Resource Management for Virtualized Systems

Carl Waldspurger (SB SM ’89 PhD ’95)
VMware R&D
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

• Support flexible policies
  – 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 cloud
VMware Basic Controls

• Shares
  – Specify relative importance
  – Entitlement directly proportional to shares
  – Abstract relative units, only ratios matter

• Reservation
  – Minimum guarantee, even when system overloaded
  – Concrete absolute units (MHz, MB)
  – Admission control: sum of reservations ≤ capacity

• Limit
  – Upper bound on consumption, even if underloaded
  – Concrete absolute units (MHz, MB)
Shares Examples

- Change shares for **VM**
- Dynamic reallocation

- Add **VM**, overcommit
- Graceful degradation

- Remove **VM**
- Exploit extra resources
Reservation Example

• Total capacity
  – 1800 MHz reserved
  – 1200 MHz available

• Admission control
  – 2 VMs try to power on
  – Each reserves 900 MHz
  – Unable to admit both

• VM1 powers on
• VM2 not admitted

[Diagram showing reservation and admission control process]
Limit Example

• Current utilization
  – 1800 MHz active
  – 1200 MHz idle

• Start CPU-bound VM
  – 600 MHz limit
  – Execution throttled

• New utilization
  – 2400 MHz active
  – 600 MHz idle
  – VM prevented from using idle resources
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 is 50% more than Bob’s
- Users manage own VMs
- Not shown: resvs, 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

• Application-level metrics
  – Users think in terms of transaction rates, response times
  – Requires detailed app-specific knowledge and monitoring
  – Can layer on top of basic physical resource controls

• Other controls?
  – Real-time latency guarantees
  – Price-based mechanisms and multi-resource tradeoffs

• Emerging DMTF standard
  – Reservation, limit, “weight” + resource pools
  – Authors from VMware, Microsoft, IBM, Citrix, etc.
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 μarchitectural 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
  – Support hot-add 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
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
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

• VMware Approach
  – Limit “skew” between progress of virtual CPUs
  – Idle VCPUs treated as if running
  – Deschedule VCPUs that are too far ahead
  – Schedule VCPUs that are behind

• Alternative: Para-virtualization
Charging and Accounting

- **Resource usage accounting**
  - Pre-requisite for enforcing scheduling policies
  - Charge VM for consumption
  - Also charge enclosing resource pools
  - Adjust accounting for SMT systems

- **System time accounting**
  - Time spent handling interrupts, BHs, system threads
  - Don’t penalize VM that happened to be running
  - Instead charge VM on whose behalf work performed
  - Based on statistical sampling to reduce overhead
Processor Scheduling: Future Directions

• Shared cache management
  – Explicit cost-benefit tradeoffs for migrations
    e.g. based on cache miss-rate curves (MRCs)
  – Compensate VMs for co-runner interference
  – Hardware cache QoS techniques

• Power management
  – Exploit frequency and voltage scaling (P-states)
  – Exploit low-power, high-latency halt states (C-states)
  – Without compromising accounting and rate guarantees
Talk Overview

• Resource controls
• Processor scheduling
• Memory management
• NUMA scheduling
• Distributed systems
• Summary
Memory Management

• Useful features
  – Efficient memory overcommitment
  – Accurate resource controls
  – Exploit deduplication opportunities
  – Leverage hardware capabilities

• Challenges
  – Reflecting both VM importance and working-set
  – Best data to guide decisions private to guest OS
  – Guest and meta-level policies may clash
Memory Virtualization

- 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
  - Traditional VMM approach
  - Composite of two mappings
  - For ordinary memory references, hardware maps VPN to MPN

- Nested page table hardware
  - Recent AMD RVI, Intel EPT
  - VMM manages PPN-to-MPN table
  - No need for software shadows
Reclaiming Memory

• Required for memory overcommitment
  – Increase consolidation ratio, incredibly valuable
  – Not supported by most hypervisors
  – Many VMware innovations [Waldspurger OSDI ’02]

• Traditional: add transparent swap layer
  – Requires meta-level page replacement decisions
  – Best data to guide decisions known only by guest
  – 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)
  - May page out to virtual disk
  - Guest OS manages memory
  - Implicit cooperation

- **Deflate Balloon** (– pressure)
  - May page in from virtual disk
  - Guest OS manages memory
  - Implicit cooperation
Page Sharing

• **Motivation**
  – Multiple VMs running same OS, apps
  – Deduplicate redundant copies of code, data, zeros

• **Transparent page sharing**
  – Map multiple PPNs to single MPN copy-on-write
  – Pioneered by Disco [Bugnion et al. SOSP ’97], but required guest OS hooks

• **VMware content-based sharing**
  – General-purpose, no guest OS changes
  – Background activity saves memory over time
Page Sharing: Scan Candidate PPN

