Performance Management in the Virtual Data Center, Part II Memory Management

Mark B. Friedman

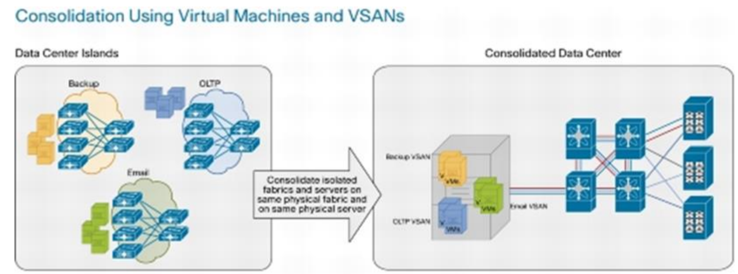
markf@demandtech.com

The Vision:

- Virtualization technology and delivery of IT Infrastructure as a Service (IaaS)
  - *Elastic, multi-tenant computing capacity encompassing virtualized servers, storage and shared network bandwidth*

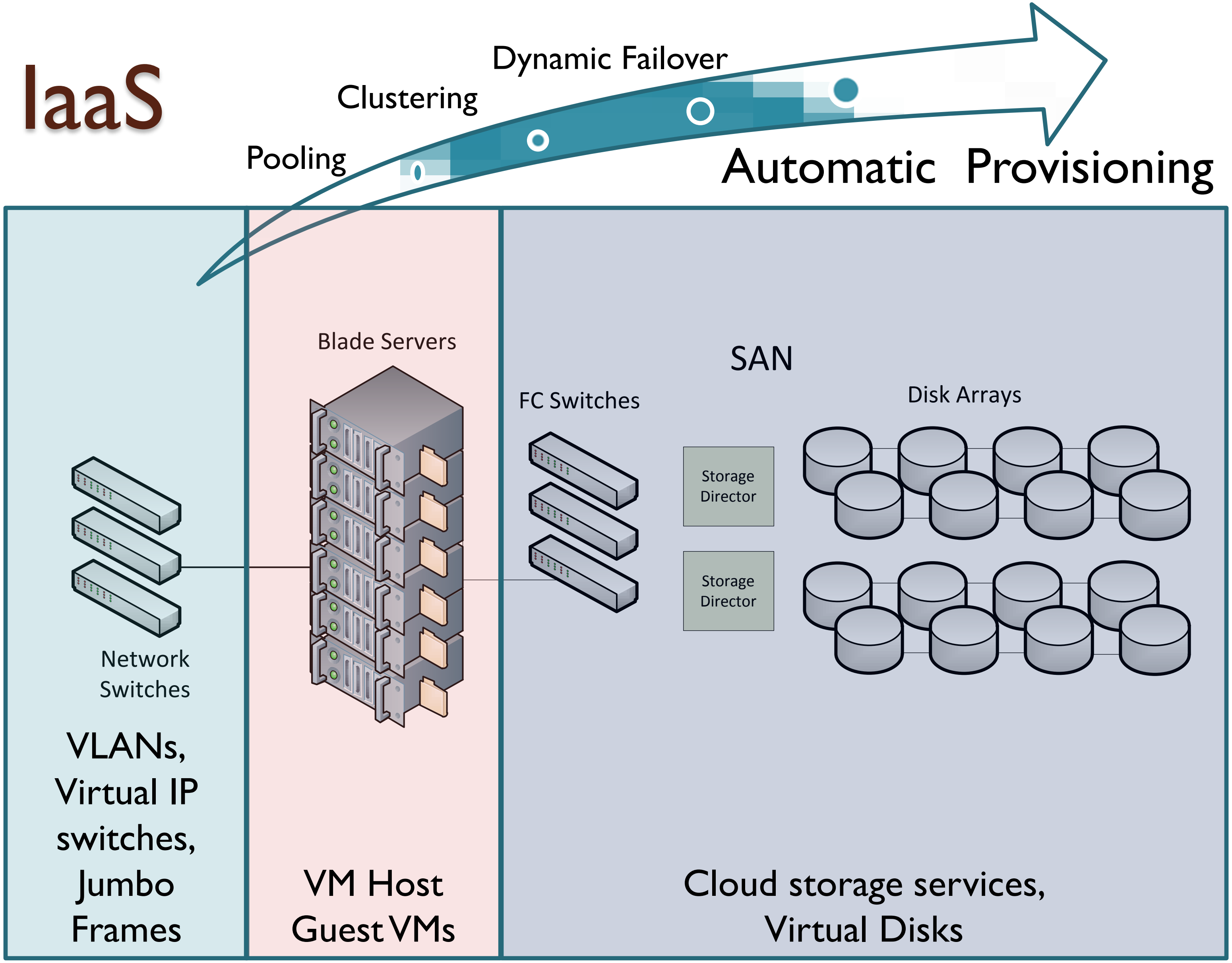
IaaS

- Advertised Benefits
  - Cost savings
  - Economies of scale
  - Capacity on demand
  - Rapid provisioning
  - High Availability
  - Management automation
  - Dynamic load balancing
  - etc.

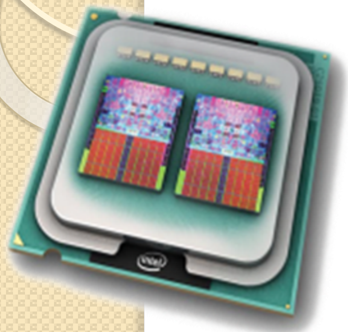


	Attribute	
More	Number of SAN Switches and Servers	Fewer
No	Shared Disk or Tape	Yes
No	Shared Disaster Recovery Facilities	Yes
Complex	Server and SAN Management	Simple
High	Overall TCO	Low

IaaS


Pooling
Clustering
Dynamic Failover
Automatic Provisioning
Blade Servers
SAN
FC Switches
Disk Arrays
Storage Director
Storage Director
Network Switches
VLANs, Virtual IP switches, Jumbo Frames
VM Host Guest VMs
Cloud storage services, Virtual Disks

e.g., Intel hardware assistance:



Intel® VT-x

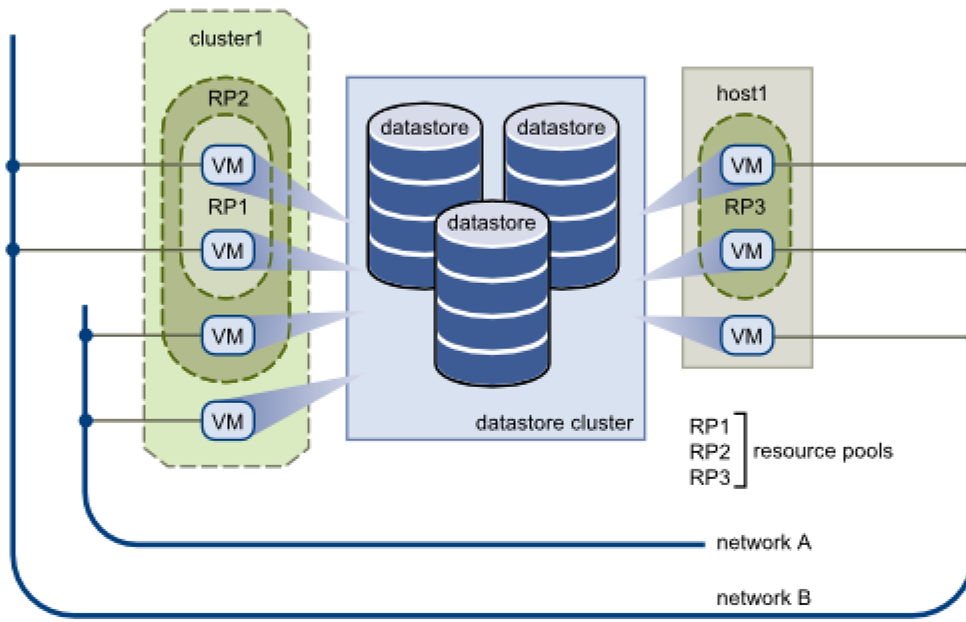
Intel® VT-x
Hardware assists for robust virtualization
Intel® VT FlexMigration – Flexible live migration
Intel® VT FlexPriority – Interrupt acceleration
Intel® EPT – Memory Virtualization

- VT-x® provides architected assists to allow guest OSes to run directly on hardware
- On Nehalem and Westmere VT-x is extended with:

Extended Page Tables (EPT)	Eliminates VM exits to the VMM for shadow page-table maintenance
Virtual Processor IDs (VPID)	Avoid flushes on VM transitions to give a lower-cost VM transition time
Guest Preemption Timer -- lets a VMM preempt a guest OS	Aids VMM vendors in flexibility and Quality of Service (QoS)
Descriptor Table Exiting –Traps on modifications of guest DTs	Allows VMM to protect a guest from internal attack
Transition Latency reductions	Continuing improvements in microarchitectural handling of VMM round trips

e.g., VMware vSphere

- Virtualizes the entire IT infrastructure, including servers, storage, and networks.
- Pools these resources and presents a uniform set of elements that are subject to policy-based management.



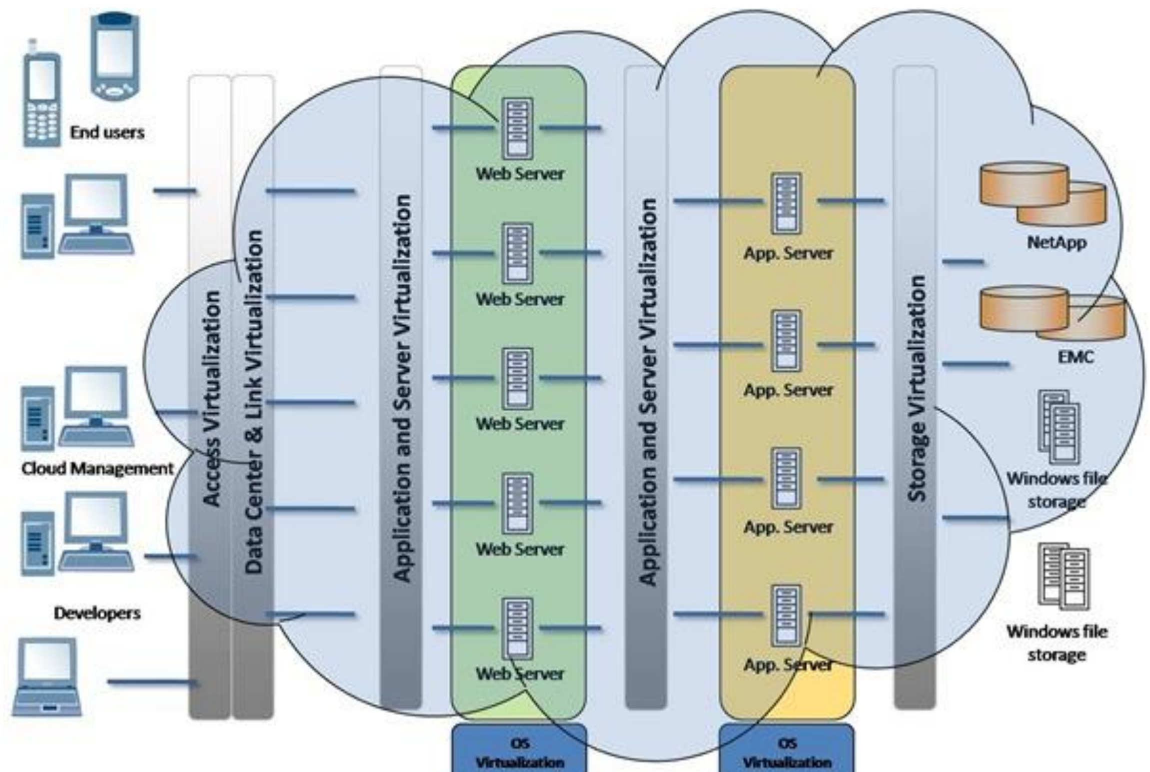
Source: http://pubs.vmware.com/vsphere-50/

e.g., VMware vSphere

- Resources that can be pooled:
  - CPU
  - RAM
  - Power
  - Storage
  - Network bandwidth
- Policy-based settings for Resource allocation
  - Priority-based (High, Normal, Low ≈ 4:2:1)
  - Reservation
  - Allocation Upper Limits

Vision vs. Reality

- VMware is both complex and costly
- Growing numbers of VMware specialists required to support this environment.



Vision vs. Reality

- For VMware to manage a virtualized datacenter effectively, ***all*** the major functions of the Server OS have to migrate into the Hypervisor OS

- Security
- Identity
- TCP/IP protocol stack
- Thread scheduling (SMP, NUMA, etc.)
- IO Device Drivers
- Virtual memory management
- Clustering and HA Failover

Windows Server OS

Vision vs. Reality: Performance

- VMware's vision is ambitious
- Cost justification is mainly via ***aggressive*** server consolidation
- Common Performance concerns include:
  - Can more efficient use of computer resources be achieved through pooling?
  - Can disruptions due to automated operations be minimized or controlled?
  - Can application scalability be achieved through JIT auto-provisioning?
  - Are VMware's performance tools adequate to monitor these complex configurations?

Vision vs. Reality: Performance

- Automation:
  - When a resource is severely ***over-committed***, vMotion will initiate re-balancing the hardware.
  - Not clear precisely what sequence of events trigger vMotion

- VMware implements ***very basic*** policies for resource sharing e.g.,

High ⇨	**Critical Infrastructure**
Normal ⇨	Production
Low ⇨	Test

Vision vs. Reality: Performance

- VMware load balancing is anonymous!

- VMware has little or no knowledge of
  - what applications the guest is running
  - what service levels that application is delivering (or requires)
  - how resources – code, data, memory pages, files, etc. – are allocated across running processes

Vision vs. Reality: Performance

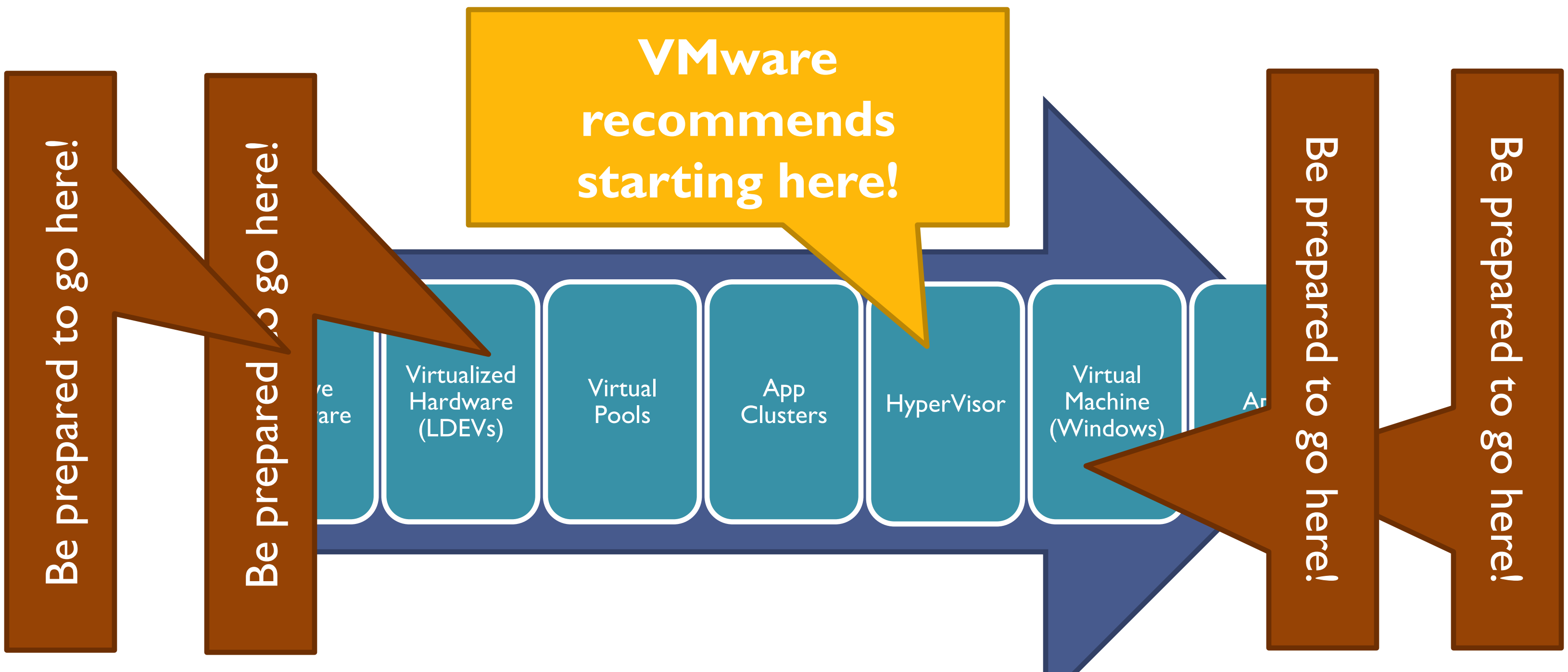
- Massive over-provisioning
  - Typical Blade Servers contain ***far more power*** than typical Application servers require
  - Additional spare capacity/redundancy also supports rapid recovery from failures
- Large-scale aggregation (pooling) reduces the impact of spikes in demand
  - e.g., in a large scale storage processor, disk space consumed grows at a glacial pace

