Resource Management for Virtualized Systems

Carl Waldspurger
VMware R&D

Columbia E6998 Lecture
April 22, 2008

Virtualized Resource Management

Physical resources

- Actual “host” hardware
- Processors, memory, I/O devices, etc.

Virtual resources

- Virtual “guest” hardware abstractions
- Processors, memory, I/O devices, etc.

Resource management

- Map virtual resources onto physical resources
- Multiplex physical hardware across VMs
- Manage contention based on admin policies

Resource Management Goals

Performance isolation

- Prevent VMs from monopolizing resources
- Guarantee predictable service rates

Efficient utilization

- Exploit undercommitted resources
- Overcommit with graceful degradation

Easy administration

- Flexible dynamic partitioning
- Meet absolute service-level agreements
- Control relative importance of VMs

Talk Overview

- Resource controls
- Processor scheduling
- Memory management
- NUMA scheduling
- Distributed systems
- Summary

Resource Controls

Useful Features

- Express absolute service rates
- Express relative importance
- Grouping for isolation or sharing

Challenges

- Simple enough for novices
- Powerful enough for experts
- Physical resource consumption vs. application-level metrics
- Scaling from single host to server farm

VMware Basic Controls

Shares

- Specify relative importance
- Entitlement directly proportional to shares
- Abstract relative units, only ratios matters

Reservation

- Minimum guarantee, even when system overcommitted
- Concrete absolute units (MHz, MB)
- Admission control: sum of reservations ≤ capacity

Limit

- Upper bound on consumption, even when undercommitted
- Concrete absolute units (MHz, MB)

VMware Resource Pools

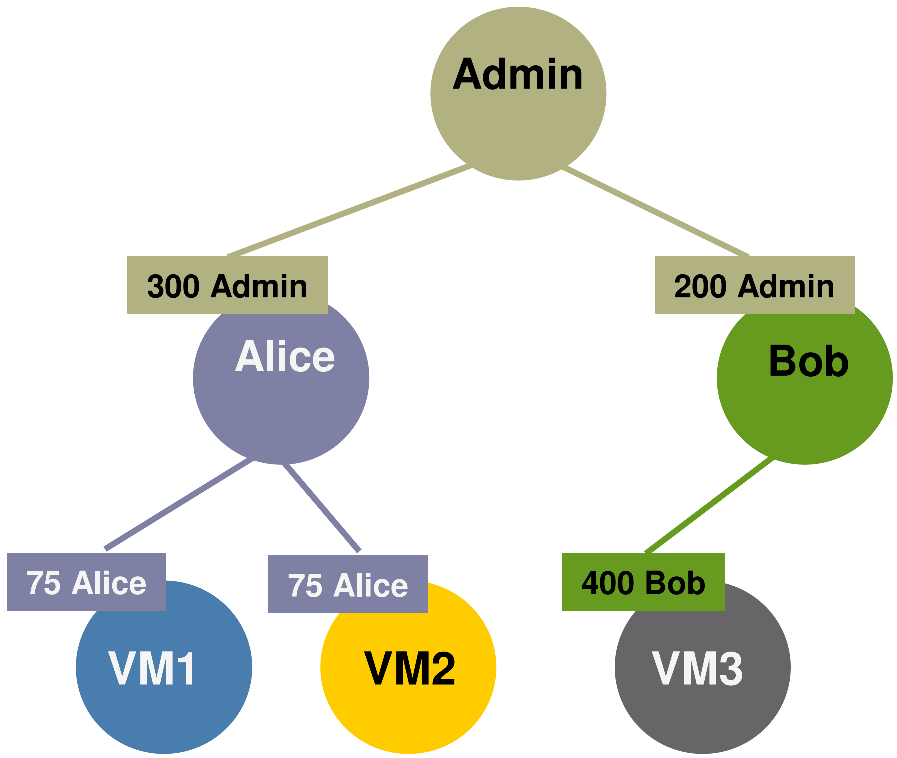
Motivation

- Allocate aggregate resources for sets of VMs
- Isolation between pools, sharing within pools
- Flexible hierarchical organization
- Access control and delegation

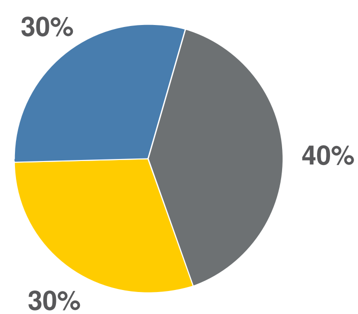
What is a resource pool?

- Named object with permissions
- Reservation, limit, and shares for each resource
- Parent pool, child pools, VMs

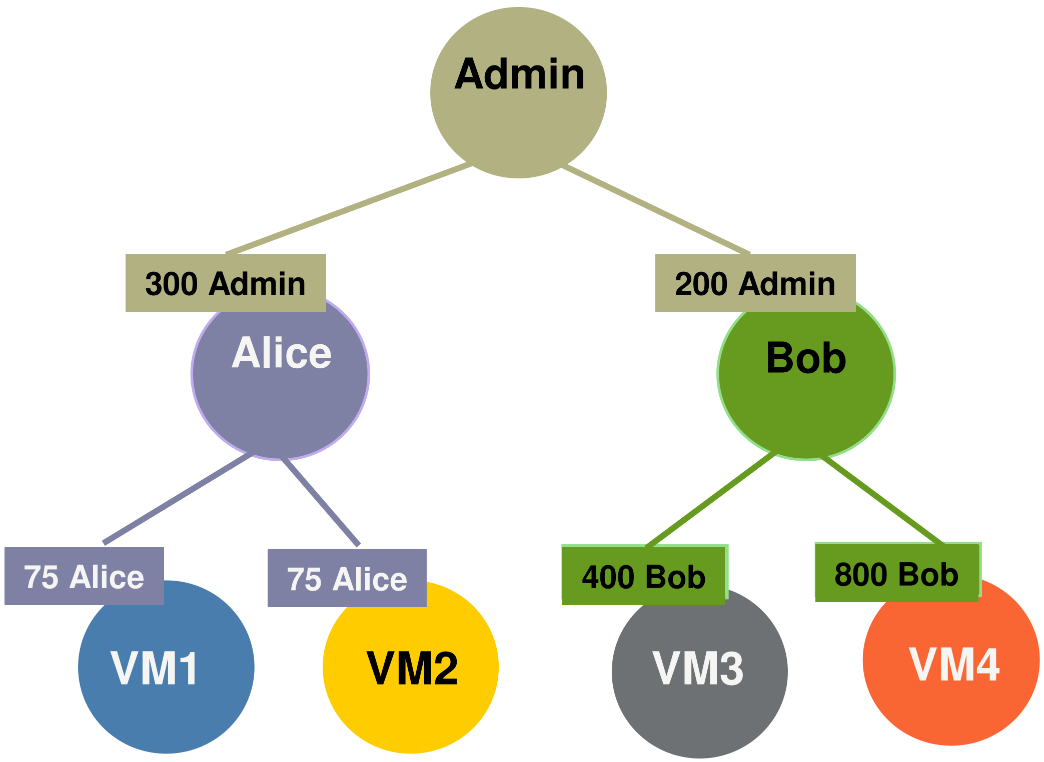
Resource Pools Example



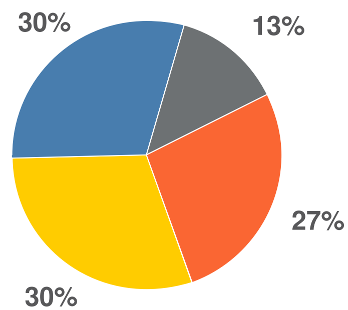
- Admin manages users
- Policy: Alice's share 50% more than Bob's
- Users manage own VMs
- Not shown: reservations, limits
- VM allocations:



Example: Bob Adds VM



- Same policy
- Pools isolate users
- Alice still gets 50% more than Bob
- VM allocations:



Resource Controls: Future Directions

Emerging DMTF standard

- Reservation, limit, “weight” + resource pools
- Model expressive enough for all existing virtualization systems
- Authors from VMware, Microsoft, IBM, HP, Sun, XenSource, etc.