VM 1  VM 2  VM 3

Machine Memory

hash page contents

...06af 110101 010111 101100

Machine Memory

hash table

hash: ...06af
VM: 3
PPN: 43f8
MPN: 123b

hint frame
Page Sharing: Successful Match

VM 1  VM 2  VM 3

Machine Memory

Hash: …06af Refs: 2 MPN: 123b

shared frame

hash table
Memory Reclamation: Future Directions

• Memory compression
  – Old idea: compression cache [Dougais USENIX ’93], Connectix RAMDoubler (MacOS mid-90s)
  – Recent: Difference Engine [Gupta et al. OSDI ’08], future VMware ESX release

• Sub-page deduplication

• Emerging memory technologies
  – Swapping to SSD devices
  – Leveraging phase-change memory
Memory 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
  – Problem: ignores usage, unproductive hoarding

• 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**
  - Redistributed VM1 → VM2
  - VM1 reduces to min size
  - VM2 throughput improves more than 30%
Allocation Policy: Future Directions

• Memory performance estimates
  – Estimate effect of changing allocation
  – Miss-rate curve (MRC) construction
• Improved coordination of mechanisms
  – Ballooning, compression, SSD, swapping
• Leverage guest hot-add/remove
• Large page allocation efficiency and fairness
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 x-Series, AMD Opteron, Intel Nehalem

• Useful features
  – Automatically map VMs to NUMA nodes
  – Dynamic rebalancing

• Challenges
  – Tension between memory locality and load balance
  – Lack of detailed counters on commodity hardware
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)
NUMA Scheduling: Future Directions

• 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 platforms

• Expose NUMA to guest?
  – Enable guest OS optimizations
  – Impact on portability
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 VMs across physical hosts
  – Dynamic load balancing
  – Support cloud computing, multi-tenancy

• Challenges
  – Migration decisions involve multiple resources
  – Resource pools can span many hosts
  – Appropriate migration thresholds
  – Assorted failure 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 (EVC)

- Details
  - Track modified pages (write-protect)
  - Pre-copy step sends modified pages
  - Keep sending "diffs" until converge
  - Start running VM on destination host
  - Exploit meta-data (shared, swapped)
VMware DRS/DPM

• DRS = Distributed Resource Scheduler
• Cluster-wide resource management
  – Uniform controls, same as available on single host
  – Flexible hierarchical policies and delegation
  – Configurable automation levels, aggressiveness
  – Configurable VM affinity/anti-affinity rules
• Automatic VM placement
  – Optimize load balance across hosts
  – Choose initial host when VM powers on
  – Dynamic rebalancing using vMotion
• DPM = Distributed Power Management
  – Power off unneeded hosts, power on when needed again
DRS System Architecture

VirtualCenter

clients

DB

UI

SDK

stats + actions

cluster

DRS

n

cluster

r

n
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 $\neq$ 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-benefit for involved hosts

• Recommend best moves (if any)
Simple Balancing Example

Recommendation: migrate VM2
DPM Details (Simplified)

• Set target host demand/capacity ratio (63% ± 18%)
  – If some hosts above target range, consider power on
  – If some hosts below target range, consider power off

• For each candidate host to power on
  – Ask DRS “what if we powered host off and rebalanced?”
  – If more hosts within (or closer to) target, recommend action
  – Stop once no hosts are above target range

• For each candidate host to power off
  – Ask DRS “what if we powered host off and rebalanced?”
  – If more hosts within (or closer to) target, recommend action
  – Stop once no hosts are below target range
Distributed I/O Management

• Host-level I/O scheduling
  – Arbitrate access to local NICs and HBAs
  – Disk I/O bandwidth management (SFQ)
  – Network traffic shaping

• Distributed systems
  – Host-level scheduling insufficient
  – Multiple hosts access same storage array / LUN
  – Array behavior complex, need to treat as black box
  – VMware PARDA approach [Gulati et al. FAST ’09]
PARDA Architecture

Queue lengths varied dynamically based on average request latency
PARDA End-to-End I/O Control

- Shares respected independent of VM placement
- Specified I/O latency threshold enforced (25 ms)
Distributed Systems: Future Directions

• Large-scale cloud management
• Virtual disk placement/migrations
  – Leverage “storage vMotion” as primitive
  – Storage analog of DRS
  – VMware BASIL approach [Gulati et al. FAST ’10]
• Proactive migrations
  – Detect longer-term trends
  – Move VMs based on predicted load
Summary

• Resource management
  – Controls for specifying allocations
  – Processor, memory, NUMA, I/O, power
  – Tradeoffs between multiple resources
  – Distributed resource management

• Rich research area
  – Plenty of interesting open problems
  – Many unique solutions
Backup Slides
CPU Resource Entitlement

- Resources that each VM “deserves”
  - Combining shares, reservation, and limit
  - Allocation if all VMs full active (e.g. CPU-bound)
  - Concrete units (MHz)

- Entitlement calculation (conceptual)
  - Entitlement initialized to reservation
  - Hierarchical entitlement distribution
  - Fine-grained distribution (e.g. 1 MHz at a time), preferentially to lowest entitlement/shares
  - Don’t exceed 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, $O(n^*m)$