- On massively over-provisioned Hosts,

  guest machine performance ⇨ native performance

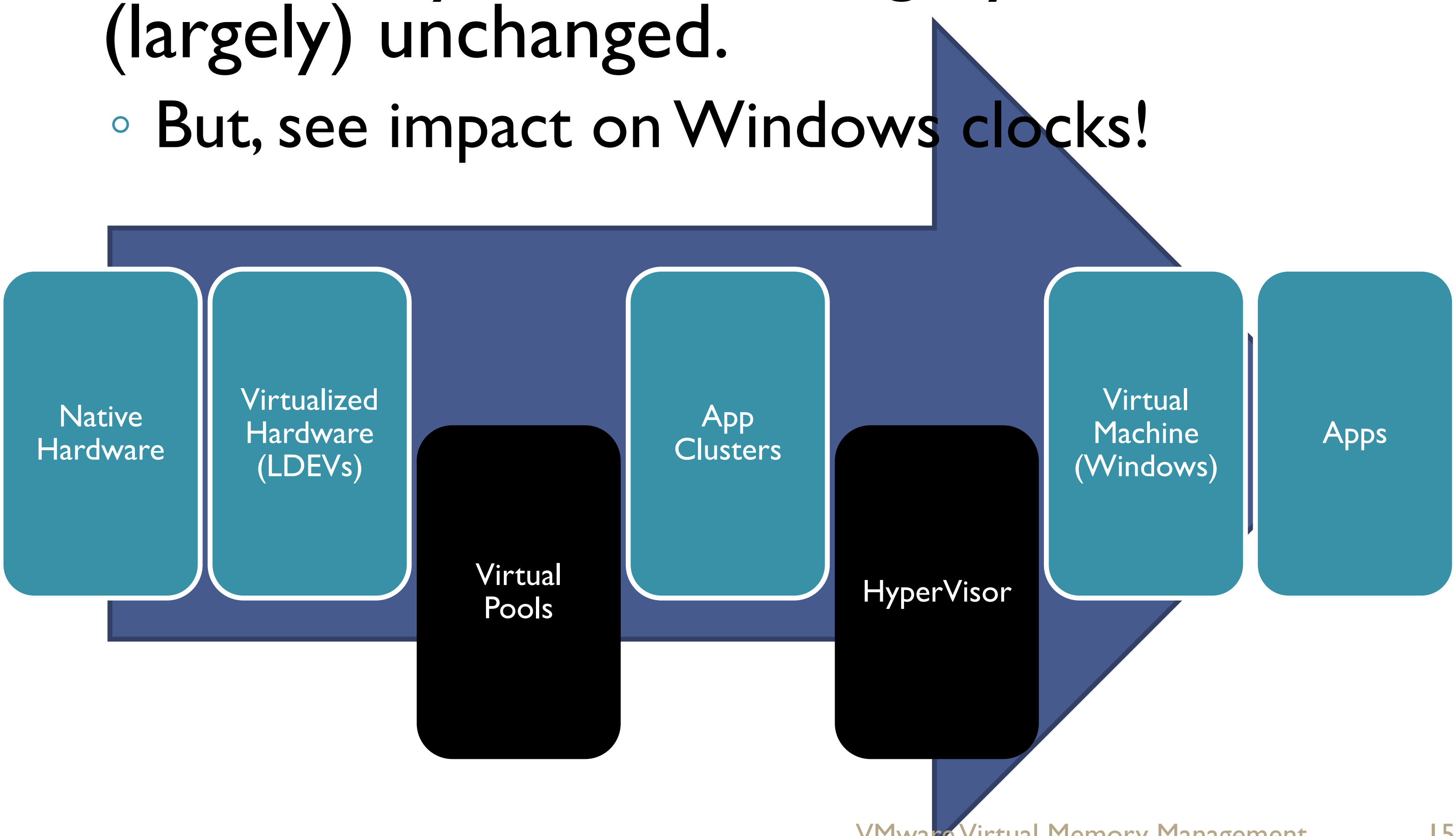
Multiple layers of performance data

- Multiple layers add complexity
- How many layers will you need to drill into to solve a performance problem?



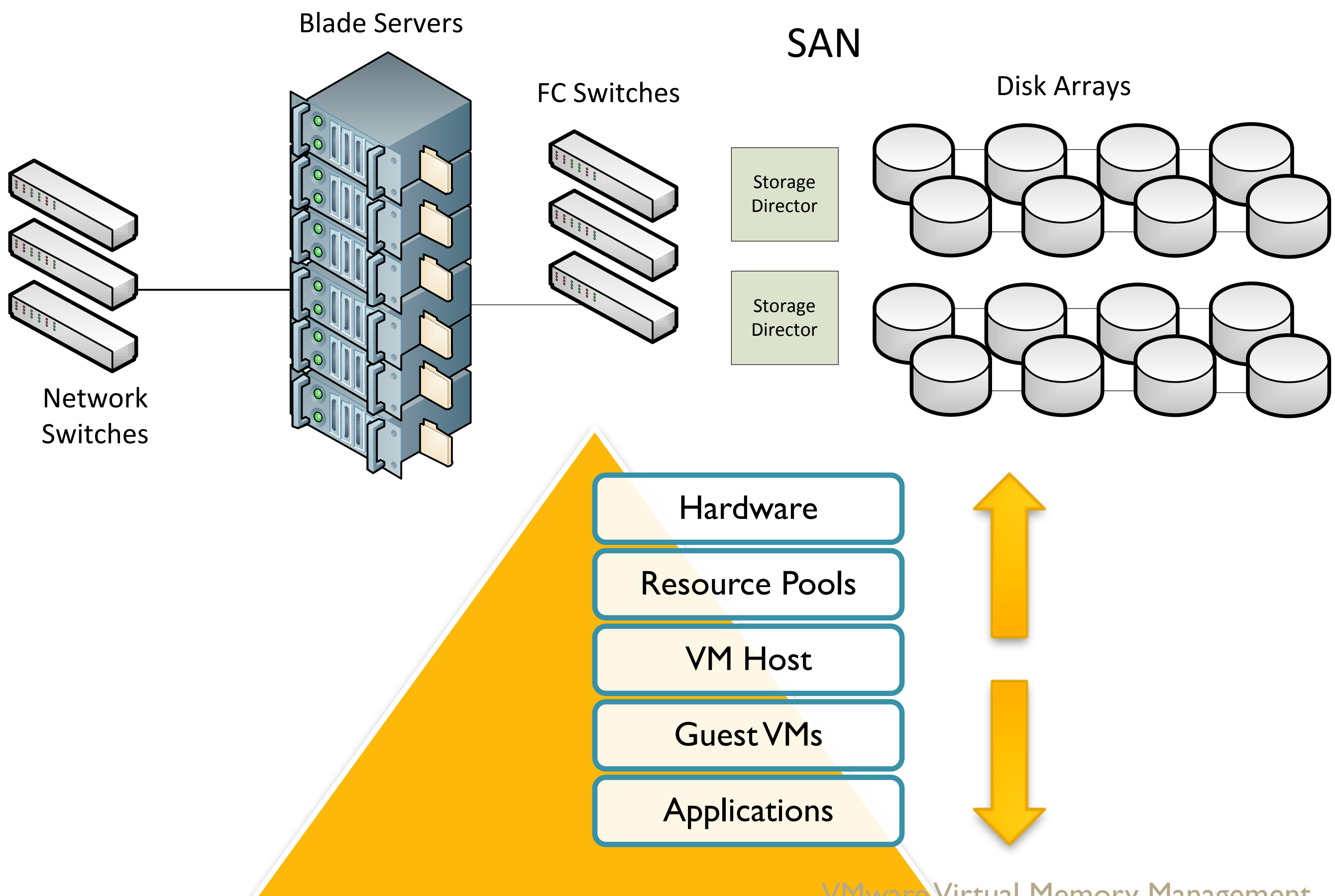
Multiple layers of performance data

- VMware adds two new layers
- Fortunately the remaining layers are (largely) unchanged.
  - But, see impact on Windows clocks!



VMware measurements

- Monitoring the virtualized datacenter requires pooling incomplete and frequently incompatible measurement data from across the datacenter.

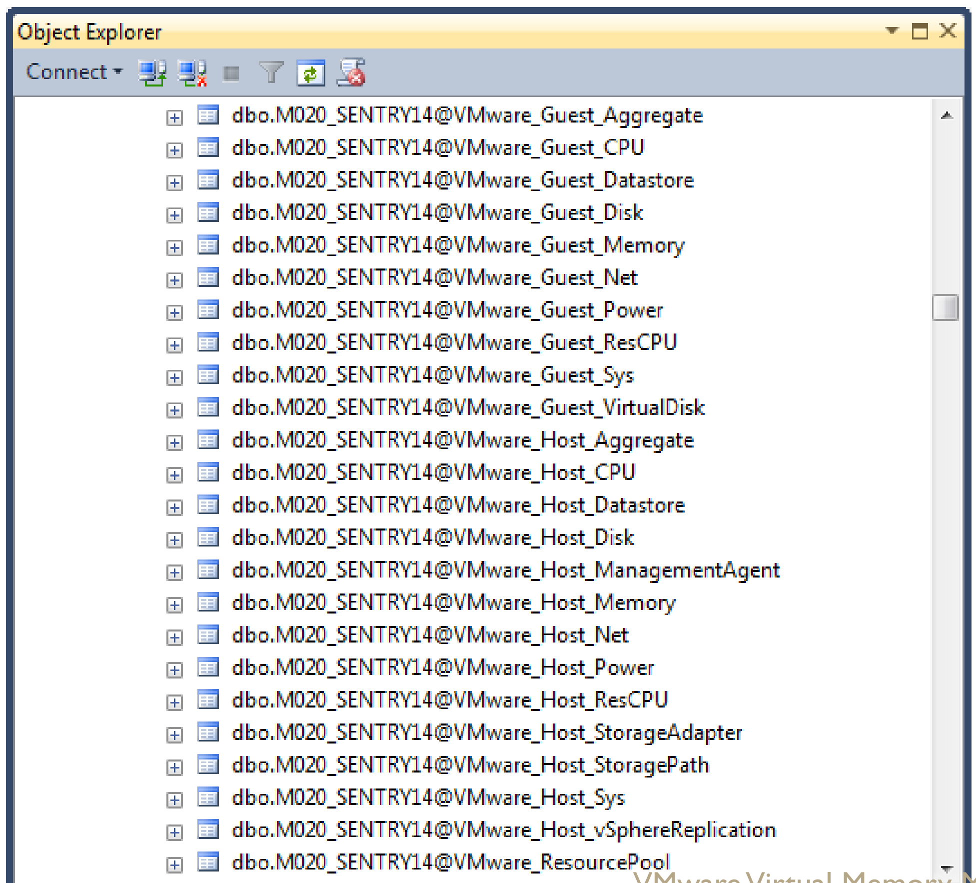


VMware measurements

- Metrics for performance monitoring
  - VMware Host resource consumption
    - CPU
    - RAM
    - Power
    - DataStores and Disks
    - Network
  - Guest machine resource consumption
  - Resource pools
    - data limited to CPU & Memory
  - vMotion actions (directly or indirectly)

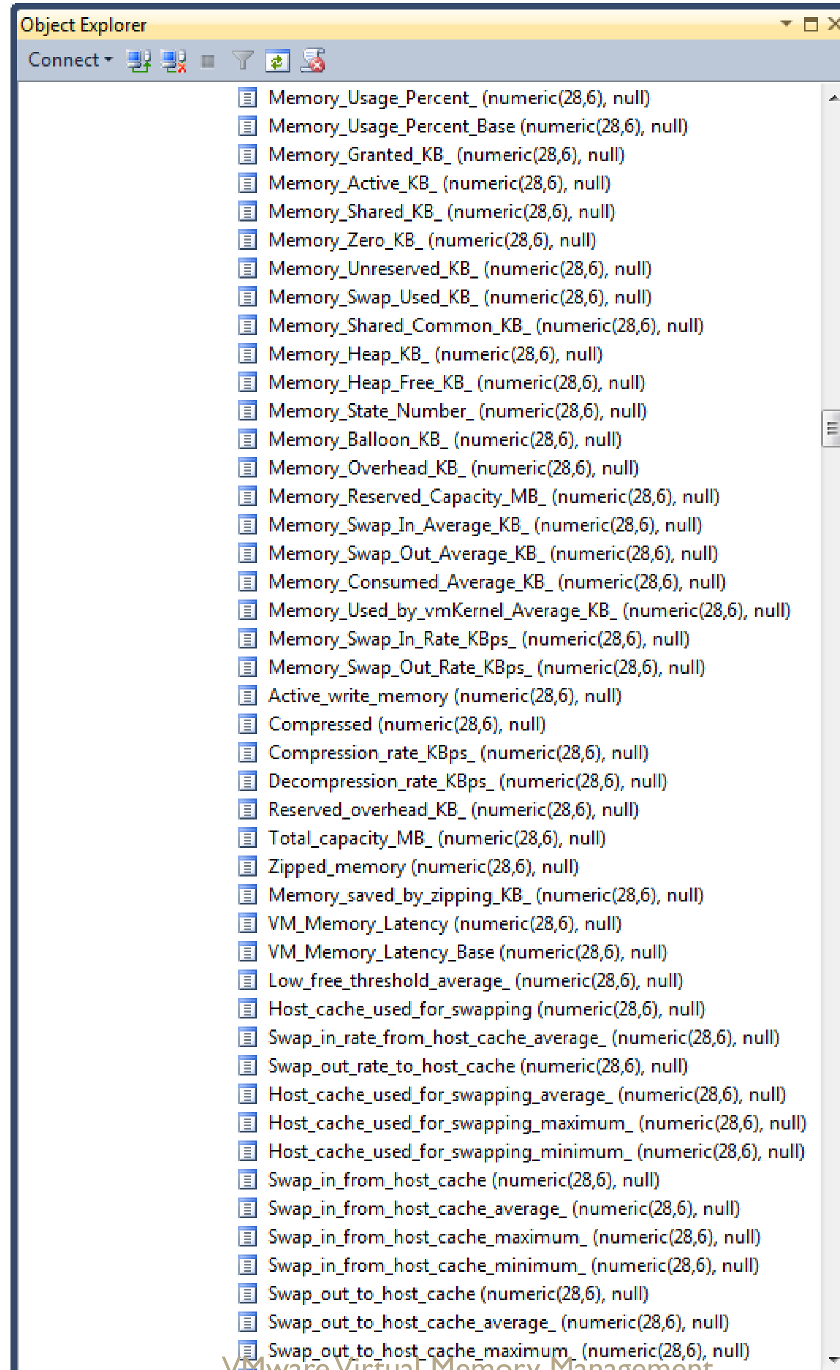
VMware measurements

- Substantial amounts of detailed information on VMware performance and operations available



VMware measurements

- Only scant documentation on many of these data fields



VMware Performance Basics

- Identify over-committed resources
  (CPU, RAM, Disk, Network)

- Available measurements
  - Resource utilization
  - Device latency
  - Device queuing (contention due to over-commitment or under-configuration)

- Augment the VMware data with
  - external SAN and disk hardware measurements
  - measurements from ***inside*** the guest OS
    - Especially, over-commitment of RAM
  - application service level measurements taken from ***outside*** the virtual infrastructure

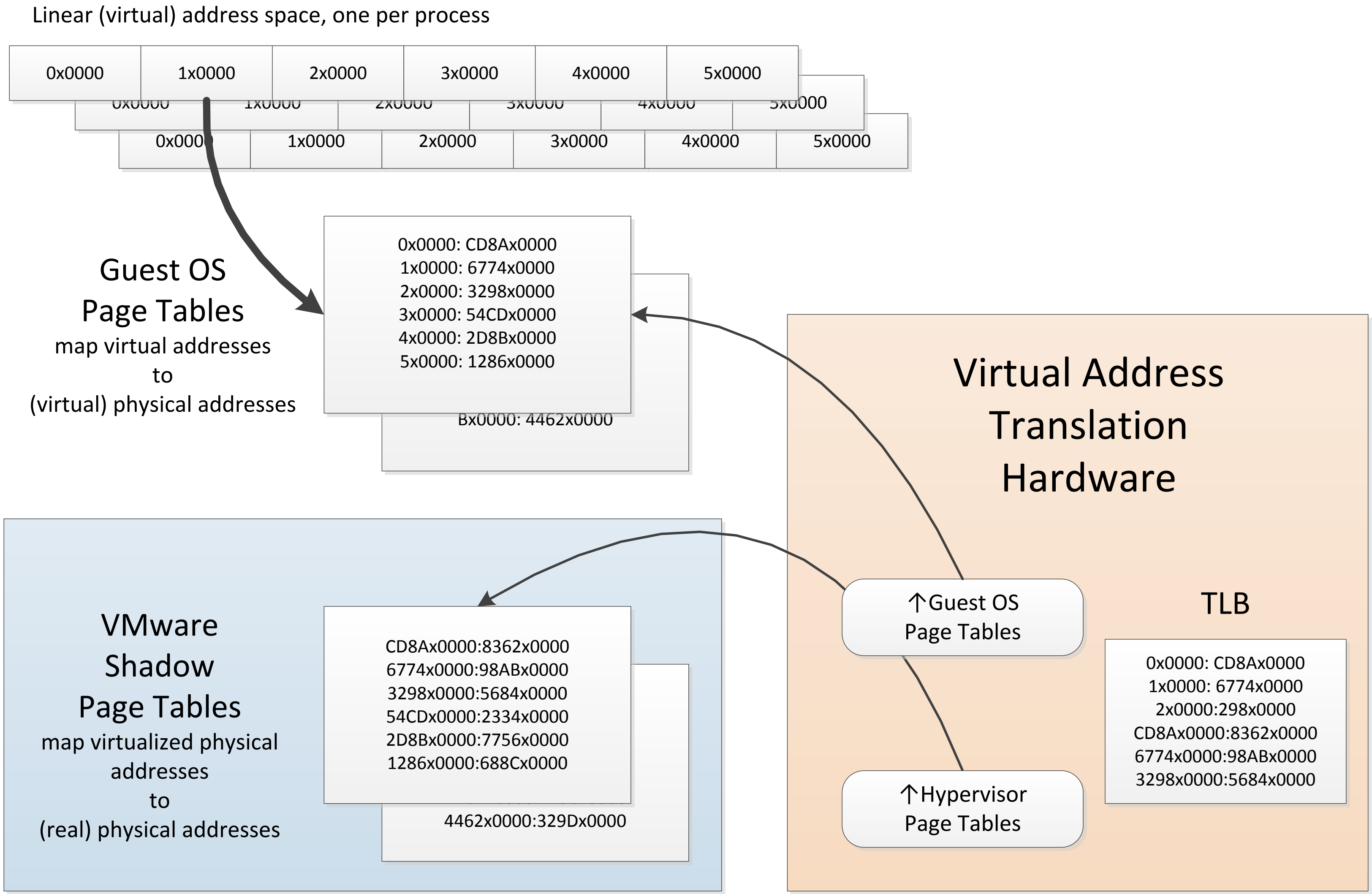
VMware Virtual Memory Management

- Extended Page Tables (hardware supported)
- Page Sharing (when VMs contain identical pages)
  - Significantly reduces memory footprint when homogenous VMs are co-located in the same VM Host