Other controls?

- Priority scheduling
- Real-time latency guarantees
- I/O-specific controls

Application-level metrics

- Users think in terms of transaction rates, response times
- Labor-intensive, requires detailed domain/app-specific knowledge
- Can layer on top of basic physical resource controls

Talk Overview

- Resource controls
- Processor scheduling
- Memory management
- NUMA scheduling
- Distributed systems
- Summary

Processor Scheduling

Useful features

- Accurate rate-based control
- Support both UP and SMP VMs
- Exploit multi-core, multi-threaded CPUs
- Grouping mechanism

Challenges

- Efficient scheduling of SMP VMs
- VM load balancing, interrupt balancing
- Cores/threads may share cache, functional units
- Lack of control over micro-architectural fairness
- Proper accounting for interrupt-processing time

VMware Processor Scheduling

Scheduling algorithms

- Rate-based controls
- Hierarchical resource pools
- Inter-processor load balancing
- Accurate accounting

Multi-processor VM support

- Illusion of dedicated multi-processor
- Near-synchronous co-scheduling of VCPUs

Modern processor support

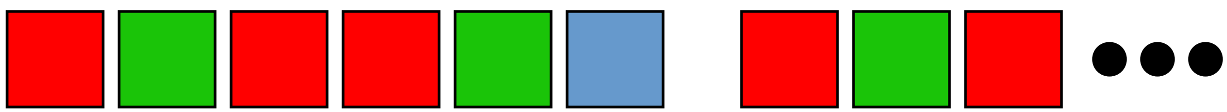
- Multi-core sockets with shared caches
- Simultaneous multi-threading (SMT)

Proportional-Share Scheduling

Simplified virtual-time algorithm

- Virtual time = usage / shares
- Schedule VM with smallest virtual time

Example: 3 VMs A, B, C with 3 : 2 : 1 share ratio



A	2	4	4	6	8	8	8	10	10
B	3	3	6	6	6	9	9	9	12
C	6	6	6	6	6	6	12	12	12

Inter-Processor Load Balancing

Motivation

- Utilize multiple processors efficiently
- Enforce global fairness
- Amortize context-switch costs
- Preserve cache affinity

Approach

- Per-processor dispatch and run queues
- Scan remote queues periodically for fairness
- Pull whenever a physical cpu becomes idle
- Push whenever a virtual cpu wakes up
- Consider cache affinity cost-benefit

Co-Scheduling SMP VMs

Motivation

- Maintain illusion of dedicated multiprocessor
- Correctness – avoid guest BSODs / panics
- Performance – consider guest OS spin locks

Approach

- Limit “skew” between progress of virtual CPUs
- Idle VCPUs treated as if running
- Co-stop – deschedule all VCPUs on skew accumulation
- Co-start – coschedule VCPUs based on skew threshold

Charging and Accounting

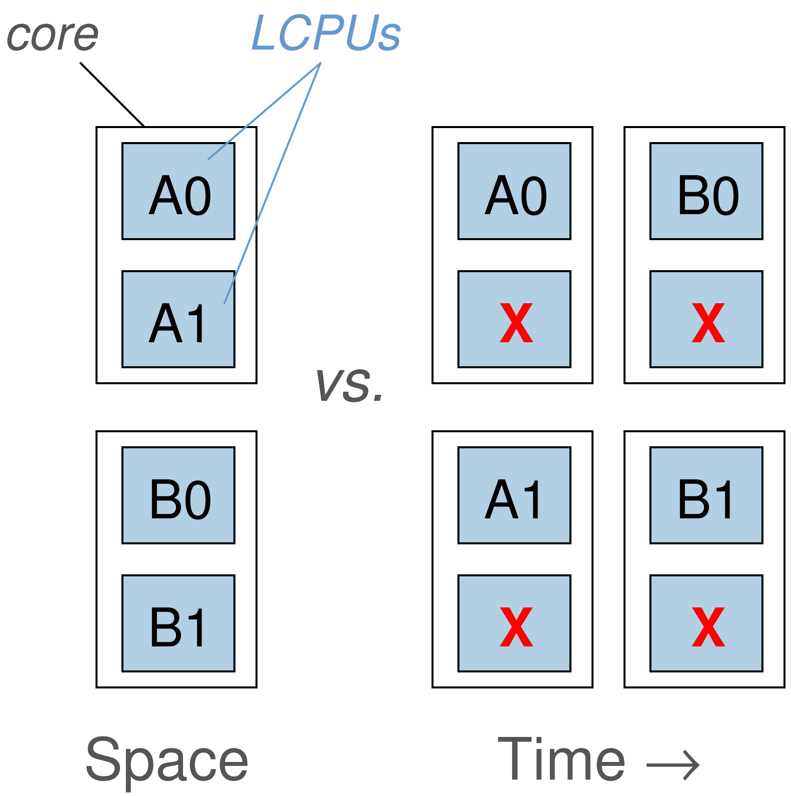
Resource usage accounting

- Charge VM for consumption
- Also charge enclosing resource pools
- Adjust accounting for SMT systems

System time accounting

- Time spent handling interrupts, bottom halves, system threads
- Don't penalize VM that happened to be running
- Instead charge VM on whose behalf system work performed
- Based on statistical sampling to reduce overhead

Hyperthreading Example



Sharing Alternatives

Intel Hyperthreading

- Two threads (LCPUs) per core
- No explicit priorities, fairness
- Halt one ⇒ all resources to other

Mapping VCPUs → LCPUs

- Time share vs. space share
- Depends on dynamic VM entitlements
- Idle thread may preempt VCPU
- Adjust accounting when partner halts

HT sharing controls

- µArch denial of service [Grunwald '02]
- Manual: any, internal, none
- Automatic quarantining

Processor Scheduling: Future Directions

Shared cache management

- Explicit cost-benefit tradeoffs for migrations
- Explore cache partitioning techniques

Power management

- Exploit frequency and voltage scaling
- Without compromising accounting and rate guarantees

Guest hot-add/remove processors

Talk Overview

- Resource controls
- Processor scheduling
- Memory management
- NUMA scheduling
- Distributed systems
- Summary

Memory Management

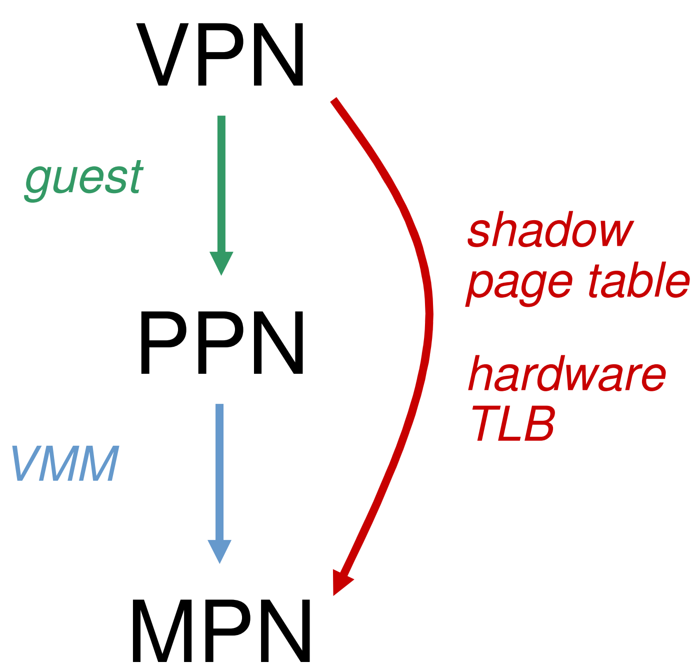
Desirable capabilities

- Efficient memory overcommitment
- Accurate resource controls
- Exploit sharing opportunities
- Leverage hardware capabilities

