VMWare ESX Memory Management

Dr. Sanjay P. Ahuja, Ph.D.
2010-14 FIS Distinguished Professor of Computer Science
School of Computing, UNF
Memory Virtualization Basics

- The virtual memory space, that is the guest’s memory space, is divided into blocks, typically 4KB, called pages. The physical memory, that is the host’s memory, is also divided into blocks, also typically 4KB (ESX/ESXi also provides support for large pages of 2 MB)

- When host physical memory is full, the data for virtual pages that are not present in host physical memory are stored on disk.

- When running a virtual machine, the hypervisor creates a contiguous addressable memory space for the virtual machine. This allows the hypervisor to run multiple virtual machines simultaneously while protecting the memory of each virtual machine from being accessed by others.

- From the view of the application running inside the virtual machine, the hypervisor adds an extra level of address translation that maps the guest physical address to the host physical address. As a result, there are three virtual memory layers in ESX: guest virtual memory, guest physical memory, and host physical memory.
Virtual memory levels (a) and memory address translation (b) in ESX
Memory Virtualization Basics

- In ESX, the address translation between guest physical memory and host physical memory is maintained by the hypervisor using a physical memory mapping data structure, or *pmap*, for each virtual machine.

- The *shadow page tables* maintain consistency with the guest virtual to guest physical address mapping in the guest page tables and the guest physical to host physical address mapping in the pmap data structure.

- This approach removes the virtualization overhead for the virtual machine’s normal memory accesses because the hardware TLB will cache the direct guest virtual to host physical memory address translations read from the shadow page tables.

- The hypervisor intercepts the virtual machine’s memory accesses and allocates host physical memory for the virtual machine on its first access to the memory. In order to avoid information leaking among virtual machines, the hypervisor always writes zeroes to the host physical memory before assigning it to a virtual machine.

- The hypervisor knows when to allocate host physical memory for a virtual machine because the first memory access from the virtual machine to a host physical memory will cause a page fault that can be easily captured by the hypervisor.
Memory Virtualization Basics

- **VM’s host memory usage** $\leq$ **VM’s guest memory size + VM’s overhead memory**

- Here, the virtual machine’s overhead memory is the extra host memory needed by the hypervisor for various virtualization data structures besides the memory allocated to the virtual machine. Its size depends on the number of virtual CPUs and the configured virtual machine memory size.
Memory Reclamation in ESX

- ESX uses several techniques to reclaim virtual machine memory, which are:
  
  1. **Transparent page sharing (TPS)** — reclaims memory by removing redundant pages with identical content
  2. **Ballooning** — reclaims memory by artificially increasing the memory pressure inside the guest
  3. **Hypervisor swapping** — reclaims memory by having ESX directly swap out the virtual machine’s memory
  4. **Memory compression** — reclaims memory by compressing the pages that need to be swapped
Memory Reclamation in ESX

- **Motivation:** Memory Overcommitment

  Host memory is overcommitted when the total amount of guest physical memory of the running virtual machines is larger than the amount of actual host memory. ESX supports memory overcommitment due to two important benefits it provides:

  - **Higher memory utilization:** With memory overcommitment, ESX ensures that host memory is consumed by active guest memory as much as possible. Memory overcommitment allows the hypervisor to use memory reclamation techniques to take the inactive or unused host physical memory away from the idle virtual machines and give it to other virtual machines that will actively use it.

  - **Higher consolidation ratio:** With memory overcommitment, each virtual machine has a smaller footprint in host memory usage, making it possible to fit more virtual machines on the host while still achieving good performance for all virtual machines.
Memory Reclamation in ESX

- For example, you can enable a host with 4G host physical memory to run three virtual machines with 2G guest physical memory each. Without memory overcommitment, only one virtual machine can be run because the hypervisor cannot reserve host memory for more than one virtual machine, considering that each virtual machine has overhead memory.

In order to effectively support memory overcommitment, the hypervisor must provide efficient host memory reclamation techniques.
When multiple virtual machines are running, some of them may have identical sets of memory content. This presents opportunities for sharing memory across virtual machines.

For example, several virtual machines may be running the same guest operating system, have the same applications, or contain the same user data.

With page sharing, the hypervisor can reclaim the redundant copies and keep only one copy, which is shared by multiple virtual machines in the host physical memory. As a result, the total virtual machine host memory consumption is reduced and a higher level of memory overcommitment is possible.

If one VM makes changes to a shared page, ESX Server stores and tracks those differences separately.
Transparent Page Sharing (TPS) – Content Based Page Sharing Algorithm in ESX
A hash value is generated based on the candidate guest physical page’s content. The hash value is then used as a key to look up a global hash table, in which each entry records a hash value and the physical page number of a shared page.

If the hash value of the candidate guest physical page matches an existing entry, a full comparison of the page contents is performed to exclude a false match.

Once the candidate guest physical page’s content is confirmed to match the content of an existing shared host physical page, the guest physical to host physical mapping of the candidate guest physical page is changed to the shared host physical page, and the redundant host memory copy (the page pointed to by the dashed arrow in the Figure) is reclaimed.

This remapping is invisible to the virtual machine and inaccessible to the guest operating system. Because of this invisibility, sensitive information cannot be leaked from one virtual machine to another.

A standard copy-on-write (CoW) technique is used to handle writes to the shared host physical pages. Any attempt to write to the shared pages will generate a minor page fault.
Transparent Page Sharing (TPS) – Content Based Page Sharing Algorithm in ESX

- A hash value is generated based on the candidate guest physical page’s content. The hash value is then used as a key to look up a global hash table, in which each entry records a hash value and the physical page number of a shared page.