- RAM "over-commitment"
  - Physical RAM is over-committed when
    - Σ VM virtual memory granted > *sizeof* (**RAM**)

- The VM Host cannot see inside the guest OS
  - After RAM is granted to the guest VM, the VMware hypervisor has little understanding of how it is used

Extended Page Tables



VMware Virtual Memory Management

- VMware allocates RAM on behalf of guest machines on demand
  - aka, virtual memory management
- VMware traditionally maintained ‘shadow’ page tables, but relied on the guest OS for page replacement based on the “age” of physical pages
- Hardware began to emerge to lessen the burden of shadow page table maintenance

VMware Virtual Memory Management

- Meanwhile, VMware retained most of its original approach to virtual memory management
- If VMware Host is forced to trim guest OS working set pages, it does so blindly.
  - Candidates for page replacement (VMware swapping) are selected ***randomly***
- **Recommendation:**
  - Watch for Memory State transitions
  - Keep swapping to a minimum

Virtual Memory Management

- VMware uses **"over-commitment"** of RAM to support *aggressive* server consolidation
  - Granting memory *on demand* is much more efficient, compared to static partitioning
    - "virtual memory"
  - Many guest machines are not actively using all the physical RAM that is granted to them

- Physical RAM is over-committed when

$\sum$ VM virtual memory granted > *sizeof* (**RAM**)

- which is typical in virtual memory management

RAM over-commitment

- Over-commitment of Physical RAM when
  - Σ VM virtual memory granted > *sizeof* (**RAM**)

- This works because the Guest OS seldom uses all the virtual memory that it is granted.
- But, beware:
  - Guest VM spin-up
  - Improved LRU in recent versions of Windows
  - Windows processes (e.g., SQL Server) that will allocate vm dynamically up to the limit of physical RAM
  - Limited disk bandwidth available for paging operations
    - Applies to both the VM Host swap files *and* its VM guest machines and their page files

RAM over-commitment

- VMs are presented with a full & consistent physical memory configuration
- All allocation requests for RAM from the virtual machine are handled ***dynamically*** by VMware
- Virtual address translation hardware provides a reliable mechanism for detecting what RAM the guest VM actually accessed
  - **Note:** the VMware Host ultimately resolves **<u>all</u>** page faults
  - But, it has ***a very limited understanding*** of actual page usage patterns, based on the initial page access only
- ***Memory Granted***: RAM allocated to the guest vm on demand

RAM over-commitment

- Case Study
  - ESX Server: a standalone Dell Optiplex 990 with an Intel i7 quad-core processor and 16GB of RAM
    - Hyper-Threading is disabled through the BIOS).
  - 4 X Windows Server 2012 Guests defined at 8 GB
    - Running identical test workloads; a .NET Framework app that allocates and exercises large amounts of virtual memory
  - See how VMware responds to a workload where

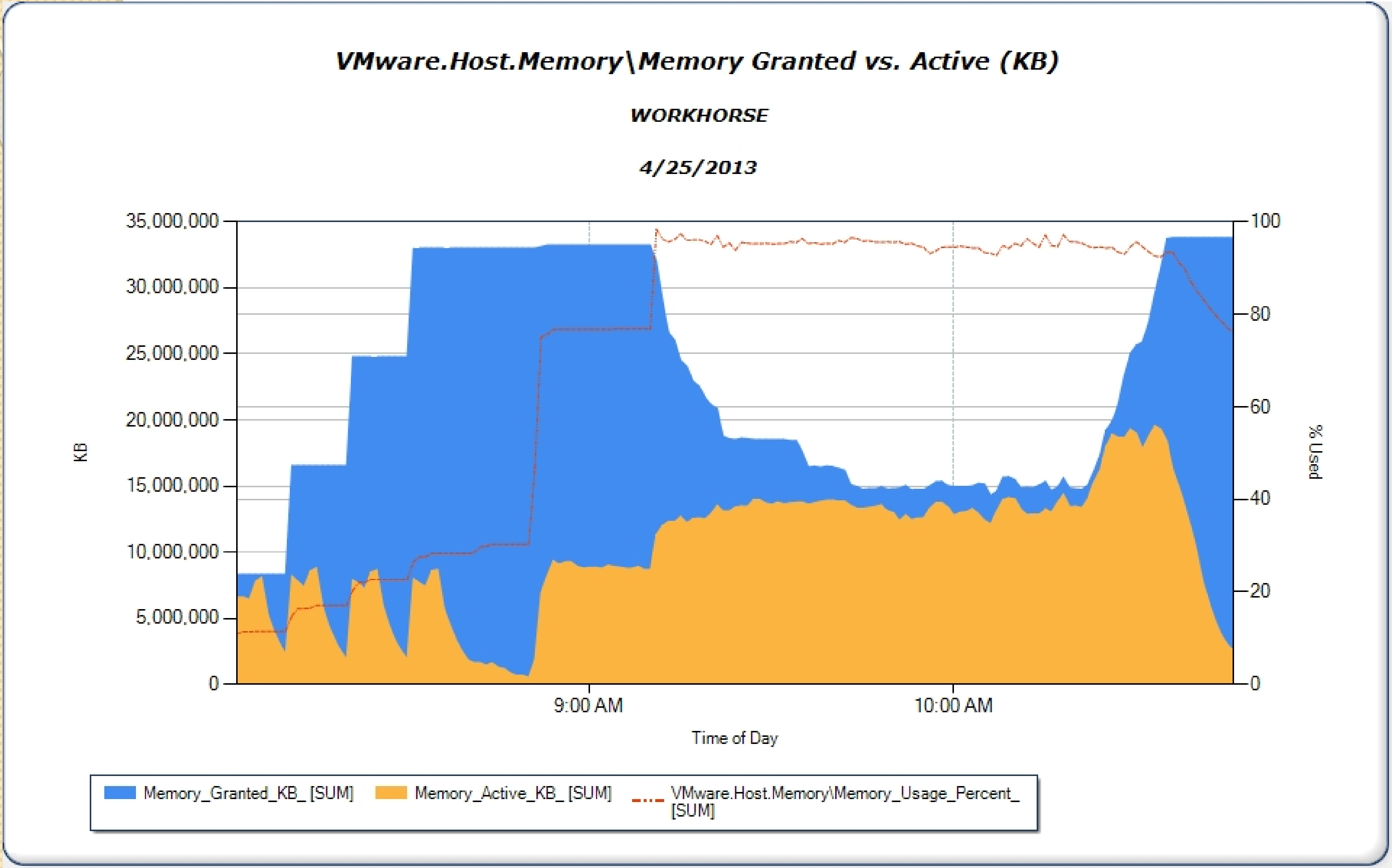
$$\Sigma \; \textit{\textbf{Memory Granted}} = 2 * \textit{\textbf{sizeof(RAM)}}$$

RAM over-commitment

- ***Page replacement is problematic because no comparable hardware mechanism exists for detecting what RAM the guest VM has freed!***

- ***Memory Active***: RAM the guest vm has referenced during the last measurement interval
  - Estimated based on sampling:
    - Each interval, flag some percentage of vm guest pages as ***invalid***
    - Use the (soft) page fault rate to estimate the vm working set

RAM over-commitment



RAM over-commitment

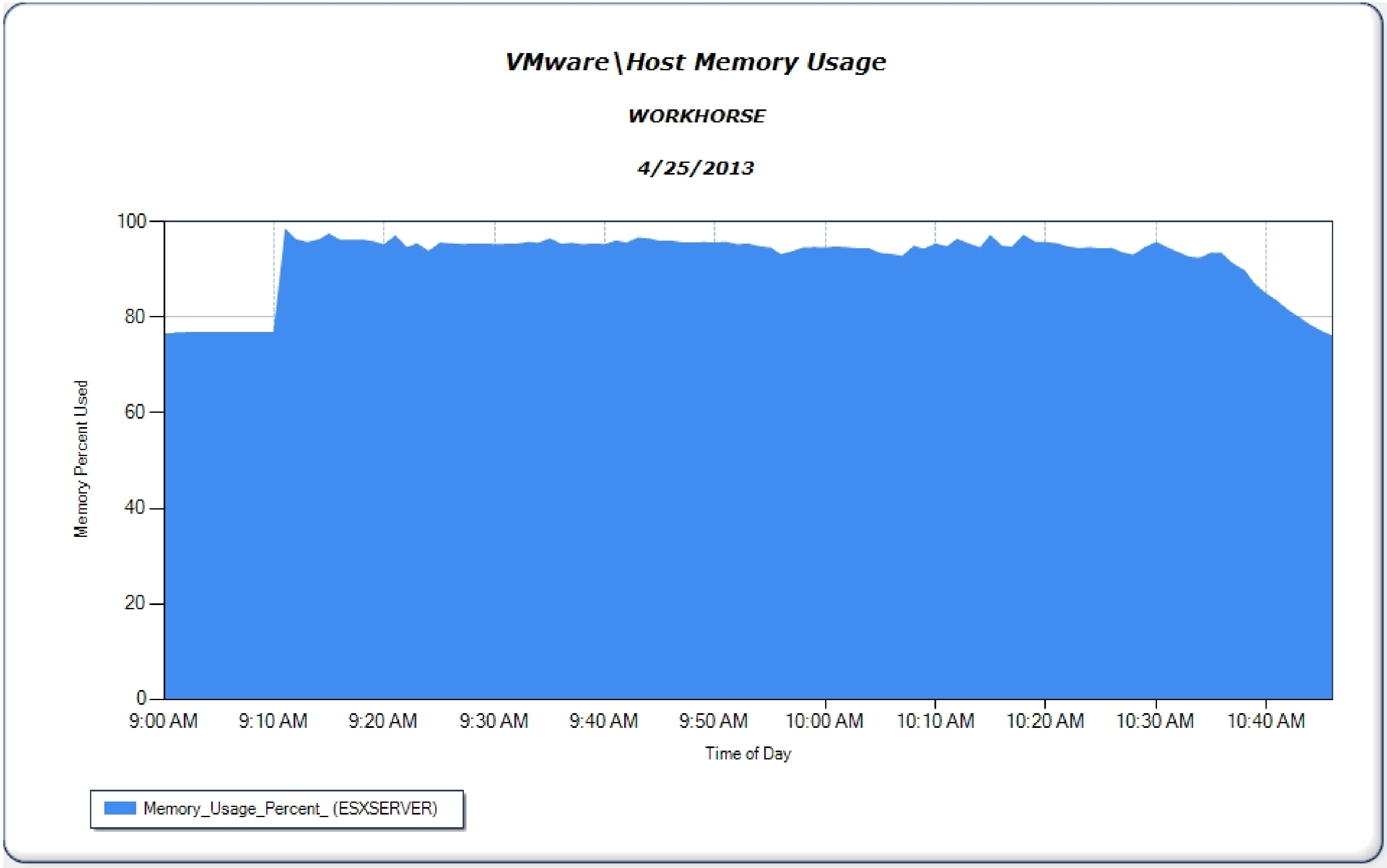
- While ***Memory Active*** < sizeof(RAM)
  - VMware balances RAM usage across guest machines
  - subject to resource allocation policies set by the customer
    - Shares
    - Reservation: sets a ***lower*** limit on the size of the guest vm working set
    - Limit: sets an ***upper*** limit on the size of the guest vm working set

RAM over-commitment

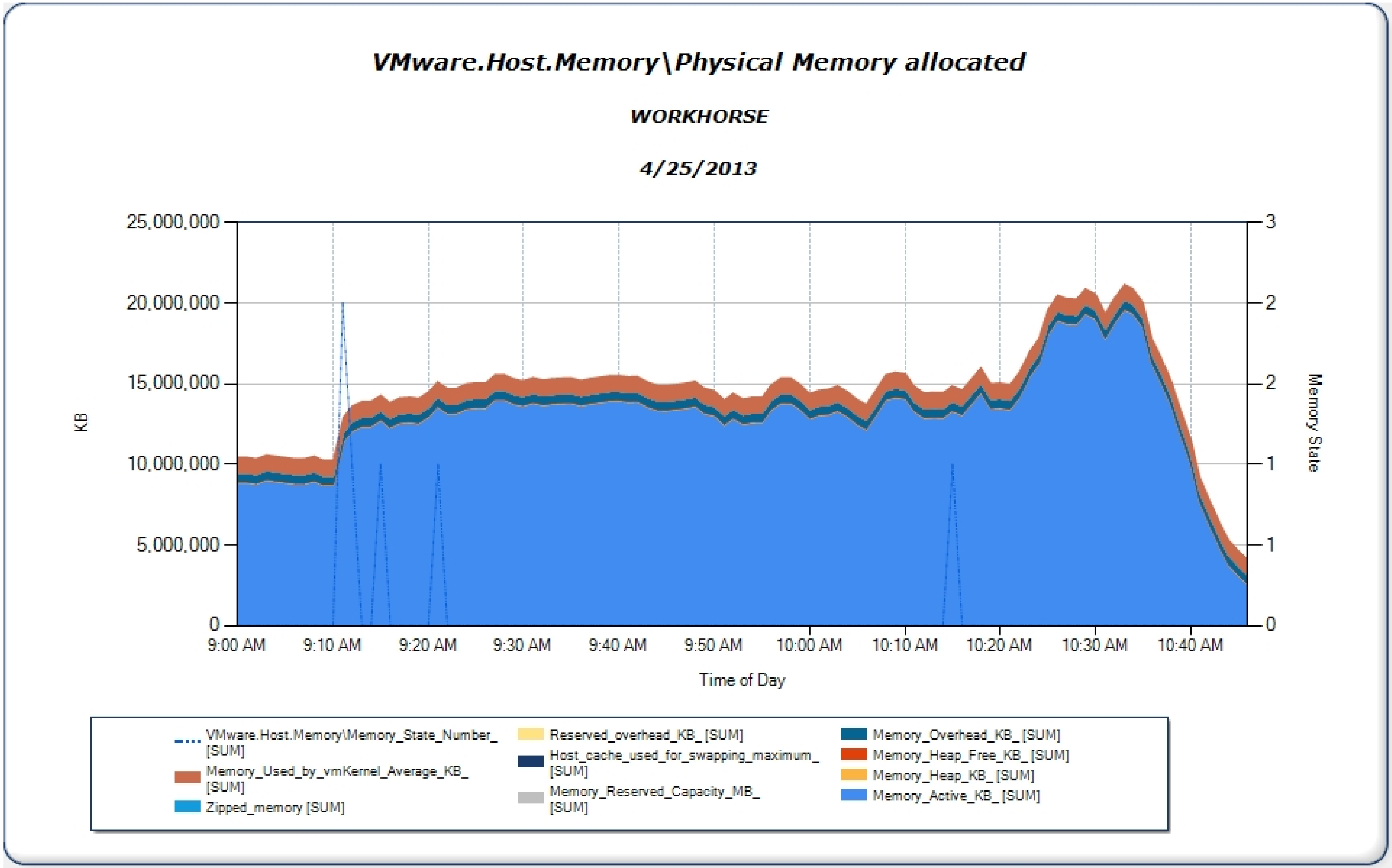
- VMware Page replacement policy

- But when ***Memory Active*** → *sizeof*(RAM)
  - First, "ballooning" attempts to induce page replacement in the virtual guest
  - If ballooning is not sufficient to relieve the memory contention, VMware then resorts to more drastic means
    - swapping, which selects swap candidates from the guest working set ***at random***
      - dedicated swap files are allocated per machine