Challenges

- Allocations should reflect both importance and working set
- Best data to guide decisions known only to guest OS
- Guest and meta-level policies may clash

Memory Virtualization



Traditional VMM Approach

Extra Level of Indirection

- Virtual → "Physical"
  Guest maps VPN to PPN
  using primary page tables
- "Physical" → Machine
  VMM maps PPN to MPN

Shadow Page Table

- Composite of two mappings
- For ordinary memory references
  hardware maps VPN to MPN

Emerging NPT/EPT hardware

- Hardware MMU support for
  nested page tables
- No need for software shadows

VMware Memory Management

Reclamation mechanisms

- Ballooning – guest driver allocates pinned PPNs, hypervisor deallocates backing MPNs
- Swapping – hypervisor transparently pages out PPNs, paged in on demand
- Page sharing – hypervisor identifies identical PPNs based on content, maps to same MPN copy-on-write

Allocation policies

- Proportional sharing – revoke memory from VM with minimum shares-per-page ratio
- Idle memory tax – charge VM more for idle pages than for active pages to prevent unproductive hoarding
- Large pages – exploit large mappings to improve TLB performance

Reclaiming Memory

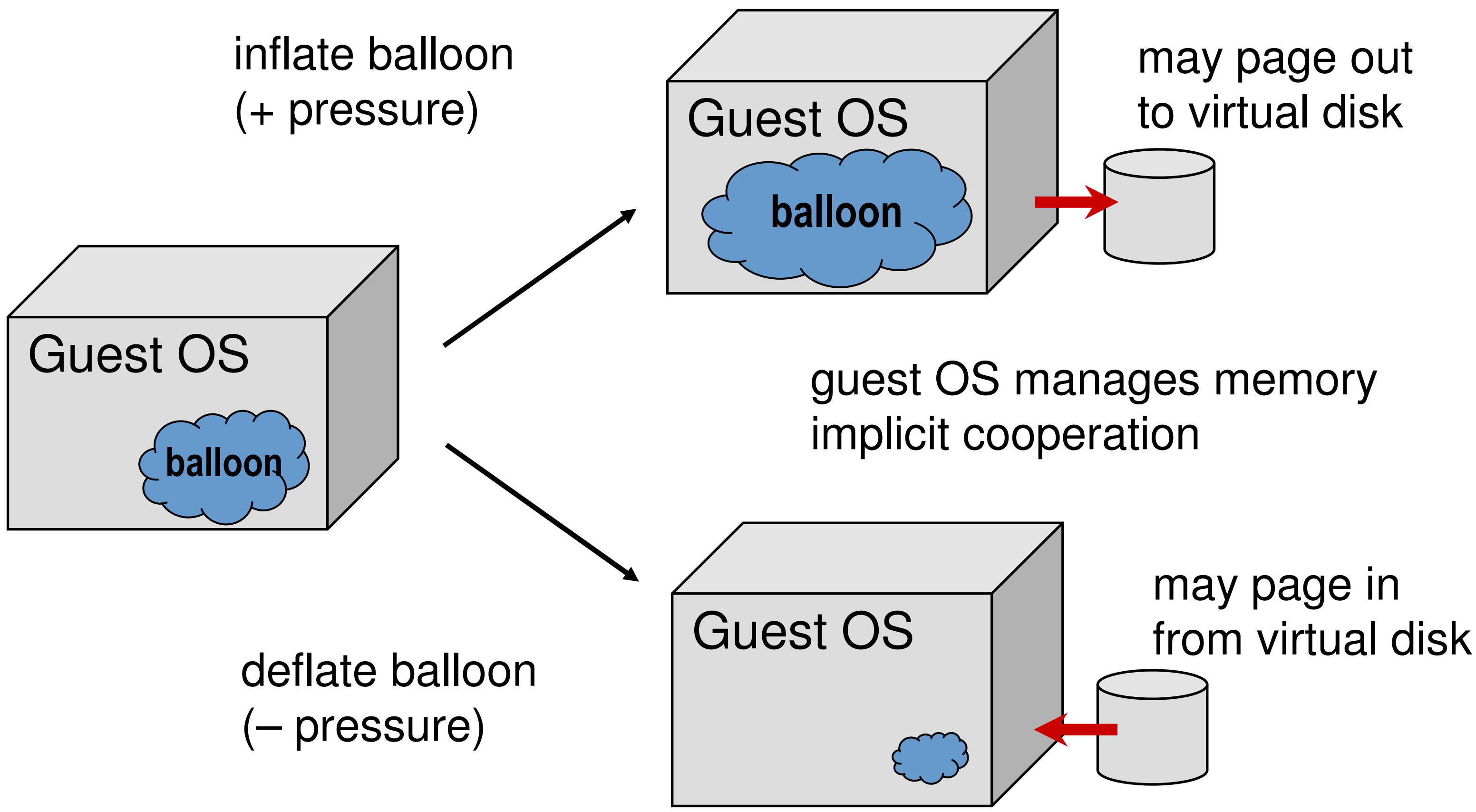
Traditional: add transparent swap layer

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

Page Sharing

Motivation

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

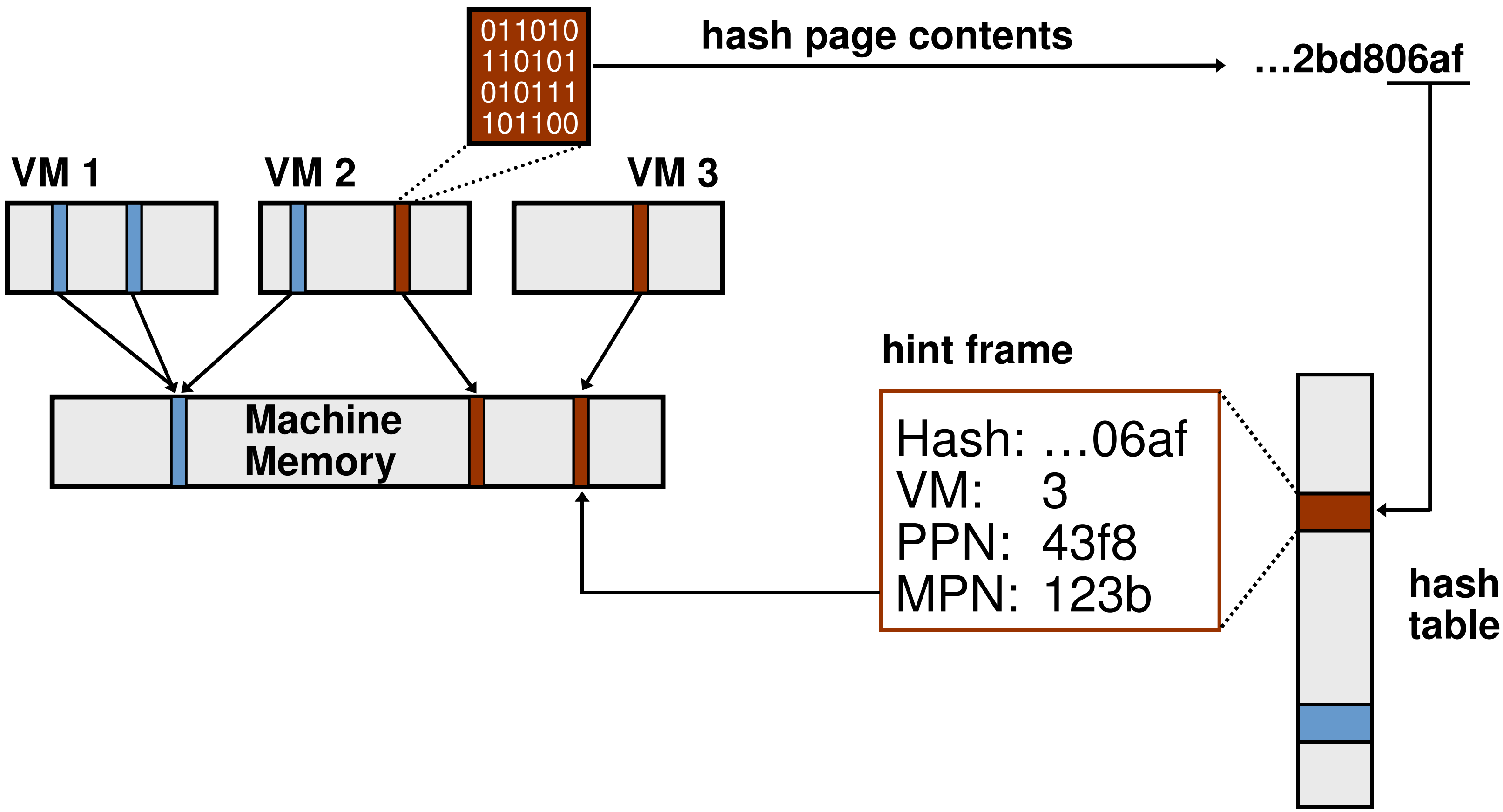
Transparent page sharing

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

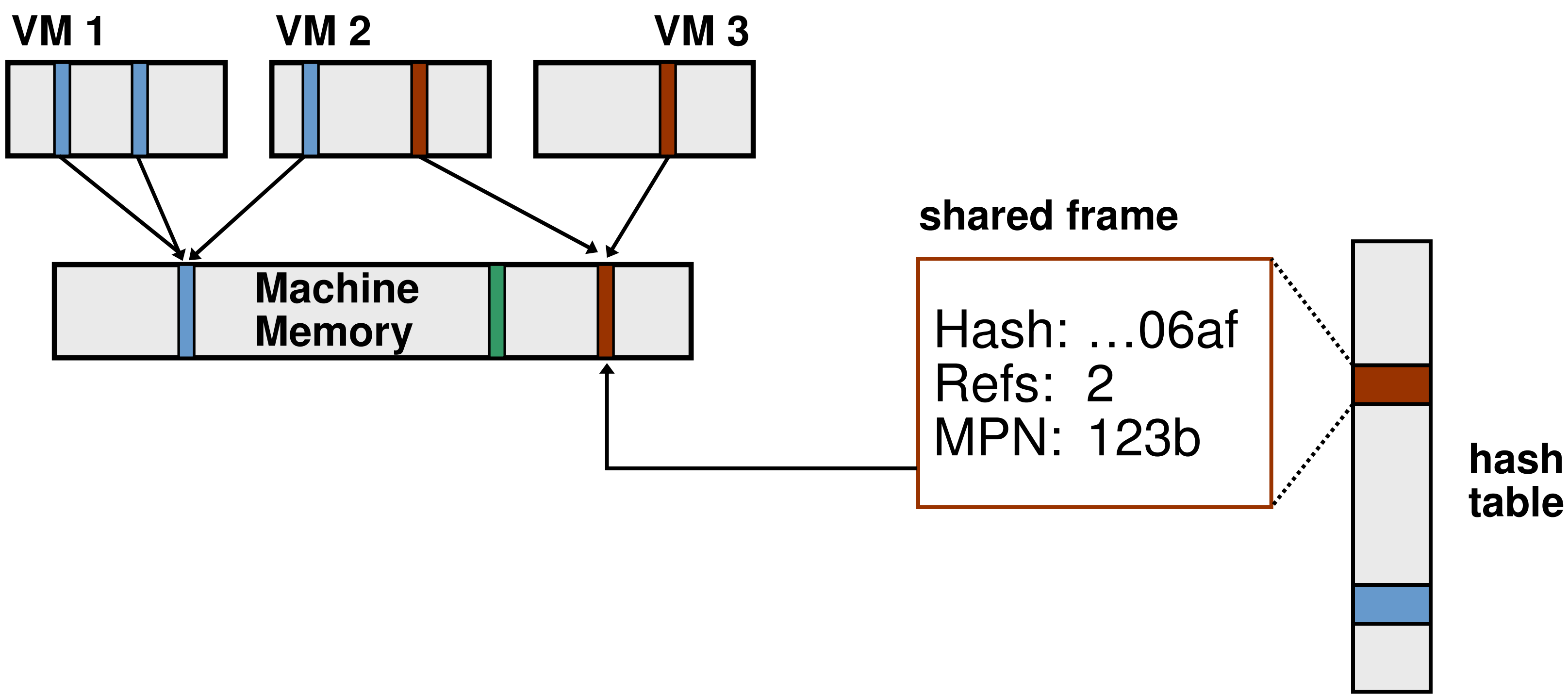
Content-based sharing

- General-purpose, no guest OS changes
- Background activity saves memory over time

Page Sharing: Scan Candidate PPN



Page Sharing: Successful Match



