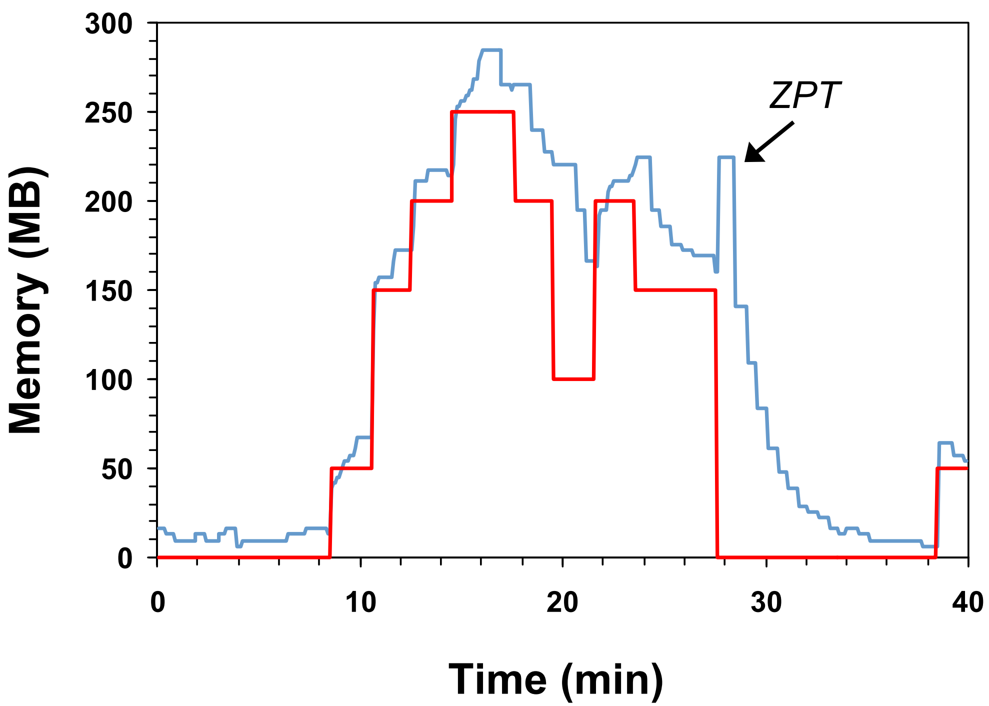
- Traditional approach
  - Optimize aggregate system-wide metric
  - Problem: no QoS guarantees, VM importance varies
- Pure share-based approach
  - Revoke from VM with min shares-per-page ratio [Waldspurger '95]
  - Problem: ignores usage, unproductive hoarding [Sullivan '00]
- Desired behavior
  - VM gets full share when actively using memory
  - VM may lose pages when working set shrinks

Reclaiming Idle Memory

- Tax on idle memory
  - Charge more for idle page than active page
  - Idle-adjusted shares-per-page ratio
- Tax rate
  - Explicit administrative parameter
  - 0% ≈ “plutocracy” … 100% ≈ “socialism”
- High default rate
  - Reclaim most idle memory
  - Some buffer against rapid working-set increases