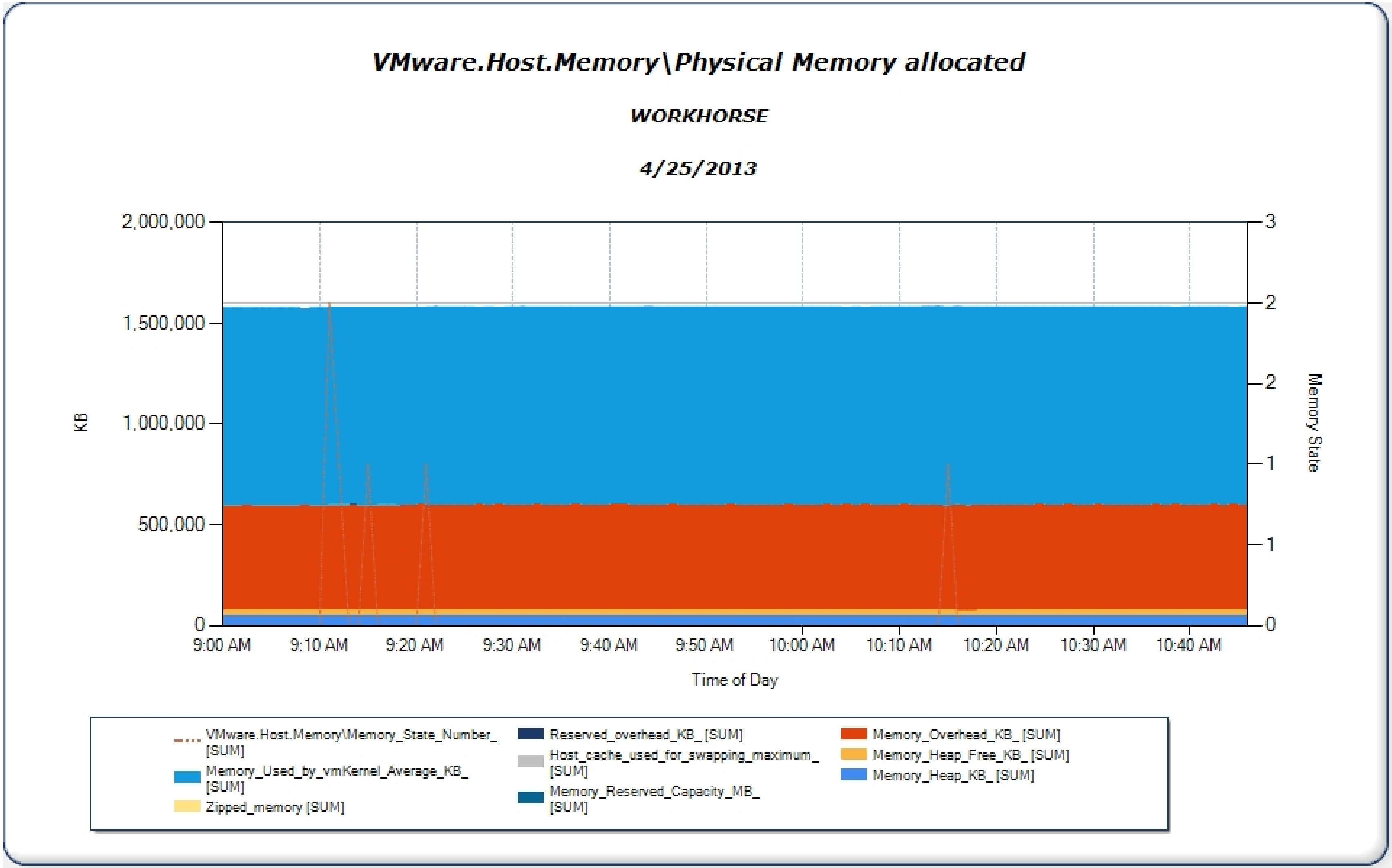
RAM over-commitment



RAM over-commitment



RAM over-commitment



VMware Performance Basics: RAM

- Memory Overhead (per MB of Guest machine VM mapped)

vCPUs	1	2	4	8
256	20	24	32	48
1024	26	30	38	53
4096	49	53	61	77
16384	140	144	152	169

- Scalability ≈ 1:100 (> = 4 GB)

see http://pubs.vmware.com/vsphere-50/index.jsp

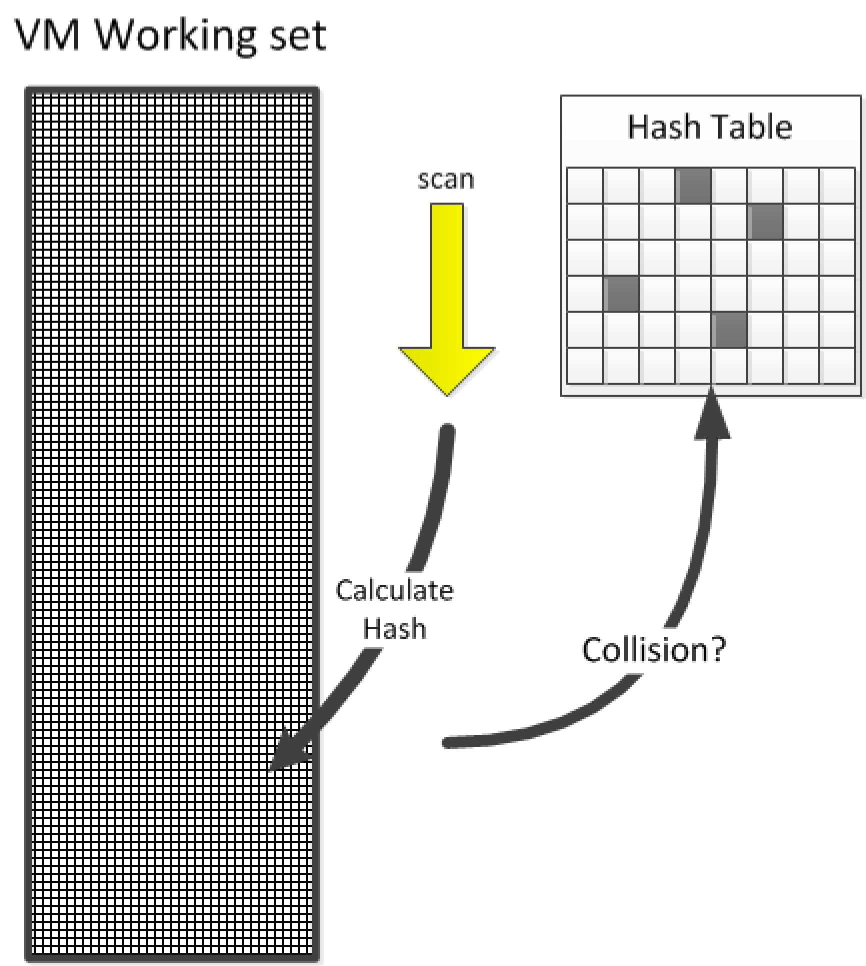
RAM over-commitment

- ***Transparent memory sharing***
- supports aggressive over-commitment
- many opportunities to share pages
  - whenever guest VMs are running the same OS, or the same application code
  - shared data pages are also possible
  - VMware scans pages continuously, looking for sharing candidates
  - a ***Copy on Write*** mechanism exists whenever a shared page is changed and can no longer be shared

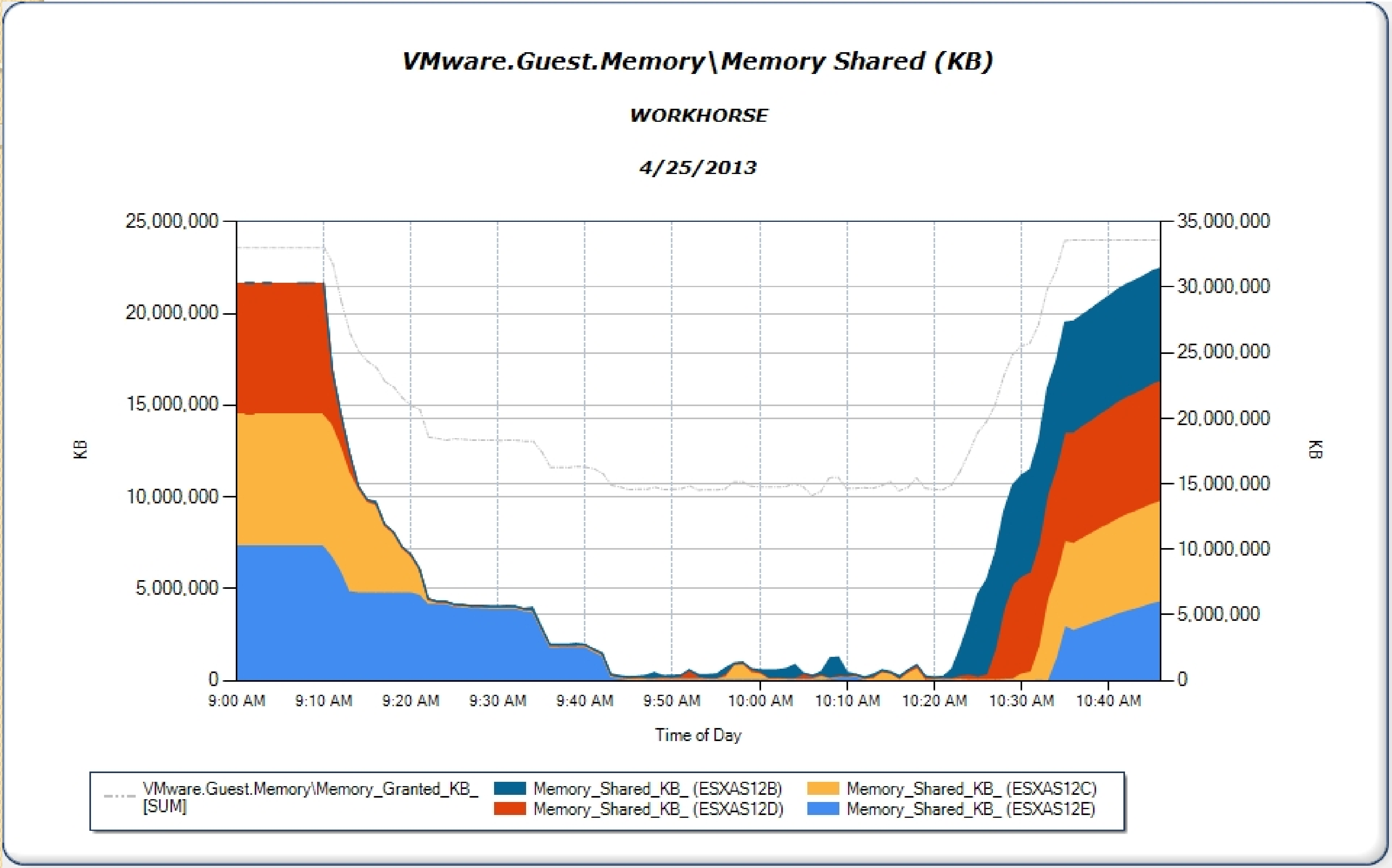
Memory sharing

- Computes a hash from the contents of a page
- Checks for a collision in a Hash table that contains the hash codes for all active pages
- If found, compare the two pages byte by byte
- If equal, share the page!

- Note: tuning parameters exist for controlling the rate at which the background thread that scans memory runs



Memory Sharing



Memory sharing

- Costs associated with memory sharing
  - Cost of memory scanning
    - Runs as a background thread
    - Note: tuning parameters exist for controlling the rate at which memory is scanned
  - Cost of additional page fault handling
    - in shadow page tables, shared pages are flagged as *Read Only*
    - 1st Write to a shared page generates a page fault
    - to handle the page fault, the updated, shared page must be copied somewhere else in physical RAM

RAM Allocation metrics

- Memory Granted
  - Guest VM virtual memory mapped to VM Host virtual memory
- Memory Usage %
  - Guest VM: Active Memory / Memory Granted
  - Host: Active Memory / Host machine physical memory size
- Memory Shared
  - Guest physical memory that is duplicated across other virtual machines and thus eligible to be shared
    - Double counted for each VM where the identical pages reside
  - includes Memory Zero pages (the Zero list inside Windows)
- Memory Overhead
  - VMware must build and maintain Shadow Page Tables for each VM

Page replacement in VMware

- Triggered by thresholds
  - ***Ballooning***
    - induce the VMware guest OS to perform page replacement
  - ***Swapping***
    - random page replacement
    - immediate relief
  - ***Compression***:
    - stolen pages written to the Compression Cache
  - ***Idle Machine Tax***
    - an LRU-like mechanism directs increased paging stealing from inactive guest machines

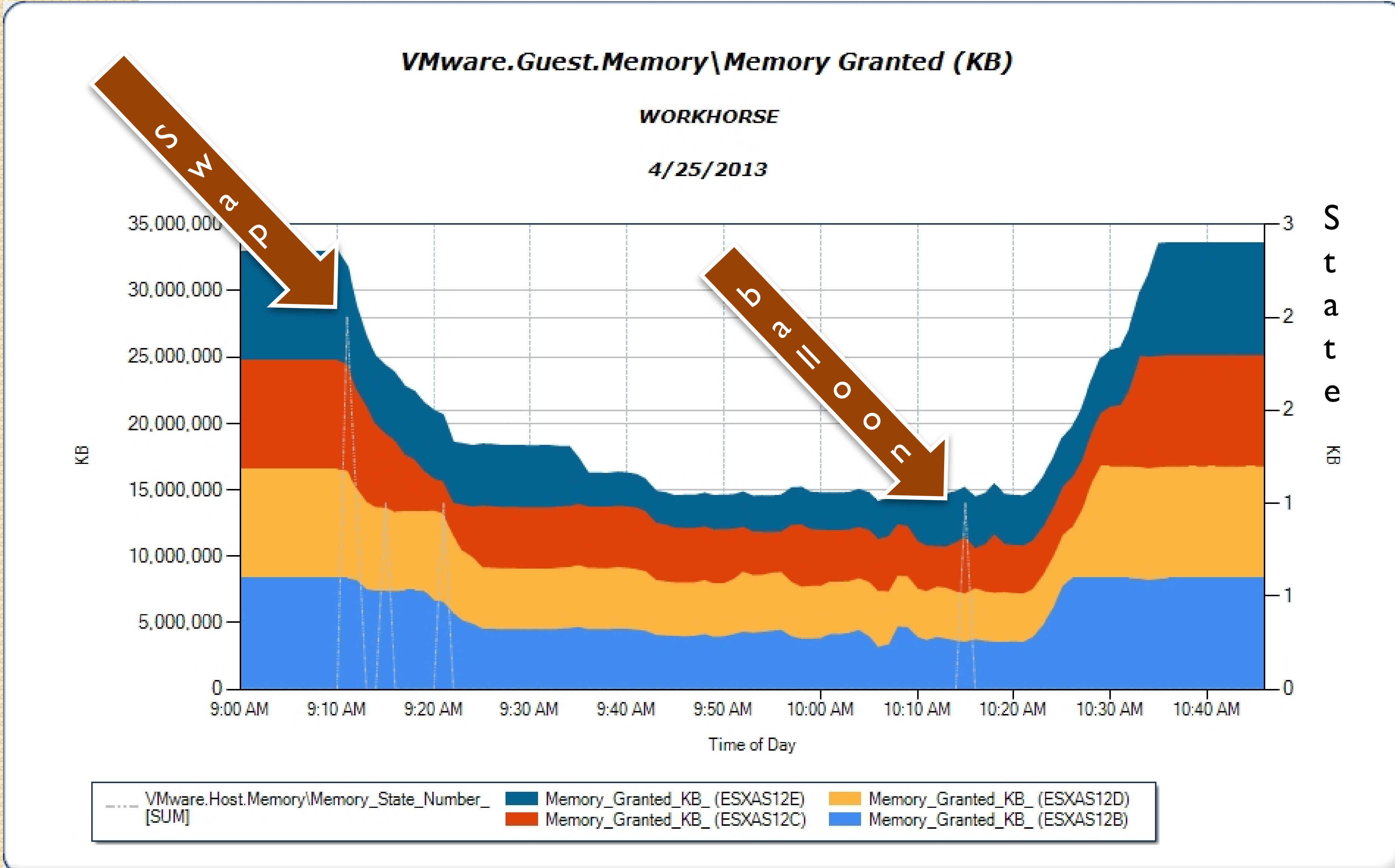
Page replacement in VMware

- Memory State

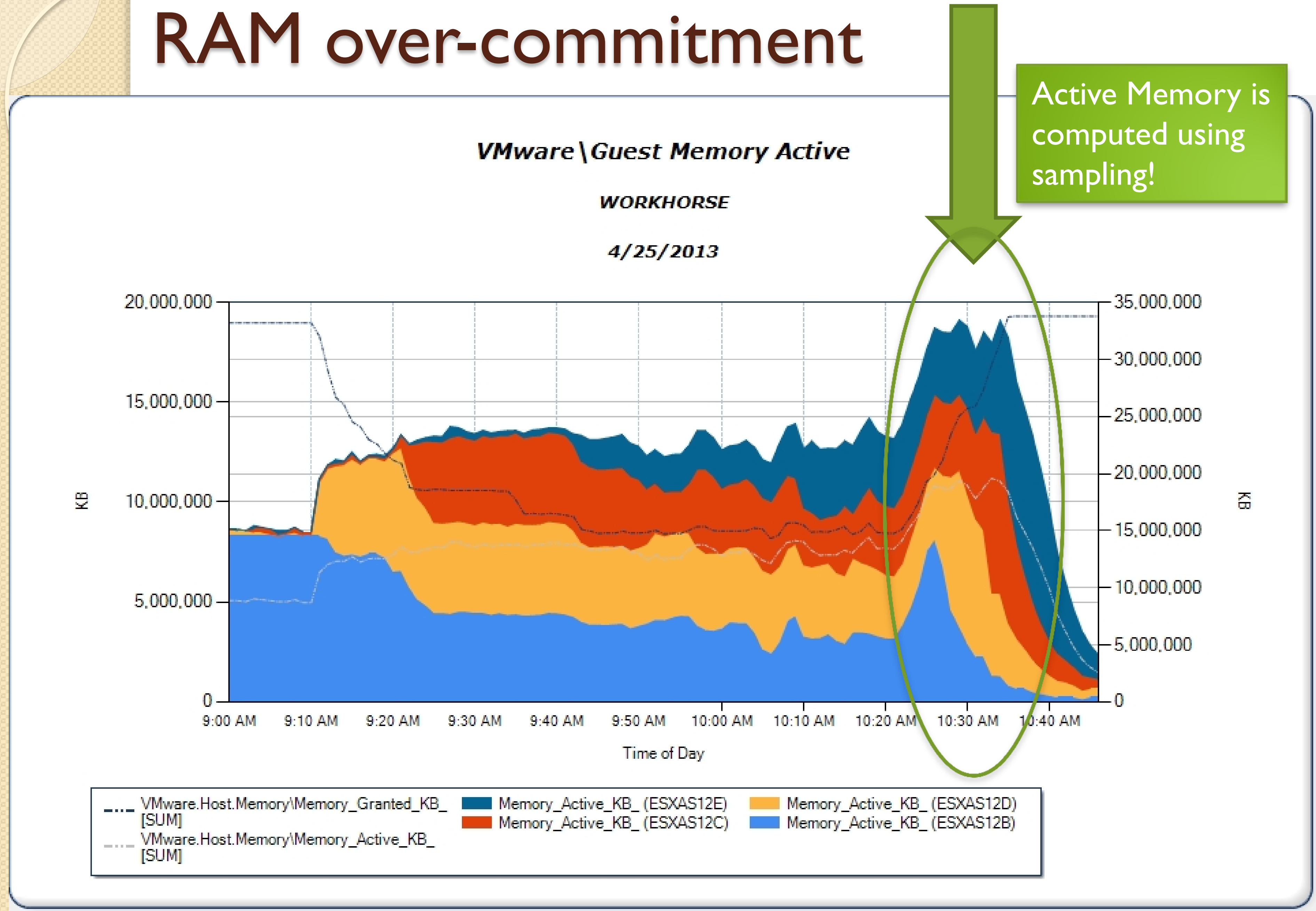
State	No.	Condition	Reclamation Action
High	**0**	Free memory >= 6% of machine memory	None
Soft	**1**	Free memory < 6%	Ballooning
Hard	**2**	Free memory < 4%	Pages Compressed or Swapped to Disk*
Low	**3**	Free memory < 2%	Blocks execution of active VMs > target allocations*