Memory Management: Future Directions

Further leverage page remapping

- Dynamic memory defragmentation for large pages
- Power management [Huang '03]

Exploit hardware trends

- I/O MMU support for isolation, remapping
- Additional optimizations for NPT/EPT

Guest hot-add/remove memory

Guest memory pressure estimation

Talk Overview

Resource controls

Processor scheduling

Memory management

NUMA scheduling

Distributed systems

Summary

NUMA Scheduling

NUMA platforms

- Non-uniform memory access
- Node = processors + local memory + cache
- Examples: IBM x460 (Intel Xeon), HP DL585 (AMD Opteron)

Useful features

- Automatically map VMs to NUMA nodes
- Dynamic rebalancing

Challenges

- Tension between memory locality and load balance
- Lack of detailed hardware counters on commodity platforms

VMware NUMA Scheduling

Periodic rebalancing

- Compute VM entitlements, memory locality
- Assign “home” node for each VM
- Migrate VMs and pages across nodes

VM migration

- Move all VCPUs and threads associated with VM
- Migrate to balance load, improve locality

Page migration

- Allocate new pages from home node
- Remap PPNs from remote to local MPNs (migration)
- Share MPNs per-node (replication)

Talk Overview

Resource controls

Processor scheduling

Memory management

NUMA scheduling

Distributed systems

Summary

Distributed Systems

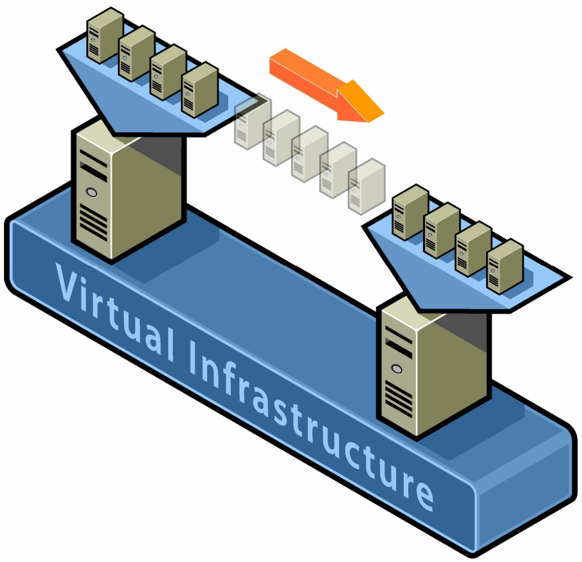
Useful features

- Choose initial host when VM powers on
- Migrate running VM across physical hosts
- Dynamic rebalancing by migrating VMs
- Configurable automation levels
- Utility computing

Challenges

- Migration decisions involve multiple resources
- Resource pools can span multiple hosts
- Appropriate migration thresholds
- Assorted failures modes (hosts, connectivity, etc.)