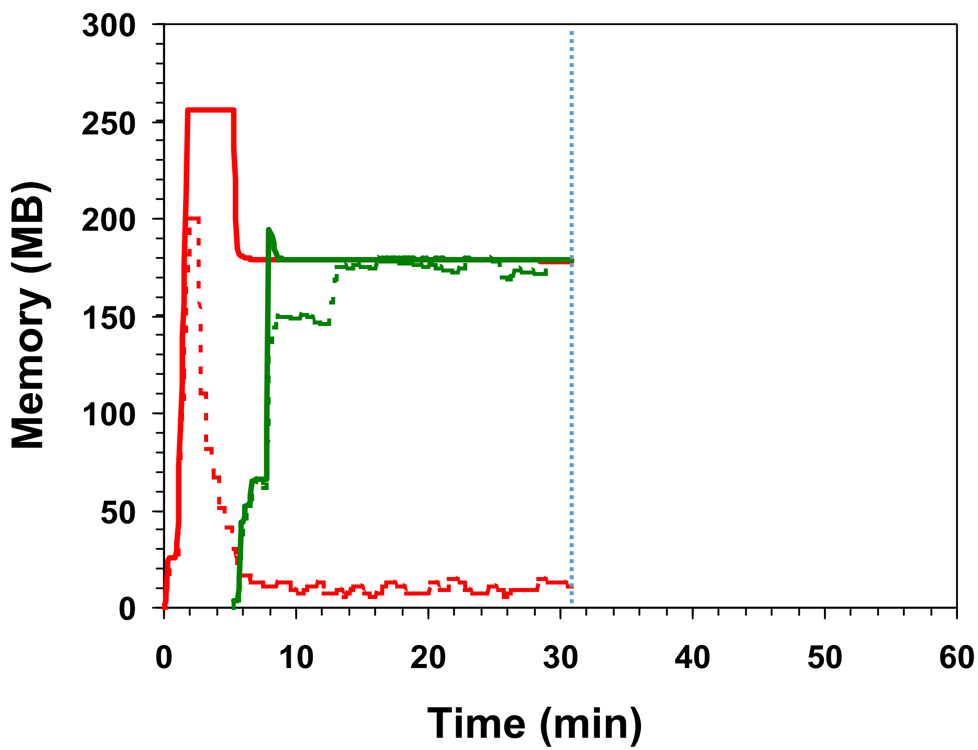
vmware™

Measuring Active Memory



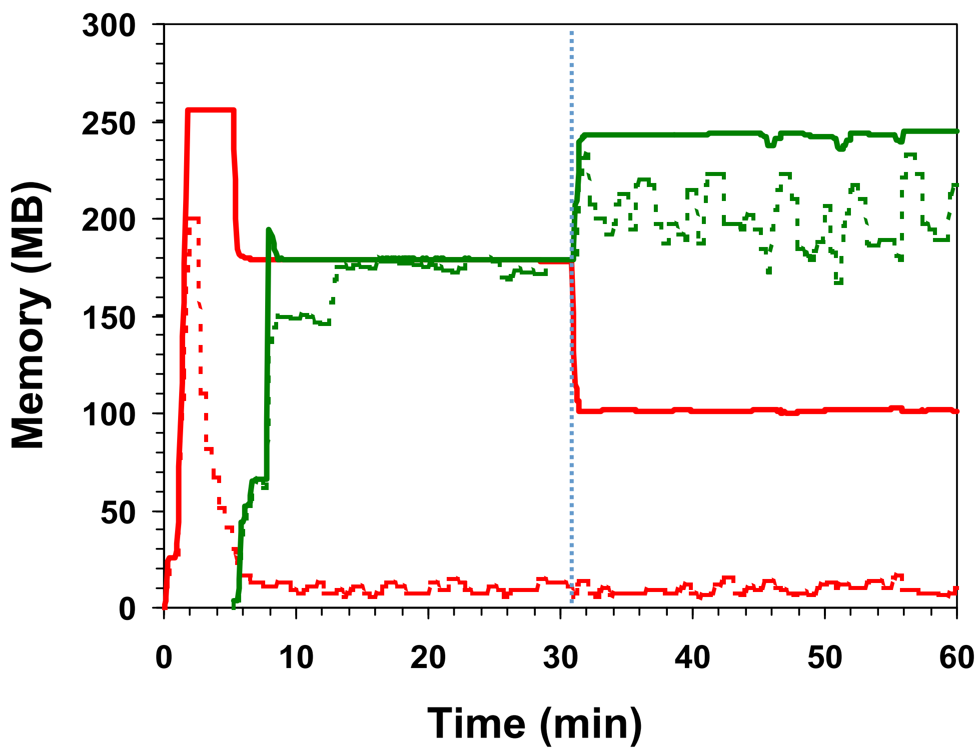
- Experiment
  - Single Windows VM
  - Memory "toucher" app
  - Active memory estimate
- Statistical sampling
  - Small random subset of pages
  - Software access bits [Joy '81]
  - Moving averages [Kim '01]
- Behavior
  - Rapid response to ↑ usage
  - Gradual response to ↓ usage
  - Windows "zero page thread"

Idle Memory Tax: 0%



- Experiment
  - 2 VMs, 256 MB, same shares
  - VM1: Windows boot+idle
  - VM2: Linux boot+dbench
  - Solid: usage, Dotted: active
- Change tax rate
- Before: no tax
  - VM1 idle, VM2 active
  - get same allocation

Idle Memory Tax: 75%



- Experiment
  - 2 VMs, 256 MB, same shares
  - VM1: Windows boot+idle
  - VM2: Linux boot+dbench
  - Solid: usage, Dotted: active
- Change tax rate
- After: high tax
  - Redistribute VM1 → VM2
  - VM1 reduced to min size
  - VM2 throughput improves 30%

Dynamic Reallocation

- Reallocation events
- Enforcing target allocations
  - Ballooning: common-case optimization
  - Swapping: dependable fallback, try sharing first
- Reclamation states
  - High – background sharing
  - Soft – mostly balloon
  - Hard – mostly swap
  - Low – swap and block VMs above target

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

Questions?

Extra Slides

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