* **Note:** CPU Swap time can be accumulated due to swapping pages to disk

Memory State

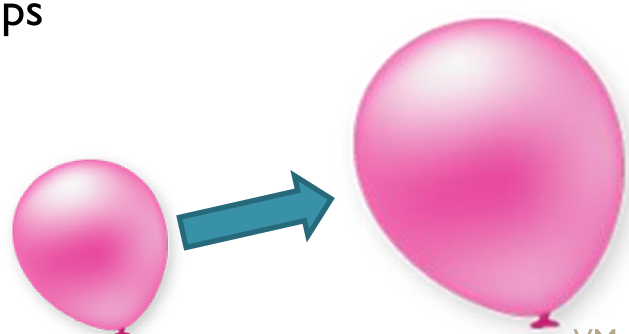


RAM over-commitment



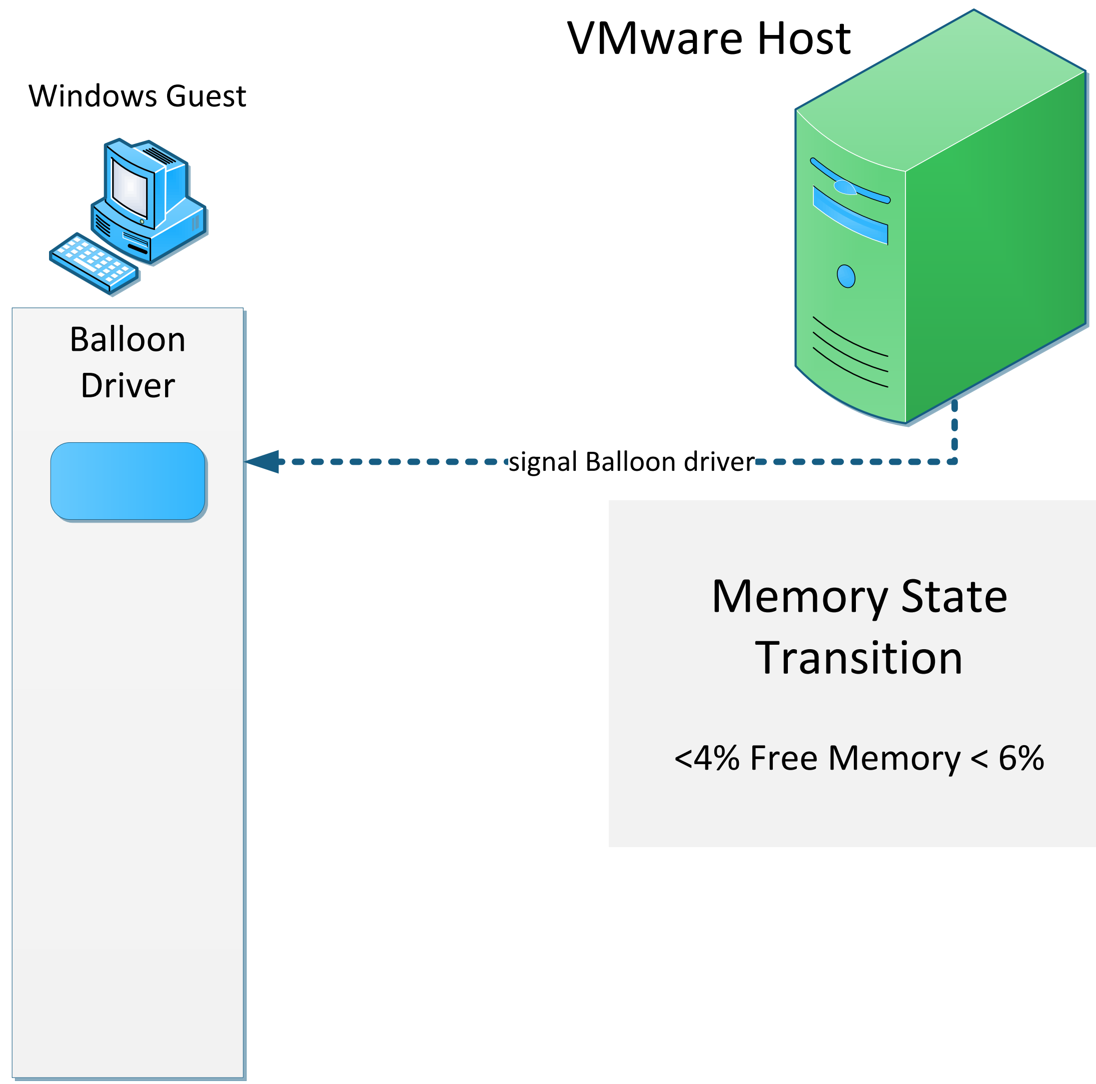
Ballooning

- When RAM is over-committed, "ballooning" tries to force guest VMs to utilize LRU to trim their pages
  - Only the Guest VM understands its memory usage patterns and can utilize LRU effectively
  - Available Bytes + Standby Cache Bytes in Windows
  - Well-behaved apps "listen" for a Windows VMM "low memory" event to release excess pages
    - e.g.,
      - SQL Server, the Exchange Store process, and .NET Framework apps

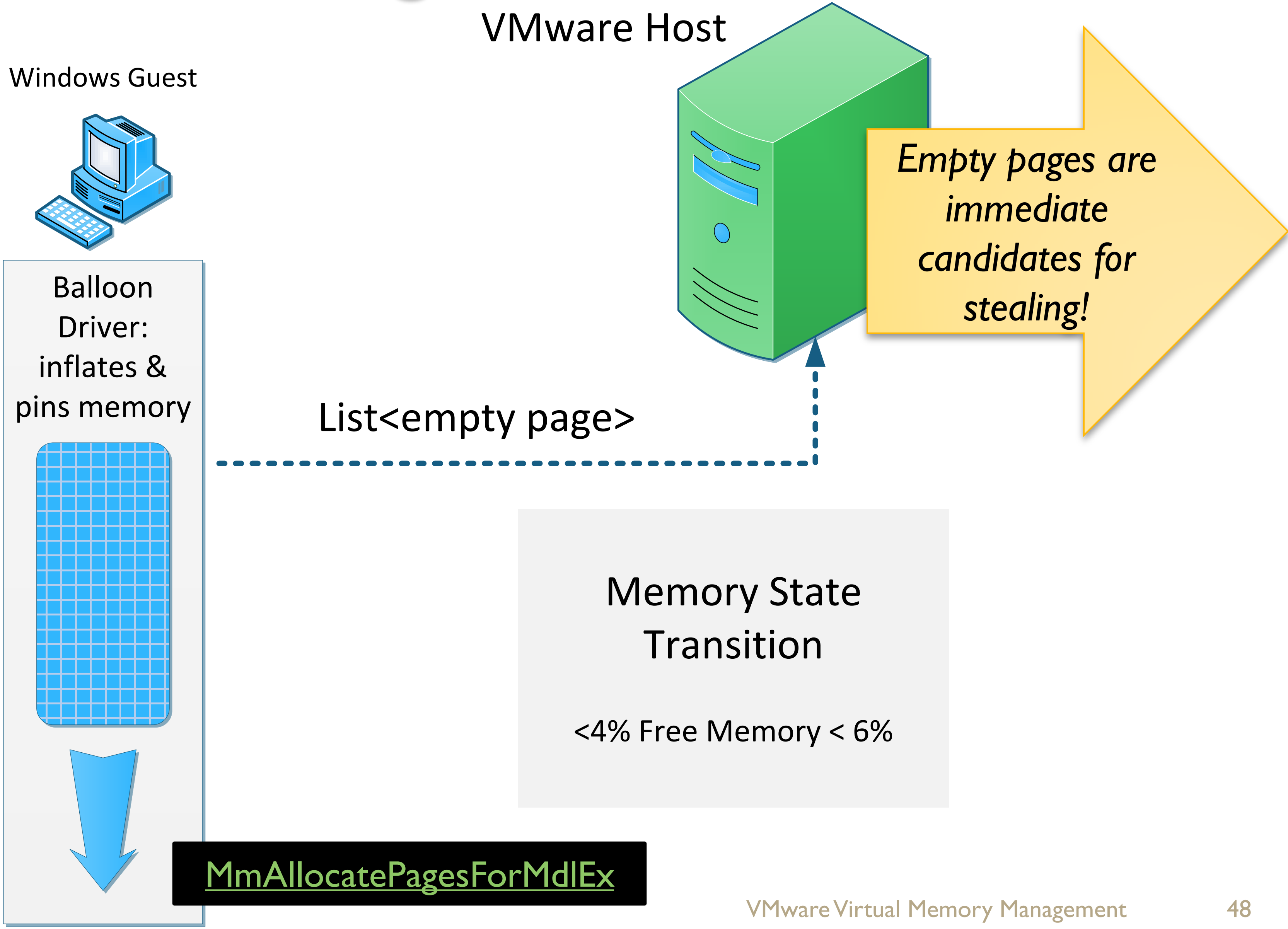


Ballooning

- When memory is under pressure, attempt to induce page stealing in the guest OS
- Notify a Balloon driver via a back channel inside the guest VM to “inflate”



Ballooning

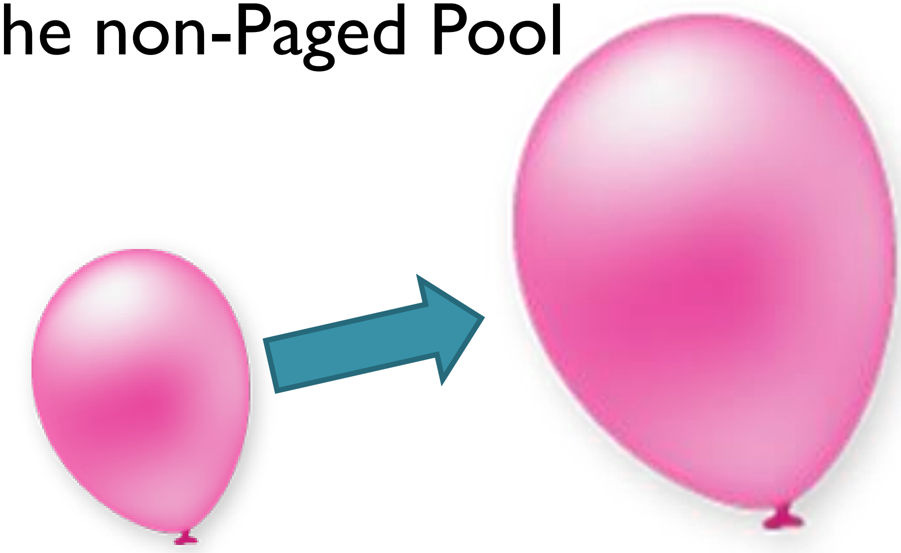


VMware Host
Windows Guest
Balloon Driver: inflates & pins memory
List<empty page>
Empty pages are immediate candidates for stealing!
Memory State Transition
<4% Free Memory < 6%
MmAllocatePagesForMdlEx

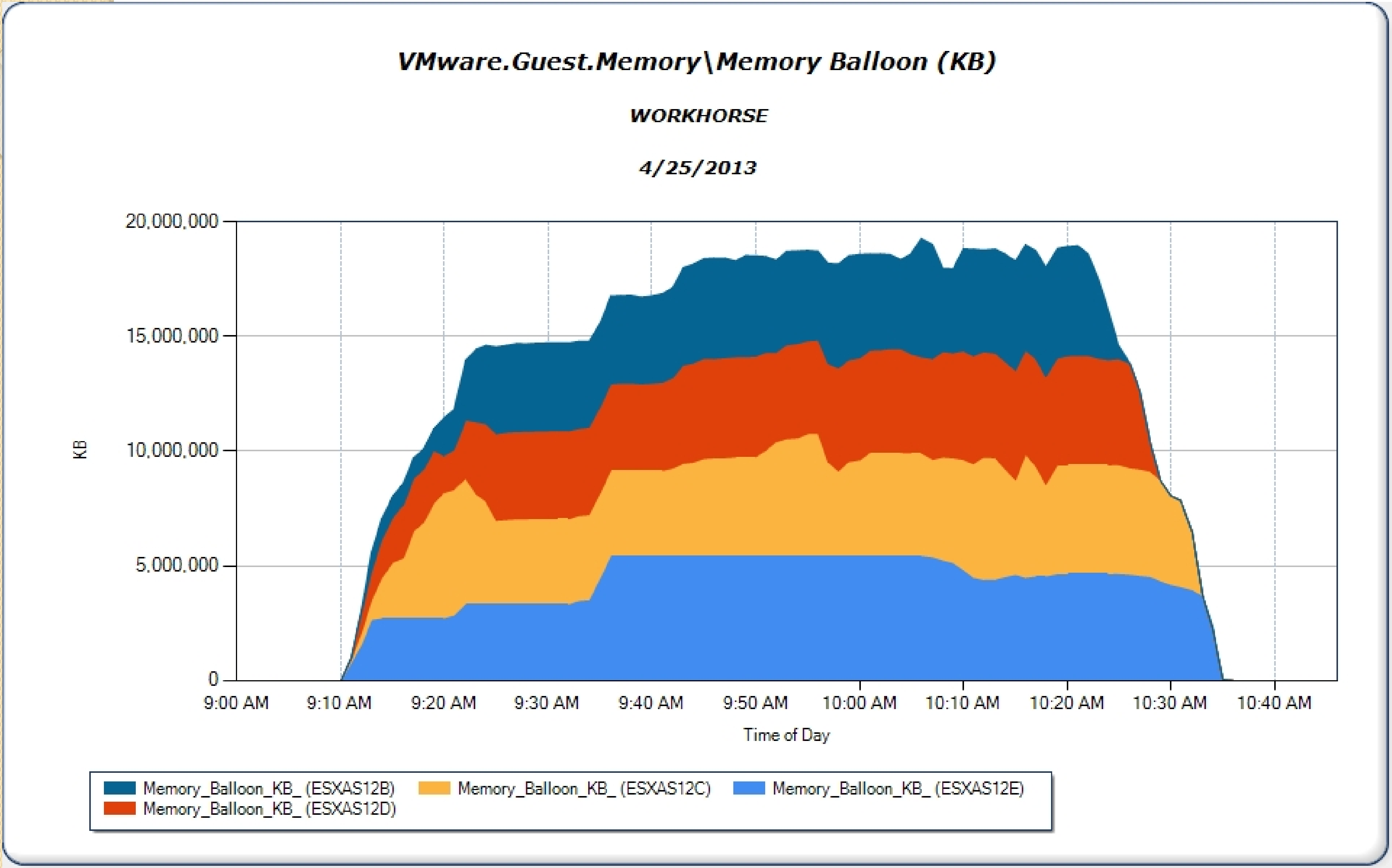
Ballooning

- vmmemctl.sys is the VMware balloon driver inside Windows that "inflates"
  - MmAllocatePagesForMdlEx
  - MmProbeAndLockPages