VMware VMotion



"Hot" migrate VM across hosts

- Transparent to guest OS, apps
- Minimal downtime (sub-second)

Requirements

- Shared storage (*e.g.* SAN/NAS/iSCSI)
- Same subnet (no forwarding proxy)
- Compatible processors

Details

- Bitmap tracks modified pages
- Pre-copy iteration sends modified pages
- Repeatedly pre-copy "diff" until converge
- Exploit meta-data (shared, swapped)

VMware DRS

DRS = Distributed Resource Scheduler

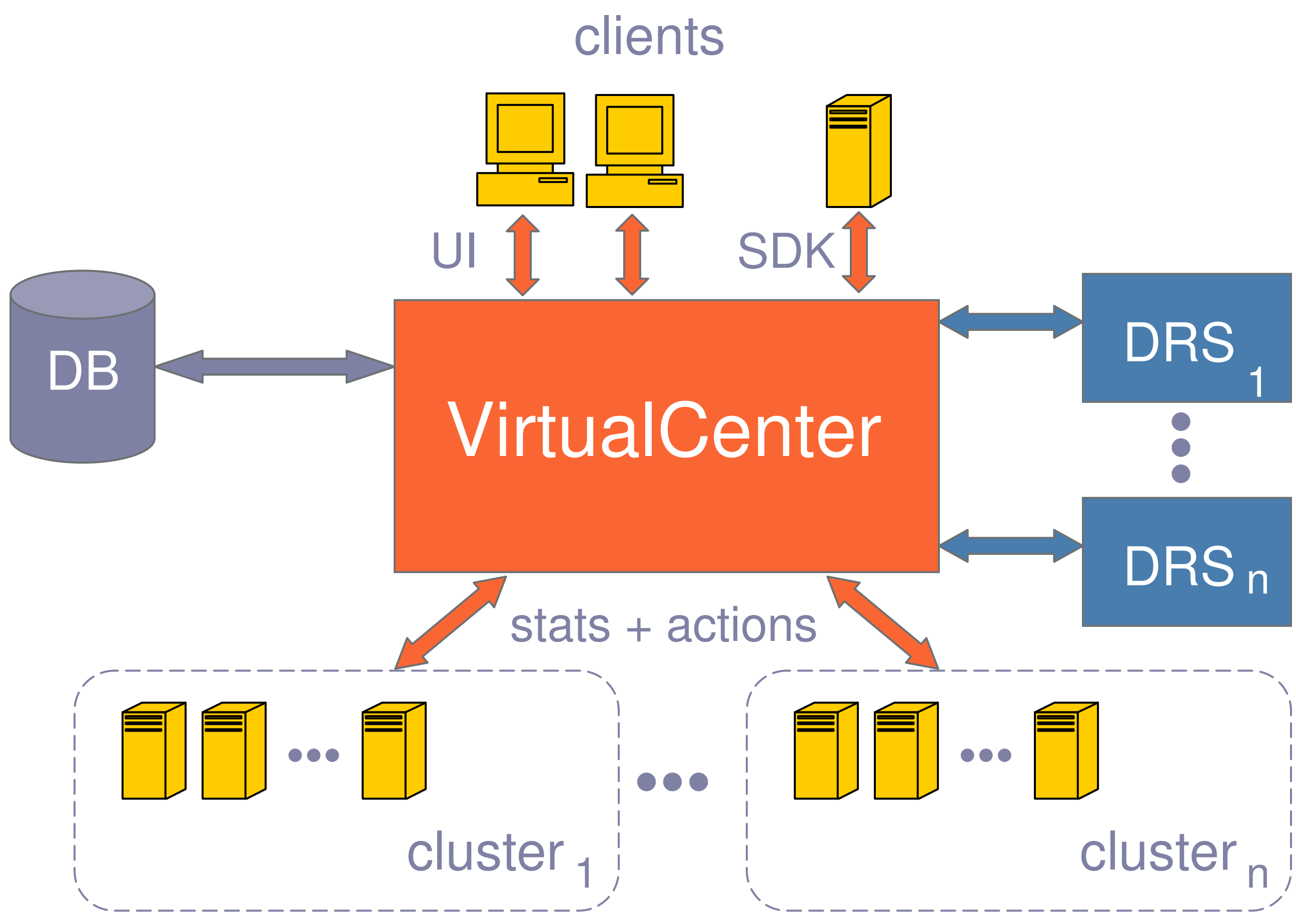
Cluster-wide resource management

- Hierarchical organization and delegation
- Flexible grouping, sharing, and isolation
- Configurable automation levels, aggressiveness
- Configurable VM affinity/anti-affinity rules

Automatic virtual machine placement

- Optimize load balance across hosts
- Choose initial host when VM powers on
- Dynamic rebalancing using VMotion
- React to dynamic load changes

DRS System Architecture



DRS Balancing Details

Compute VM entitlements

- Based on resource pool and VM resource settings
- Don't give VM more than it demands
- Reallocate extra resources fairly

Compute host loads

- Load ≠ utilization unless all VMs equally important
- Sum entitlements for VMs on host
- Normalize by host capacity

Consider possible VMotions

- Evaluate effect on cluster balance
- Incorporate migration cost for involved hosts

Recommend best moves (if any)

Simple Balancing Example

Host normalized entitlement = 1.25

Host normalized entitlement = 0.5

Recommendation: migrate VM2

Distributed Systems: Future Directions

I/O resource management

- Quality of service for networking, SAN
- End-to-end control difficult, complex switching/routing fabric
- Lack of standards, even in non-virtualized environments
- May need to treat storage array as a black box

Proactive migrations

- Detect longer-term trends
- Move VMs based on predicted load

Large-scale WAN/Grid management

Summary

Resource management

- Controls for specifying allocations
- Processor, memory, NUMA, I/O, power
- Tradeoffs between multiple resources

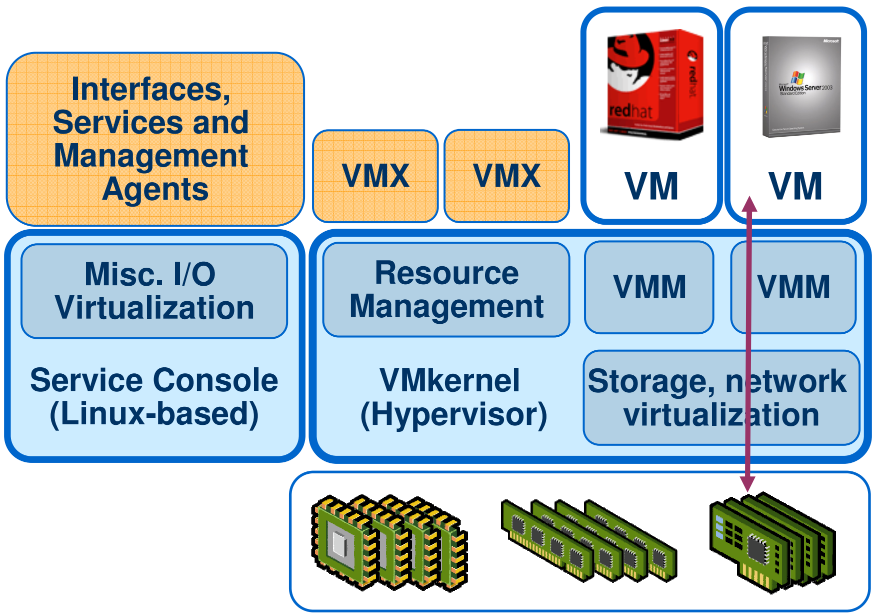
Rich research area

- Plenty of interesting open problems
- Many unique solutions

Backup Slides

Backup Slides…

VMware ESX Server



Bare-metal hypervisor

- Runs directly on hardware
- Commercial product, ESX 3.x
- Designed to run VMs efficiently

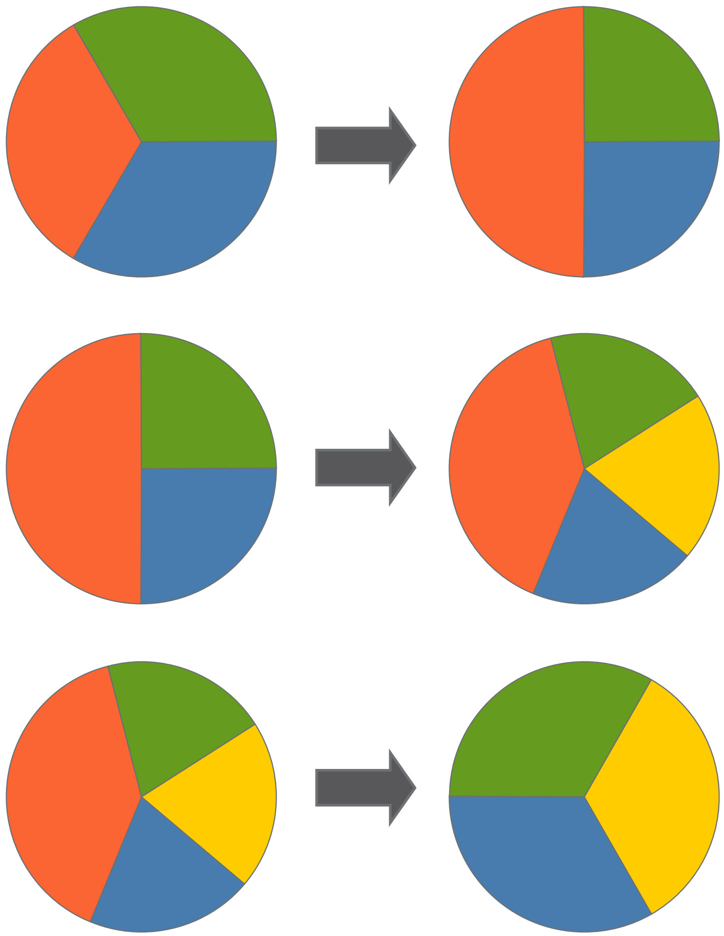
Resource management

- Multiplex physical resources
- Provide QoS, enable overcommit

High-performance I/O

- Direct I/O for most devices
- Service console for legacy I/O, third-party management agents