- Allocates from the non-Paged Pool

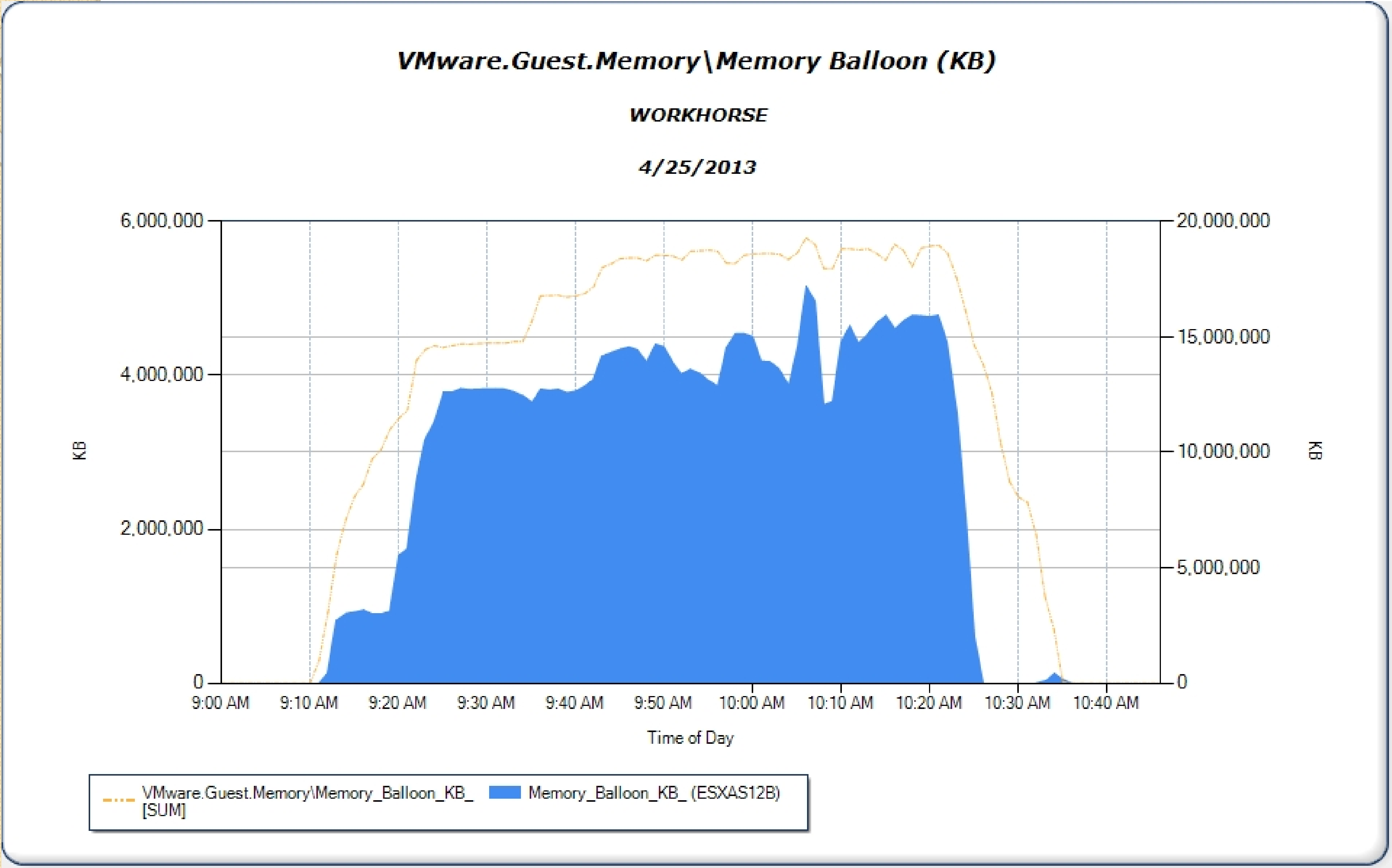


Ballooning



VMware.Guest.Memory\Memory Balloon (KB)
WORKHORSE
4/25/2013
20,000,000
15,000,000
10,000,000
5,000,000
0
KB
9:00 AM
9:10 AM
9:20 AM
9:30 AM
9:40 AM
9:50 AM
10:00 AM
10:10 AM
10:20 AM
10:30 AM
10:40 AM
Time of Day
Memory_Balloon_KB_ (ESXAS12B)
Memory_Balloon_KB_ (ESXAS12C)
Memory_Balloon_KB_ (ESXAS12E)
Memory_Balloon_KB_ (ESXAS12D)

Ballooning



VMware.Guest.Memory\Memory Balloon (KB)
WORKHORSE
4/25/2013
6,000,000
4,000,000
2,000,000
0
KB
20,000,000
15,000,000
10,000,000
5,000,000
0
KB
9:00 AM
9:10 AM
9:20 AM
9:30 AM
9:40 AM
9:50 AM
10:00 AM
10:10 AM
10:20 AM
10:30 AM
10:40 AM
Time of Day
VMware.Guest.Memory\Memory_Balloon_KB_ [SUM]
Memory_Balloon_KB_ (ESXAS12B)

Ballooning

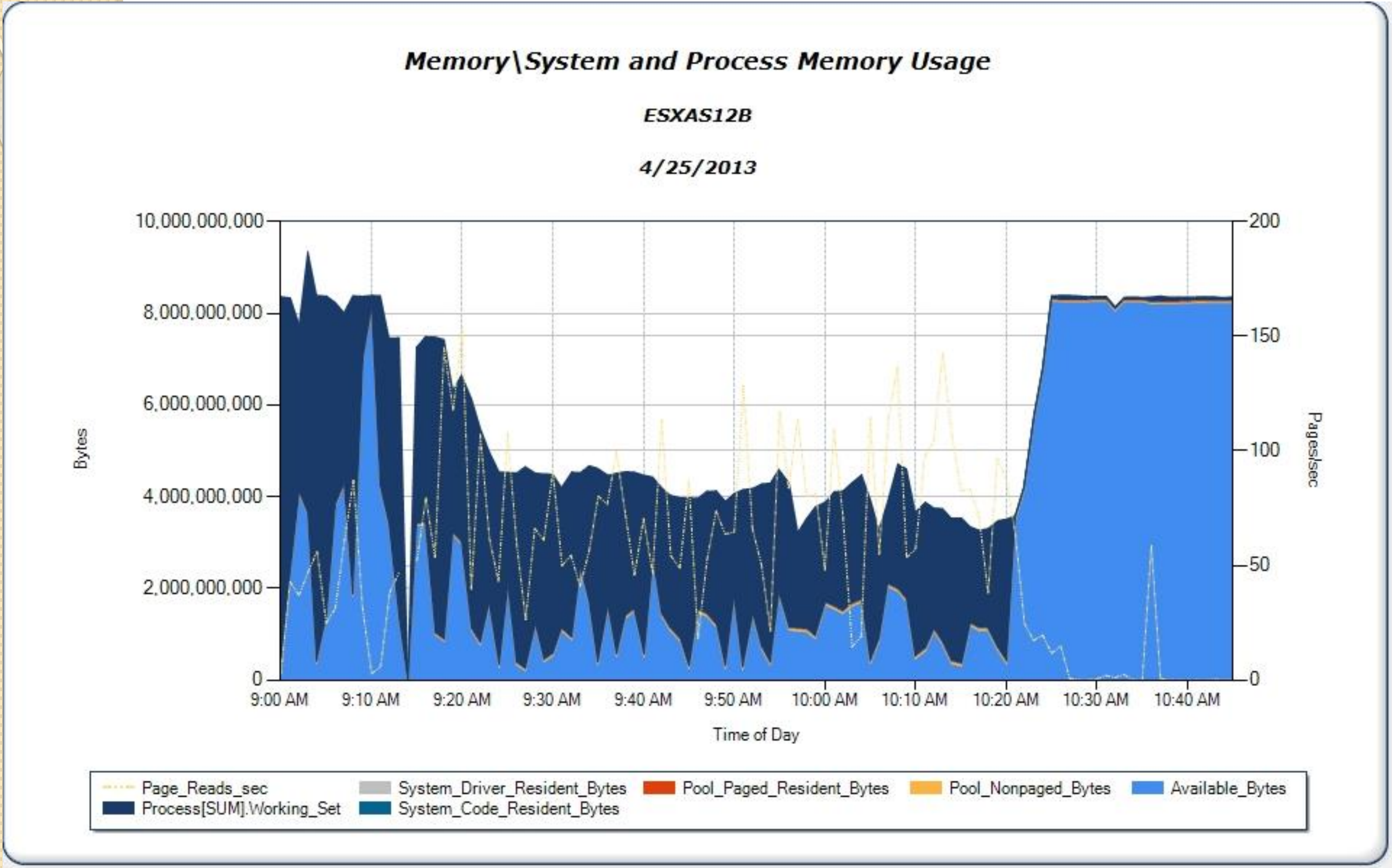
- When the Balloon driver "inflates," it allocates pages from the non-Paged pool.
  - These empty pages can be "stolen" immediately by the VMware Host
  - **Note:** the size of the non-Paged pool is very restricted in earlier versions of 32-bit Windows
    - Windows also currently supports pinning normally allocated pages in memory
- When the Balloon driver "inflates," it may drive paging, within the guesy OS, due to increased memory contention

Ballooning

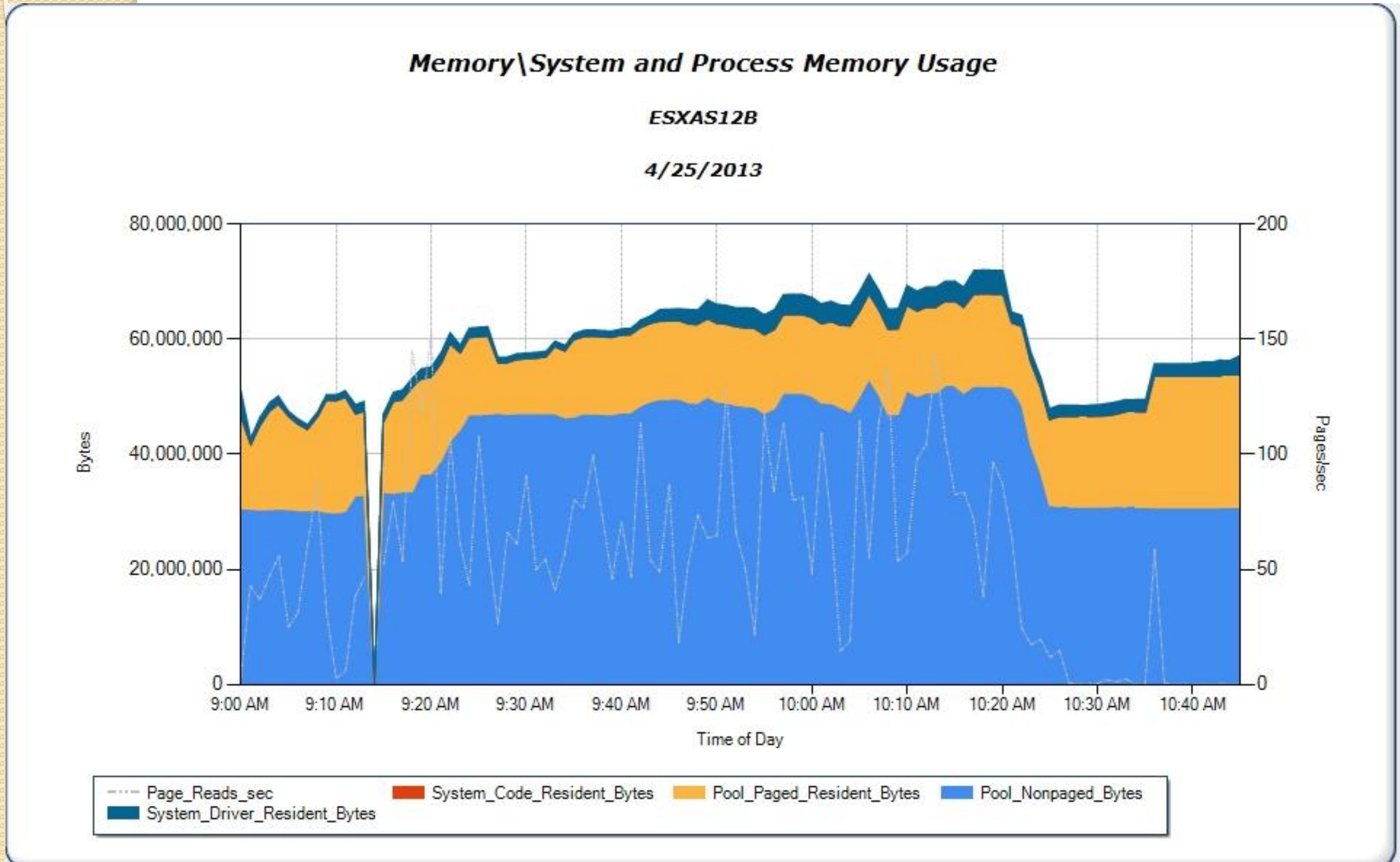
- When the Balloon driver "inflates," it may drive paging due to memory contention

- Telltale signs inside the guest OS:
  - spike in size of the non-Paged Pool
  - spike in Demand Zero paging
  - increase in overall paging
  - applications that "listen" for low memory notifications will attempt to release older pages
    - SQL Server buffers
    - .NET Framework garbage collection (GC)