Shares Examples



Change shares for **VM**

Dynamic reallocation

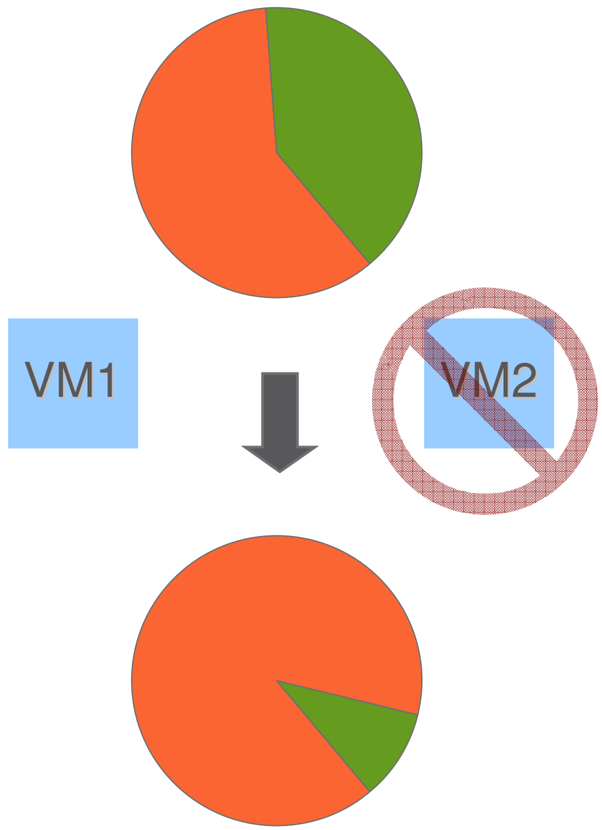
Add **VM**, overcommit

Graceful degradation

Remove **VM**

Exploit extra resources

Reservation Example



Total capacity

- 600 MHz reserved
- 400 MHz available

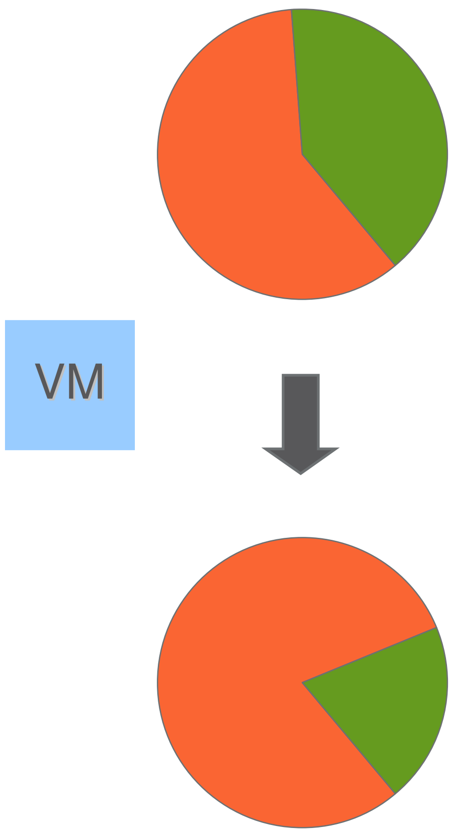
Admission control

- 2 VMs try to power on
- Each reserves 300 MHz
- Unable to admit both

VM1 powers on

VM2 not admitted

Limit Example



Current utilization

- 600 MHz active
- 400 MHz idle

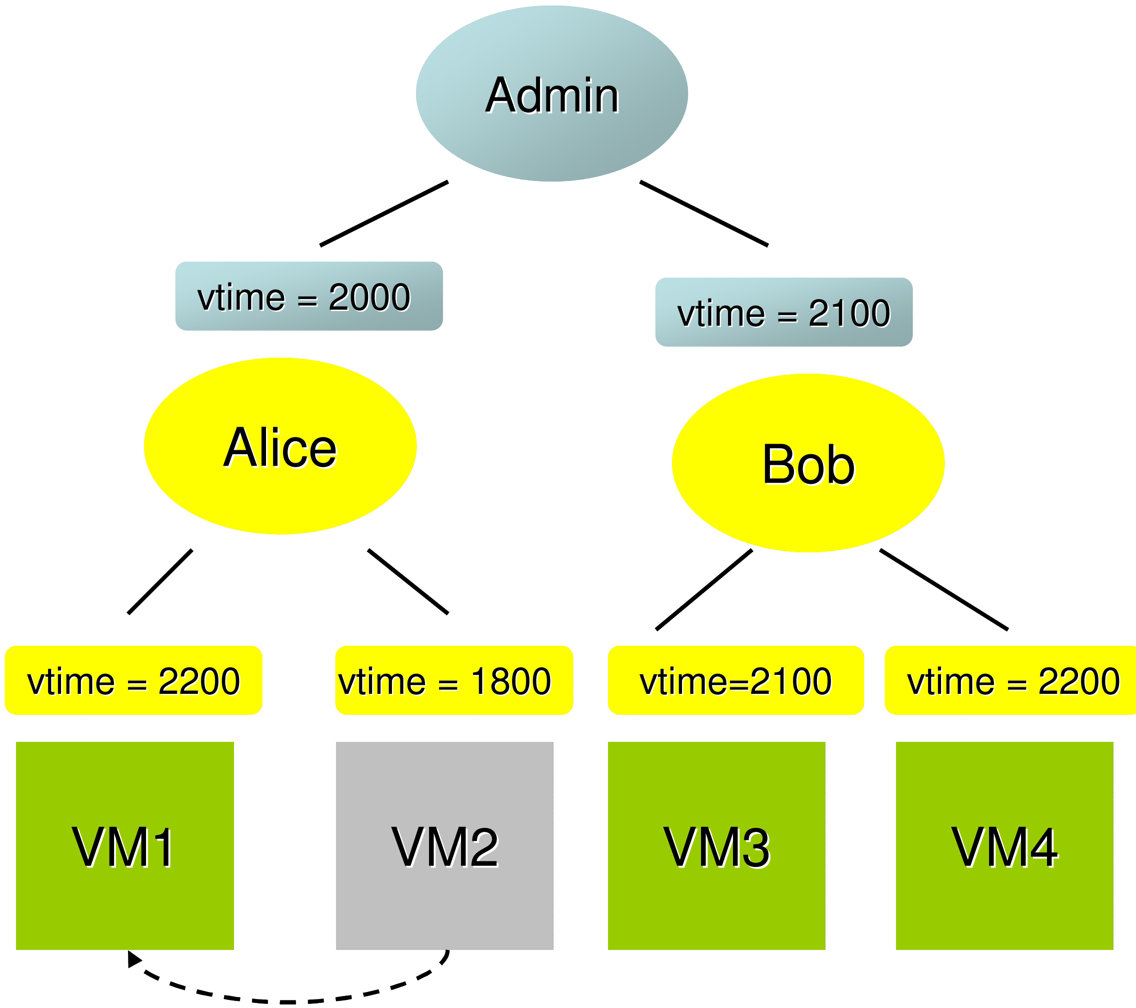
Start CPU-bound VM

- 200 MHz limit
- Execution throttled

New utilization

- 800 MHz active
- 200 MHz idle
- VM prevented from using idle resources

Hierarchical Scheduling



Motivation

- Enforce fairness at each resource pool
- Unused resources flow to closest relatives

Approach

- Maintain virtual time at each node
- Recursively choose node with smallest virtual time

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

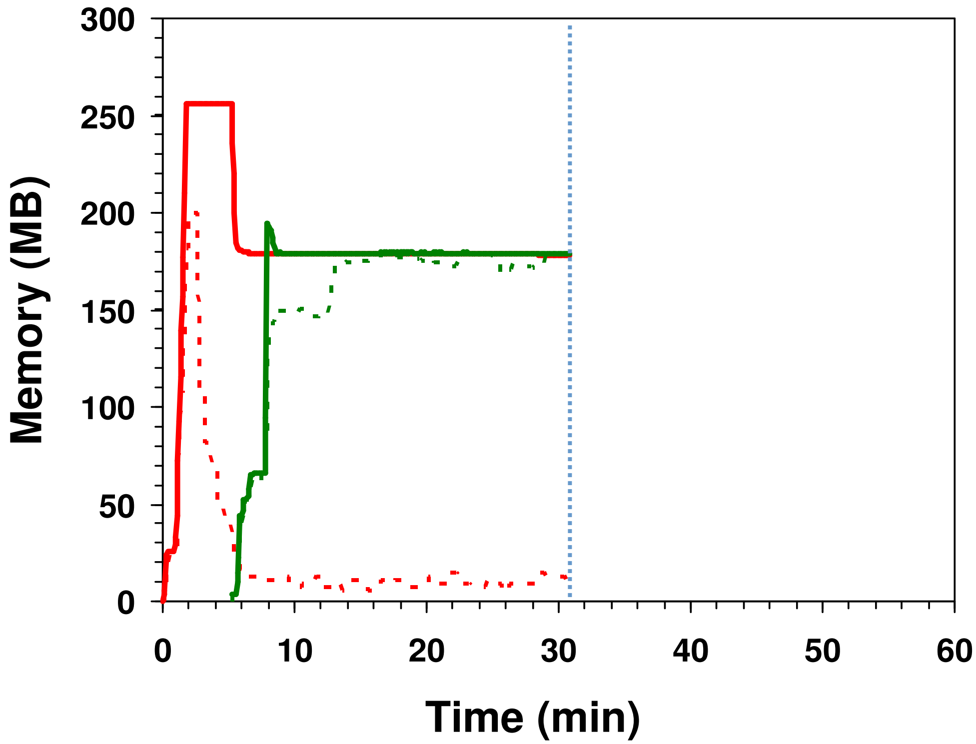
Tax rate

- Explicit administrative parameter
- 0% ≈ “plutocracy” … 100% ≈ “socialism”

High default rate

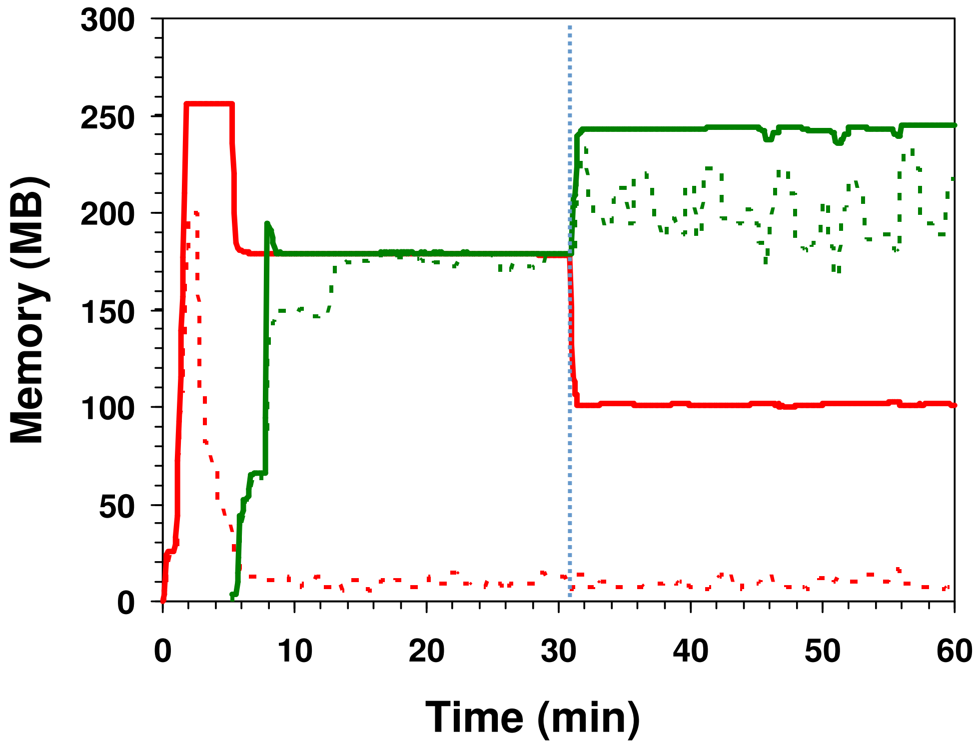
- Reclaim most idle memory
- Some buffer against rapid working-set increases