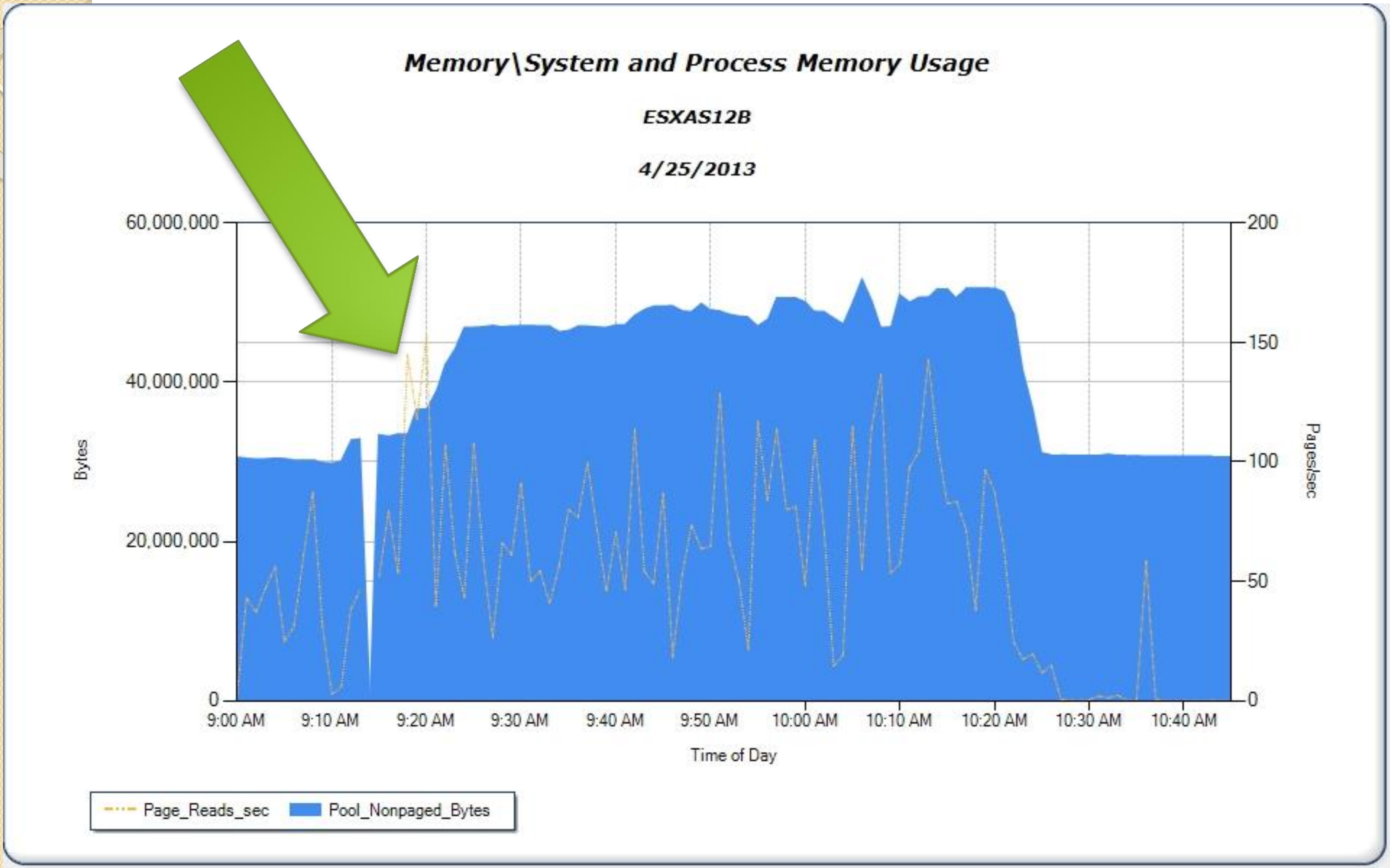
Ballooning: the view from inside



Ballooning: the view from inside



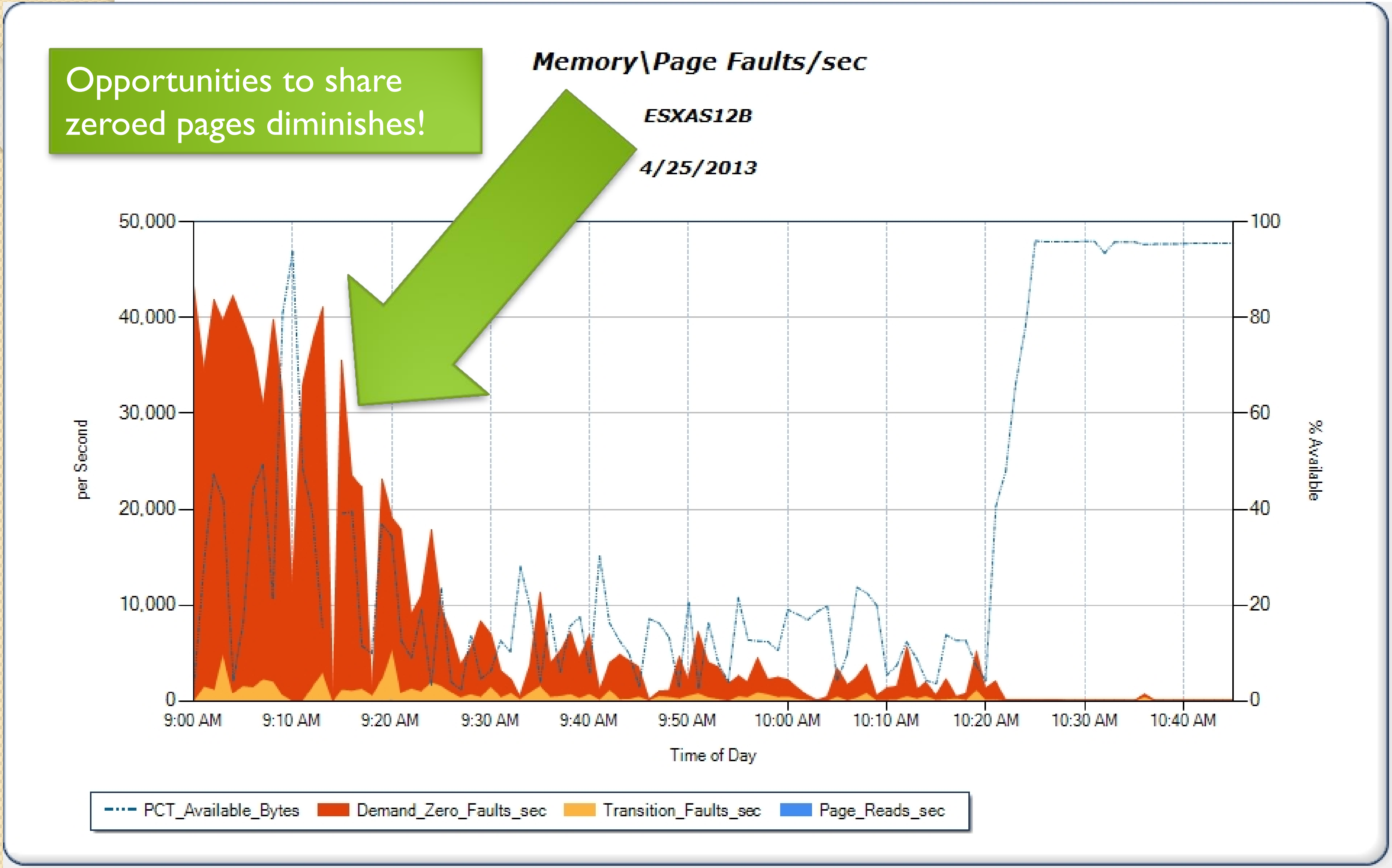
Ballooning: the view from inside



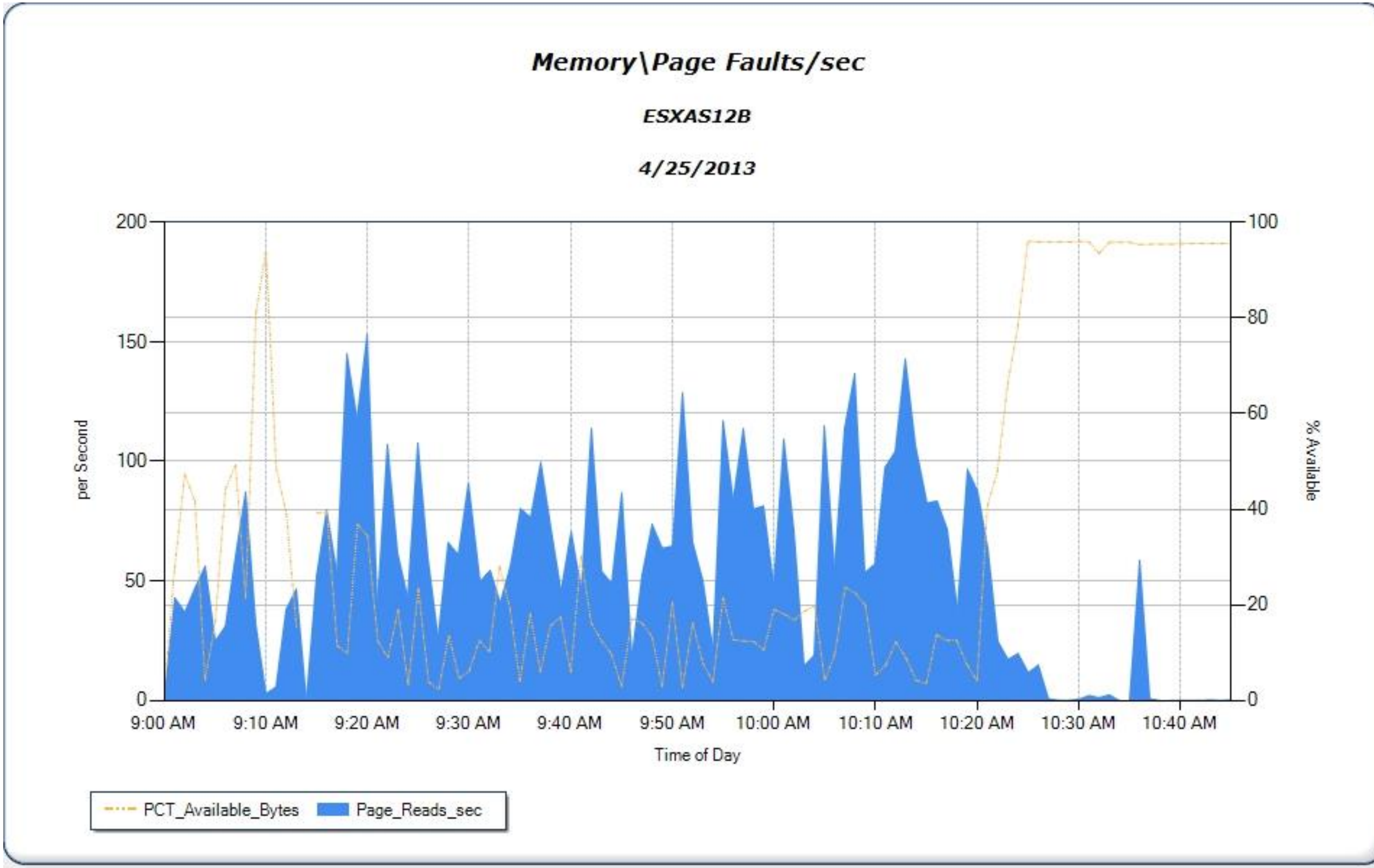
Ballooning

- Telltale signs of ballooning from inside the guest OS:
  - the size of the non-Paged Pool spikes!
    - but not near as large as the VMware counters suggest
  - spike in Demand Zero paging?
  - increase in overall paging to disk?
    - limited by the speed of the paging file (virtual disk)
  - applications that "listen" for low memory notifications will attempt to release older pages
    - SQL Server buffers
    - .NET Framework garbage collection (GC)

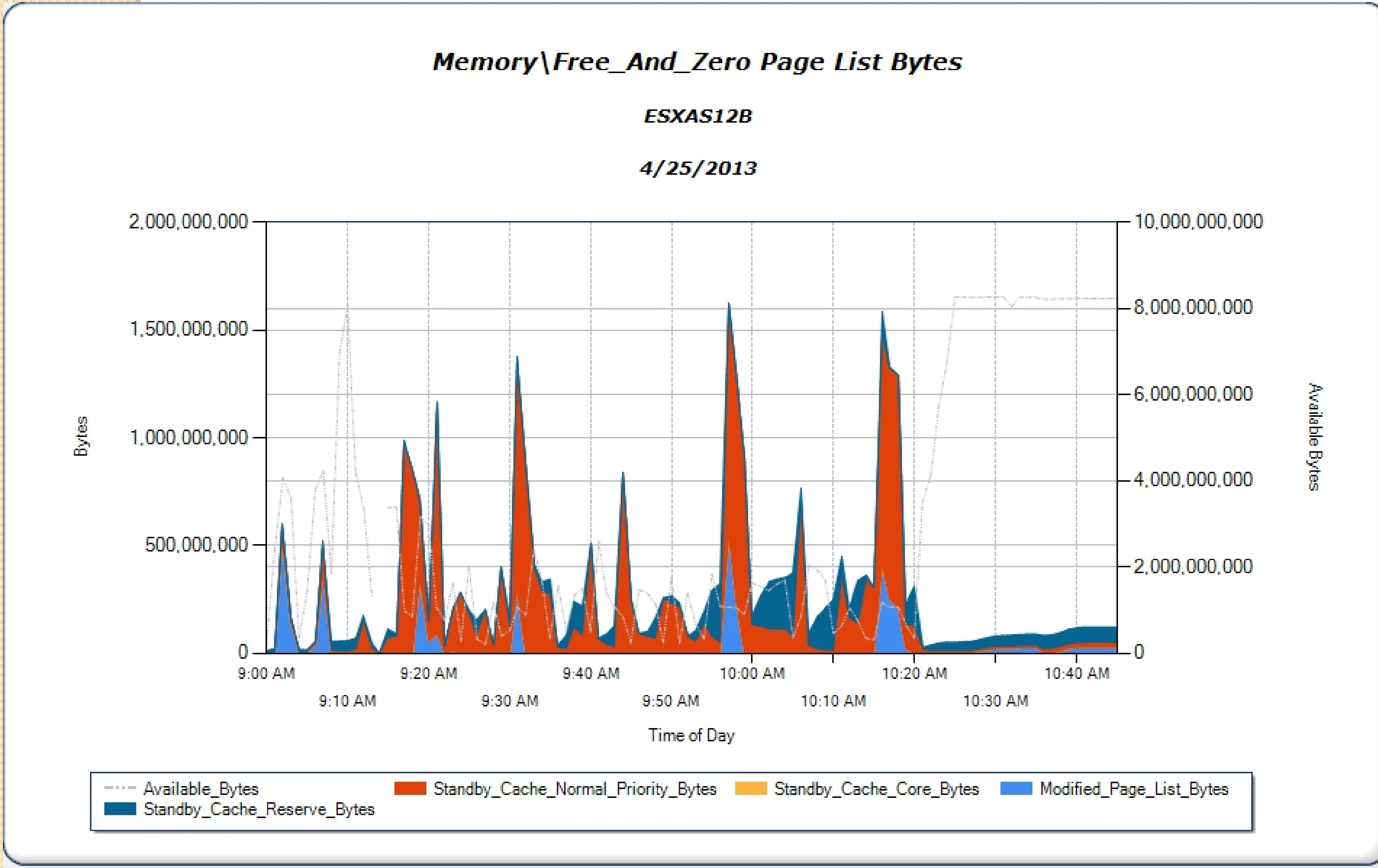
Ballooning: the view from inside



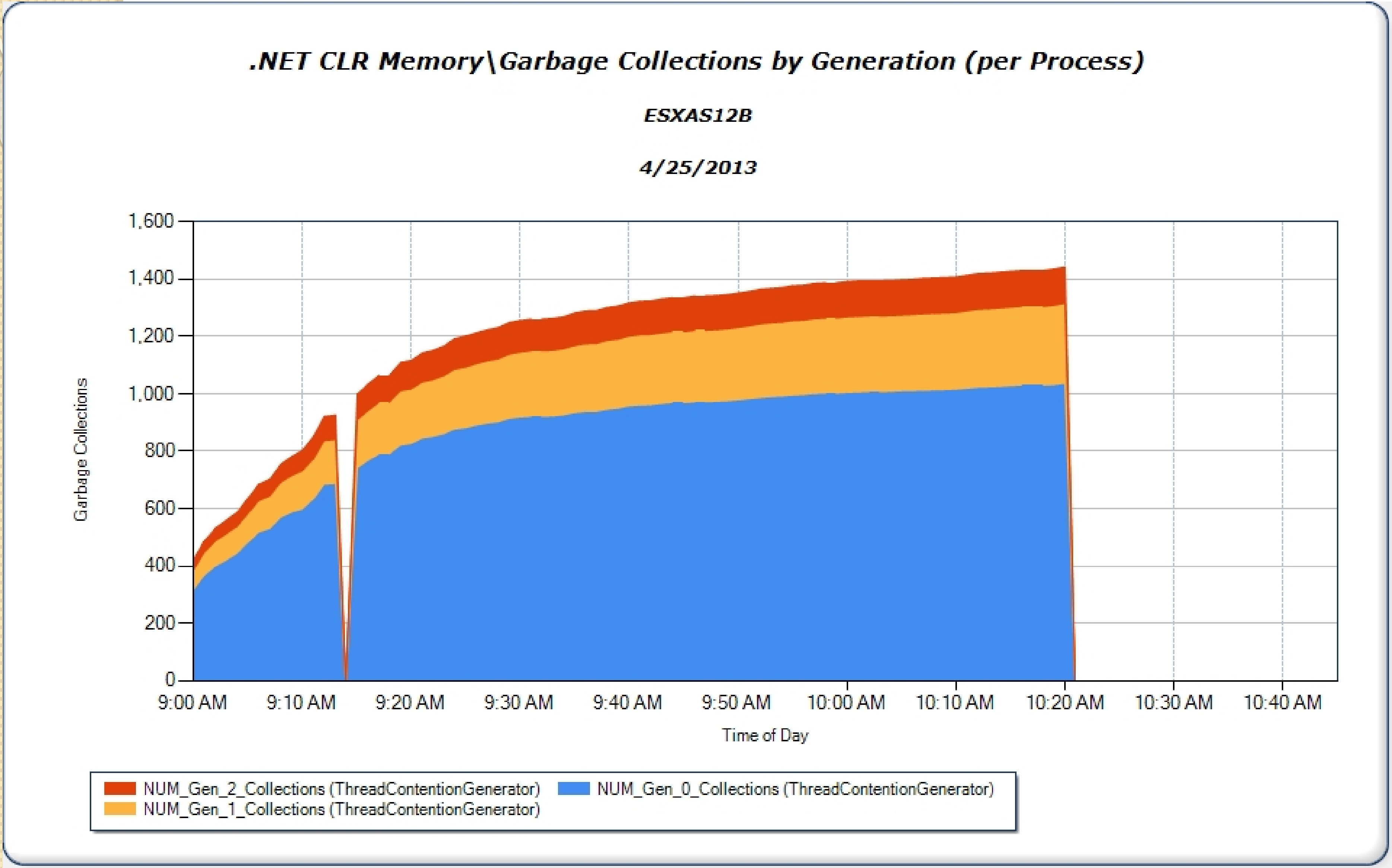
Ballooning: the view from inside



Ballooning: the view from inside



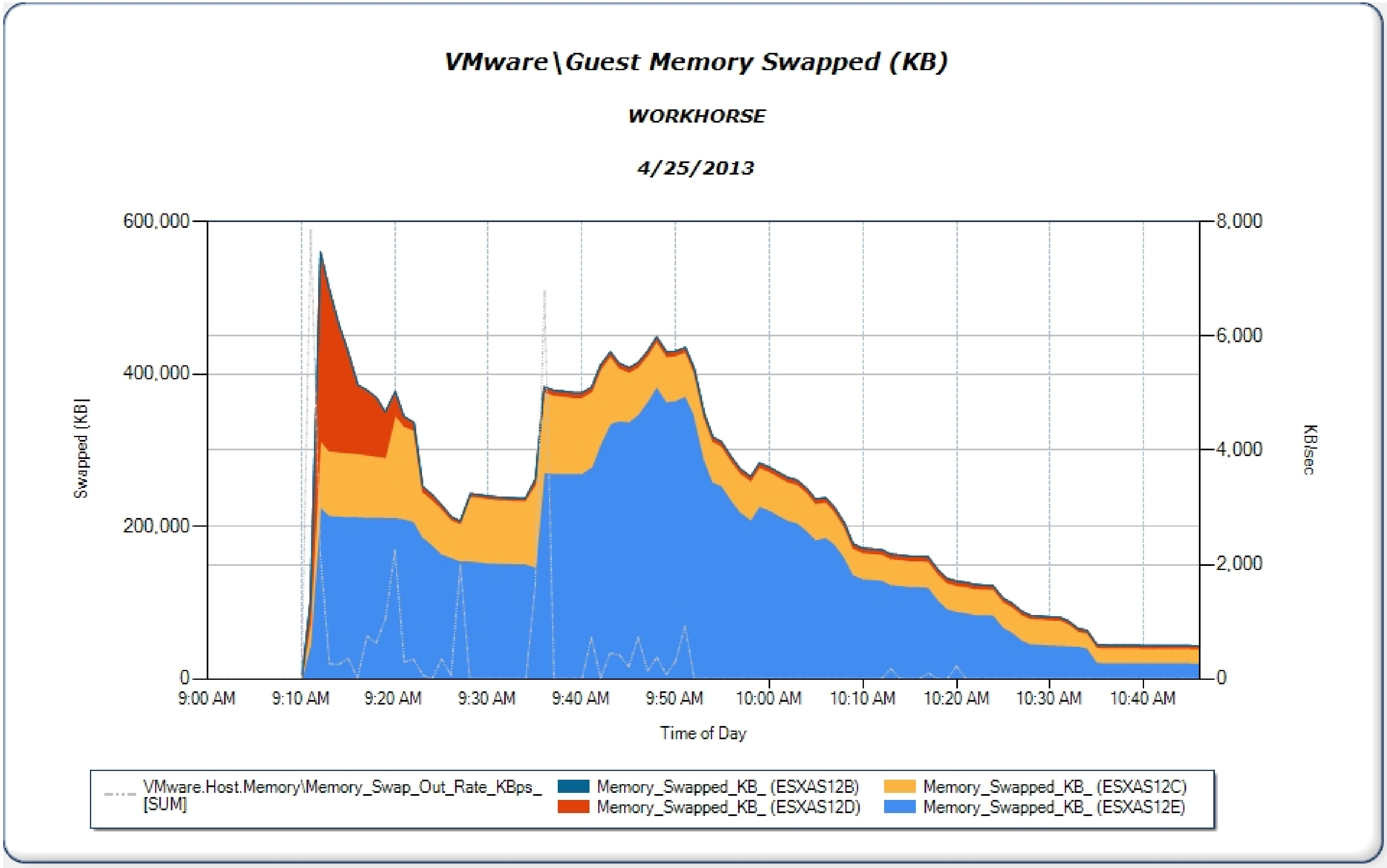
Ballooning: the view from inside



Swapping

- Swapping removes pages from the vm working set ***at random***
  - Random page replacement algorithms perform surprisingly well in simulations;
    - However, some worst-case scenarios are possible
    - e.g., VMware might select a Guest OS page containing active page table entries
  - In general, LRU-based page replacement is usually far superior.
    - VMware could move towards LRU-based page replacement in the future
- VMware Swapping is opaque to the underlying guest machine:
  - Can have a major performance impact that is often ***not*** apparent from internal performance measurements!

Memory Swapping



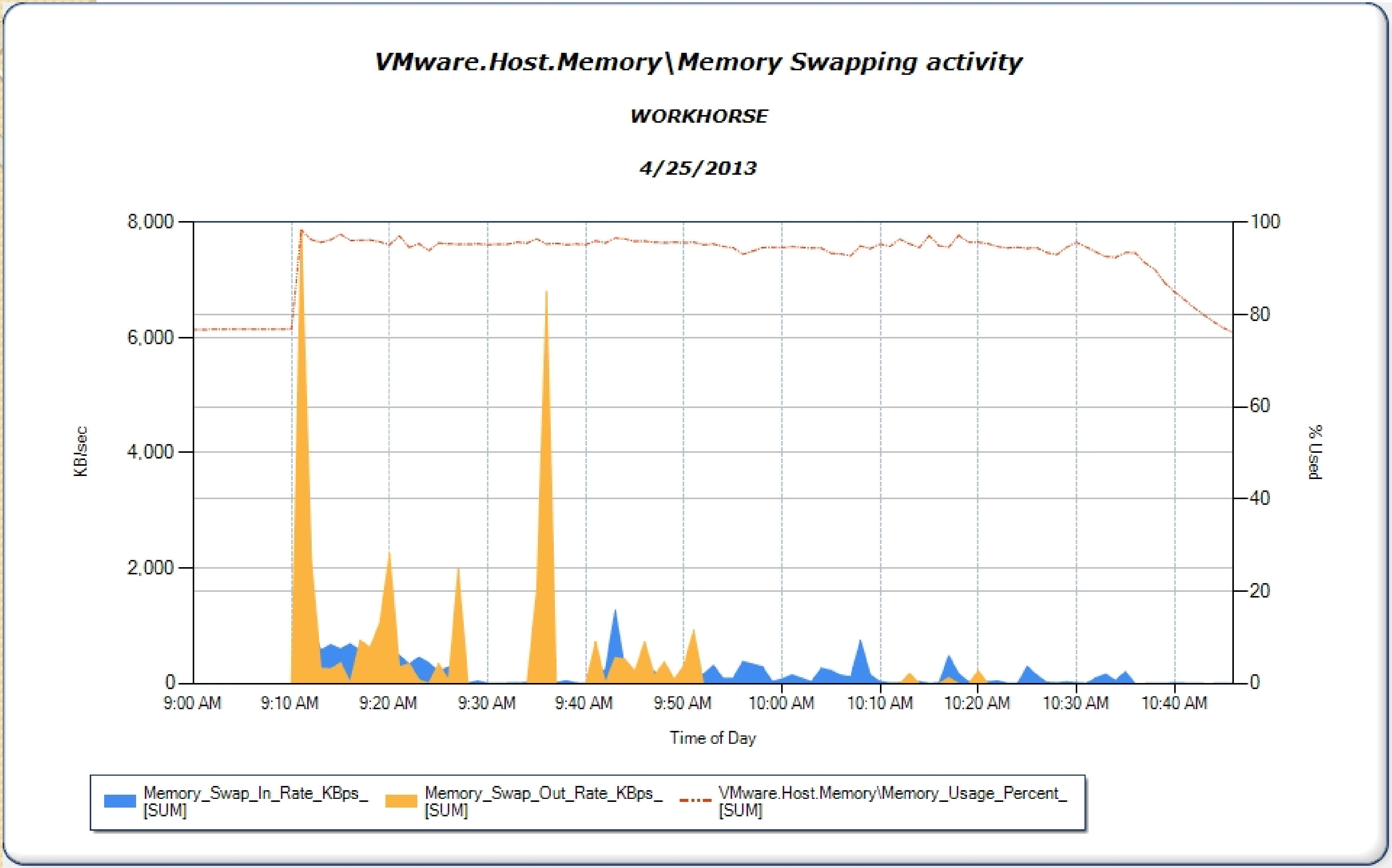
Swapping

- If Swapping occurs, there are some useful mitigation strategies
  - VM Host disk configuration
  - VM swapping to a memory-resident cache using compression

- But, generally, VMware Swapping should be avoided, where possible.
  - While swapping is bad, thrashing is even worse:

    swap in rate ≈ swap out rate

Swapping


VMware.Host.Memory\Memory Swapping activity
WORKHORSE
4/25/2013
KB/sec
% Used
8,000
6,000
4,000
2,000
0
100
80
60
40
20
0
9:00 AM
9:10 AM
9:20 AM
9:30 AM
9:40 AM
9:50 AM
10:00 AM
10:10 AM
10:20 AM
10:30 AM
10:40 AM
Time of Day
Memory_Swap_In_Rate_KBps_ [SUM]
Memory_Swap_Out_Rate_KBps_ [SUM]
VMware.Host.Memory\Memory_Usage_Percent_ [SUM]

RAM Best Practices

- Track Memory State transitions
- Limit use of Memory maximum Limits, but
- Protect key VMs using minimum ***Reservation*** sizes
- One VM that hogs virtual memory can impact the performance of all VMs residing in the same VM Host
  - Try to avoid swapping
  - For ballooning, monitor key Windows virtual memory contention indicators:
    - Pool non-Paged Bytes
    - Demand Zero paging, Pages/sec (to disk)
- Configure homogenous VMs to run on the same VM Host to maximize opportunities for content-based page sharing

Vision vs. Reality: Performance

- How well does VMware today address common performance concerns?
  - Can more efficient use of computer resources be achieved through pooling?

    Yes, definitely!
  - Can disruptions due to automated operations be minimized or controlled?

    Probably not.
  - Can application scalability be achieved through JIT auto-provisioning?

    Not without visibility into apps!
  - Are VMware's performance tools adequate to monitor these complex configurations?

    Not yet!

Vision vs. Reality: Performance

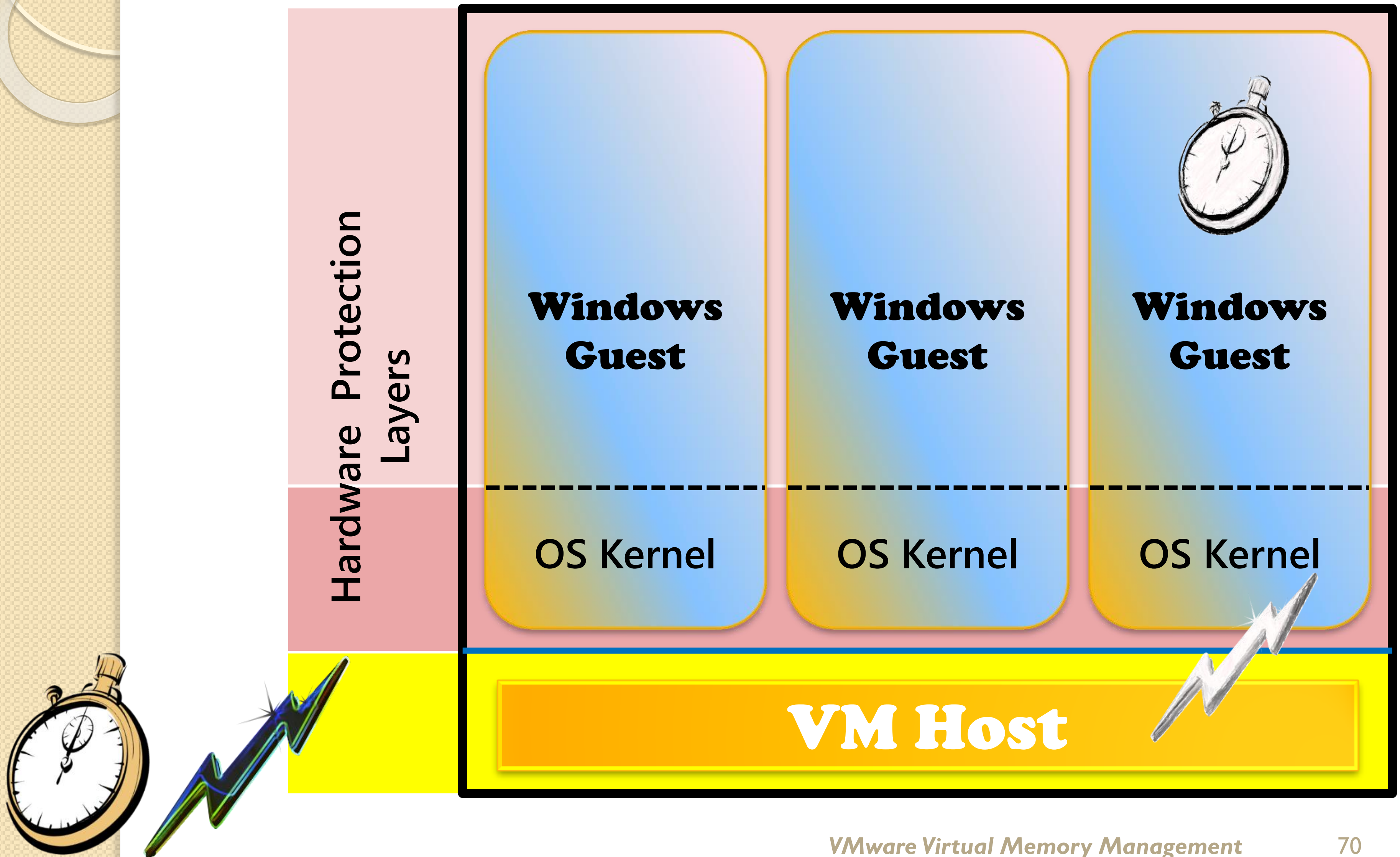
- Virtual data center performance challenges:
  - Resolving performance problems can require marshaling data and expertise from across many IT data center management disciplines.
  - VMware knowledge tends to be silo-ed in the IT technical support organization.
  - VMware has been secretive about its proprietary technology, including licensing that restricts customers from publishing performance benchmarks, etc.
    - Primarily due to competitive pressure from Microsoft, which is aggressively bundling virtualization into its OS.

Supplementary material

- Impact of virtualized clocks on Windows performance counters
  - Anytime you need to drill down from VMware to Windows performance counters, you need to understand whether VMware has perturbed the counter data

- Windows Physical Memory usage counters

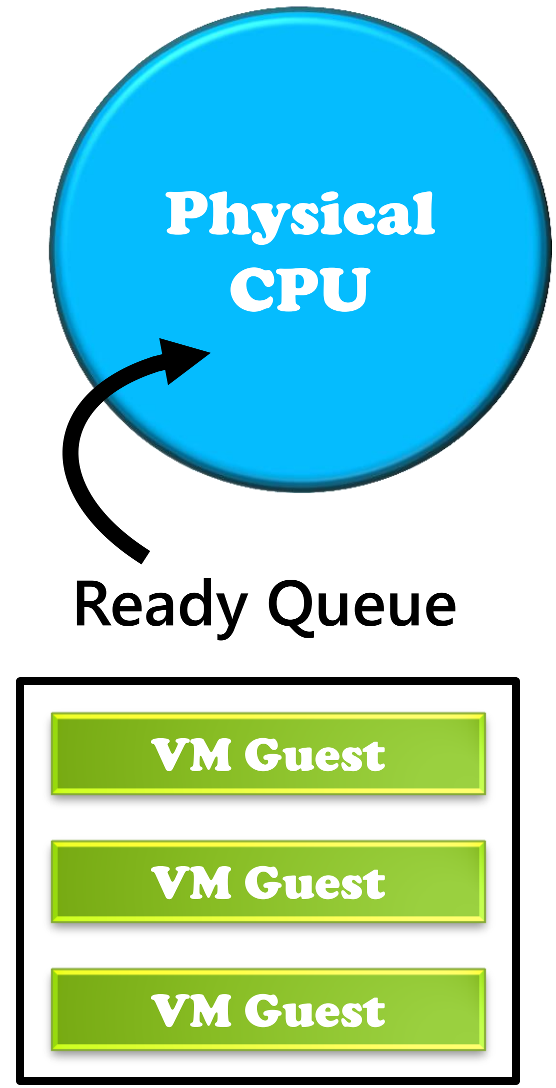
Virtualization of Clocks



Windows Guest machine clocks

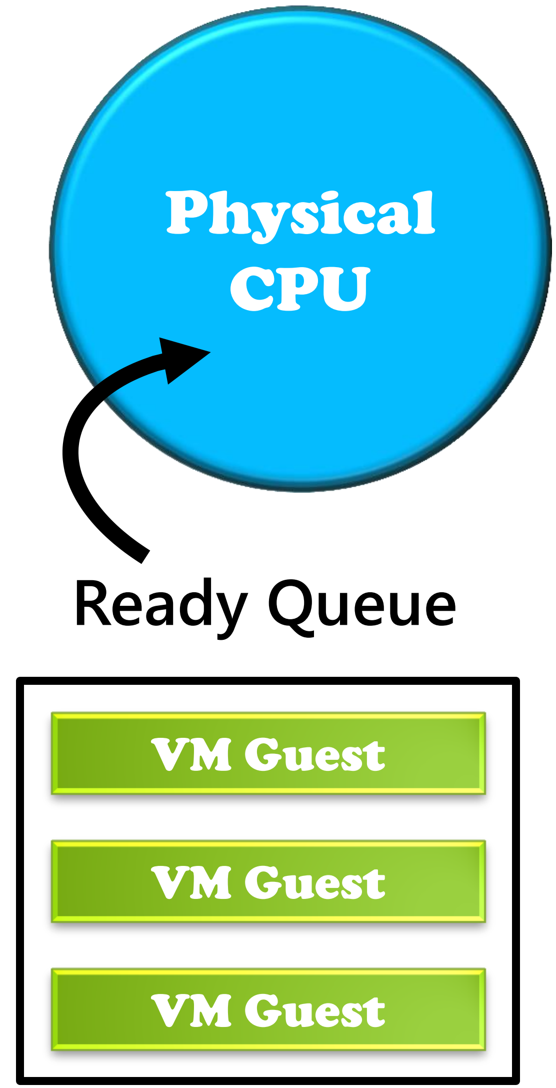
- Windows OS Scheduler wakes up 64 times per second
  - Update the System clock
  - Perform CPU accounting
- Clock interrupts are virtualized
- *rdtsc* instruction is also virtualized

- How does this effect measurements made from inside the guest OS?



Windows Guest machine clocks

- *All* guest machine clocks and timers are impacted
  - e.g., OS Scheduler periodic timer interrupts are not received at regular 15.6 ms quantum intervals
  - some intervals may be delayed when the Guest is accumulating Ready time
  - some interrupts may be *dropped* if the Guest machine is delayed long enough
- VMware responds to guest machine timer requests using *apparent time*
  - Ensures monotonically increasing clocks with values approaching the "real" time
  - Provides consistent clock values across multiple logical CPUs



Windows Guest performance monitoring

- Virtualization is transparent to the OS (by design)
  - Guest machine scheduling delays are only visible from outside the guest machine
  - VMware Host Performance counters
- All Guest machine clocks and timers are impacted
- When benchmarking, run a dedicated virtual machine configured with

logical processors < physical CPUs

Windows Guest performance monitoring

- % Processor Time measurements
  - Legacy measurements based on sampling
  - Potentially, fewer CPU time usage samples are gathered each performance monitoring interval
  - Time between samples is no longer guaranteed to be uniform
  - These impact accuracy, but they are (probably) not showstoppers.
  - In theory, VMware Host measurements of Guest machine *Ready time* delays could be used to *correct* internal measurement data that does not reflect the amount of time the VM guest spends queued, waiting for processor(s).

Windows Guest performance monitoring

- All rate/sec counters are subject to errors.
  - e.g., Physical Disk\Transfers/sec, TCP\Segments/sec, etc.
- These calculations are all based on underlying event counters that are reliable:

$$\frac{counter_{t1} - counter_{t0}}{duration}$$

- However, the denominator, the interval duration, is suspect.
- But,
  - Guest machine *Ready time* delays can be used to *correct* duration values to reflect any time the VM guest spent queued.
  - Internal event counts for disk, network activity, etc., can be reconciled against VM Host measurements over the same interval.

Page replacement in Windows is also triggered by thresholds

(Available Bytes)