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%

NUMA Scheduling: Future Directions

Exploit hardware trends

- NUMA + multi-core ≈ nested NUMA
- Inter-node distance weightings (SLIT table)

Better page migration heuristics

- Determine most profitable pages to migrate
- Some high-end systems (*e.g.* SGI Origin) had per-page remote miss counters
- Not available on commodity x86 hardware

Additional Topics

Host-level I/O management

- Arbitrate access to local NICs and HBAs
- Disk I/O bandwidth management
- Network traffic shaping

Guest timer virtualization

- Representing time when VM is descheduled
- VMware timer sponge, para-virtualized approaches

Power management

Multi-resource management

Resource Entitlement

Resources that each VM "deserves"

- Combining shares, reservation, and limit
- Allocation if all VMs fully active (*e.g.*, cpu-bound)
- Concrete units (MHz)

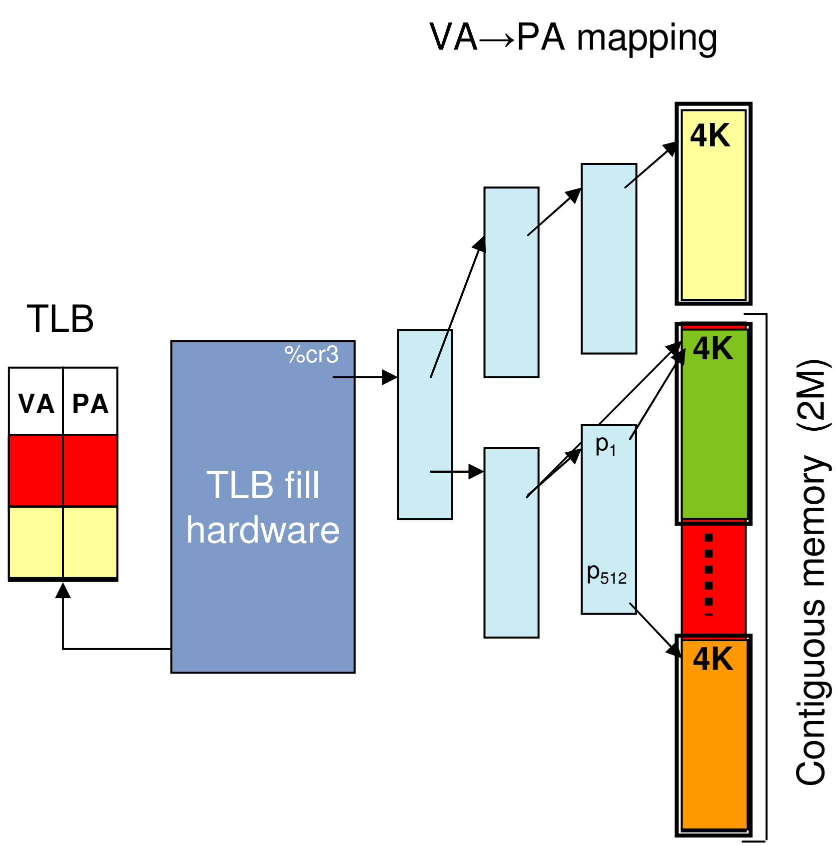
Entitlement calculation (conceptual)

- Entitlement initialized to its Reservation
- Hierarchical entitlement distribution
- Entitlements distributed 1 MHz at a time
- Preferentially to lowest Entitlement / Shares
- Up to Limit

What if VM idles?

- Don't give VM more than it demands
- CPU scheduler distributes resources to active VMs
- Unused reservations not wasted

Large Pages



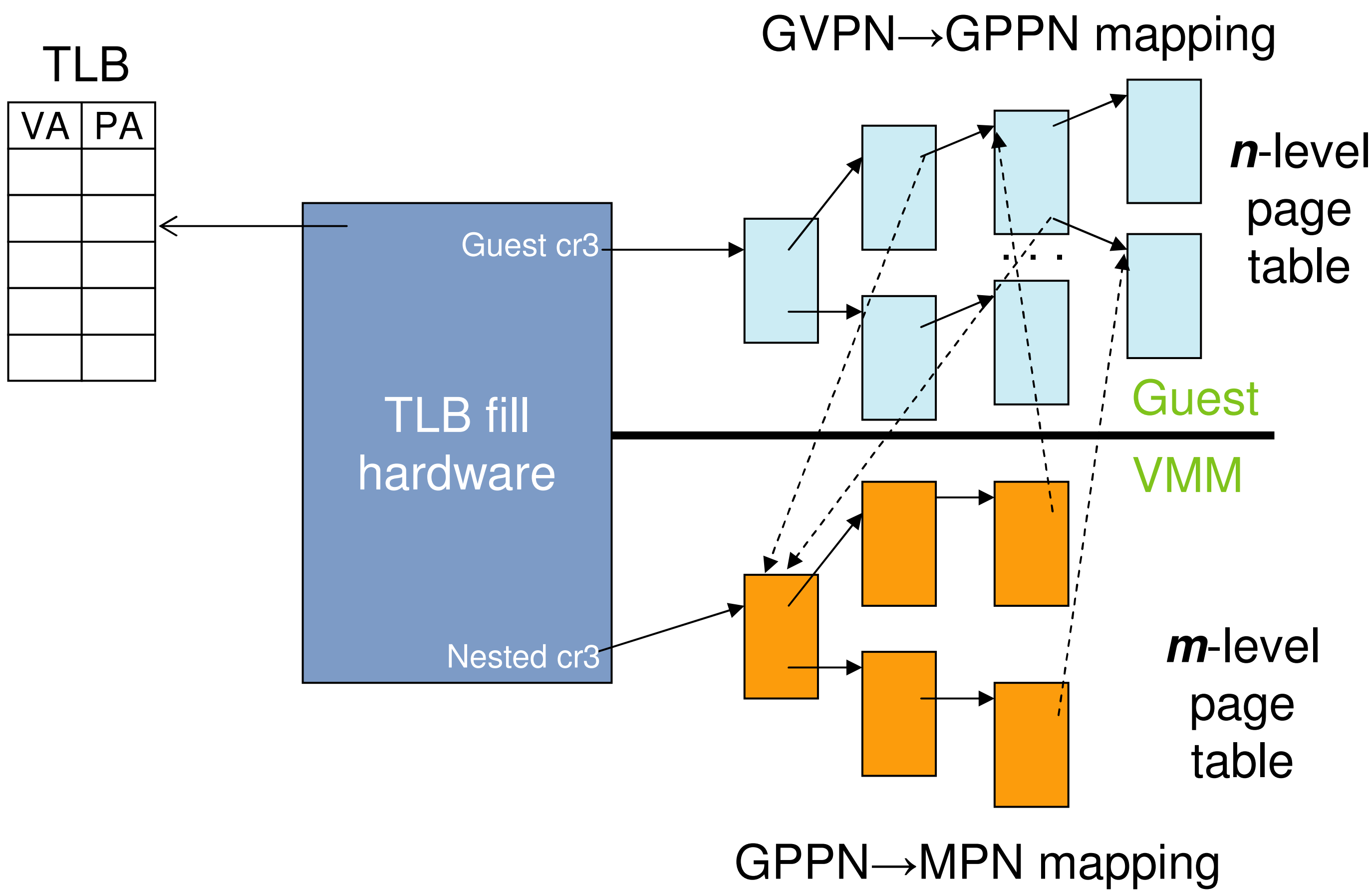
Small page (4 KB)

- Basic unit of x86 memory management
- Single page table entry maps to small 4K page

Large page (2 MB)

- 512 contiguous small pages
- Single page table entry covers entire 2M range
- Helps reduce TLB misses
- Lowers cost of TLB fill

Nested Page Tables



Quadratic page table walk time, O(***n****m*)