- In the page fault handler, the hypervisor will transparently create a private copy of the page for the virtual machine and remap the affected guest physical page to this private copy. In this way, virtual machines can safely modify the shared pages without disrupting other virtual machines sharing that memory.

- Note that writing to a shared page does incur overhead compared to writing to non-shared pages due to the extra work performed in the page fault handler.
Balloonning

- When the hypervisor runs multiple guests and the total amount of the free host memory becomes low, none of the guests will free guest physical memory because the guest OS cannot detect the host’s memory shortage. Guests run isolated from each other and don’t even know they are virtual machines.

- Ballooning makes the guest operating system aware of the low memory status of the host.

- A balloon driver is loaded into the guest operating system. This balloon driver, `vmmemctl`, communicates with the hypervisor through a private channel. If the hypervisor needs to reclaim guest memory, it sets a proper target balloon size for the balloon driver, making it “inflate” by allocating guest physical pages within the guest.

- Typically, the hypervisor inflates the guest balloon when it is under memory pressure. By inflating the balloon, the hypervisor transfers the memory pressure from the host to the guest.
Ballooning: Inflating the balloon in a virtual machine
Ballooning: Inflating the balloon in a virtual machine

- In the Figure, four guest physical pages are mapped in the host physical memory. Two of the pages are used by the guest application and the other two pages (marked by stars) are in the guest operating system free list.

- Since the hypervisor cannot identify the two pages in the guest free list, it cannot reclaim the host physical pages that are backing them.

- Assuming the hypervisor needs to reclaim two pages from the virtual machine, it will set the target balloon size to two pages. After obtaining the target balloon size, the balloon driver allocates two guest physical pages inside the virtual machine and pins them.

- Once the memory is allocated, the balloon driver notifies the hypervisor about the page numbers of the pinned guest physical memory so that the hypervisor can reclaim the host physical pages that are backing them.

- In the Figure, dashed arrows point at these pages. The hypervisor can safely reclaim this host physical memory because neither the balloon driver nor the guest operating system relies on the contents of these pages. This means that no processes in the virtual machine will intentionally access those pages to read/write any values. Thus, the hypervisor does not need to allocate host physical memory to store the page contents.
Ballooning: Inflating the balloon in a virtual machine

- If any of these pages are re-accessed by the virtual machine for some reason, the hypervisor will treat it as a normal virtual machine memory allocation and allocate a new host physical page for the virtual machine.

- When the hypervisor decides to deflate the balloon—by setting a smaller target balloon size—the balloon driver deallocates the pinned guest physical memory, which releases it for the guest’s applications.

- By inflating the balloon, a virtual machine consumes less physical memory on the host, but more physical memory inside the guest. As a result, the hypervisor offloads some of its memory overload to the guest operating system while slightly loading the virtual machine. That is, the hypervisor transfers the memory pressure from the host to the virtual machine.

- Ballooning induces guest memory pressure. In response, the balloon driver allocates and pins guest physical memory. The guest operating system determines if it needs to page out guest physical memory to satisfy the balloon driver’s allocation requests.

- If the virtual machine has plenty of free guest physical memory, inflating the balloon will induce no paging and will not impact guest performance.
Ballooning: Inflating the balloon in a virtual machine

- As illustrated in the Figure, the balloon driver allocates the free guest physical memory from the guest free list. Hence, guest-level paging is not necessary.

- However, if the guest is already under memory pressure, the guest operating system decides which guest physical pages to be paged out to the virtual swap device in order to satisfy the balloon driver’s allocation requests.

- The genius of ballooning is that it allows the guest operating system to intelligently make the hard decision about which pages to be paged out without the hypervisor’s involvement.

- For ballooning to work as intended, the guest operating system must install and enable the balloon driver, which is included in VMware Tools. The guest operating system must have sufficient virtual swap space configured for guest paging to be possible.

- Ballooning might not reclaim memory quickly enough to satisfy host memory demands.
Hypervisor Swapping

- In the cases where ballooning and transparent page sharing are not sufficient to reclaim memory, ESX employs hypervisor swapping to reclaim memory.

- At virtual machine startup, the hypervisor creates a separate swap file for the virtual machine. Then, if necessary, the hypervisor can directly swap out guest physical memory to the swap file, which frees host physical memory for other virtual machines.

- Both page sharing and ballooning take time to reclaim memory. Ballooning speed relies on the guest operating system’s response time for memory allocation.

- In contrast, hypervisor swapping is a guaranteed technique to reclaim a specific amount of memory within a specific amount of time. However, hypervisor swapping is used as a last resort to reclaim memory from the virtual machine due to limitations on performance.
Hypervisor Swapping – Limitations on Performance

- Page selection problems: Under certain circumstances, hypervisor swapping may severely penalize guest performance. This occurs when the hypervisor has no knowledge about which guest physical pages should be swapped out, and the swapping may cause unintended interactions with the native memory management policies in the guest operating system.

- High swap-in latency: Swapping in pages is expensive for a VM. If the hypervisor swaps out a guest page and the guest subsequently accesses that page, the VM will get blocked until the page is swapped in from disk. High swap-in latency, which can be tens of milliseconds, can severely degrade guest performance.

- ESX mitigates the impact of interacting with guest operating system memory management by randomly selecting the swapped guest physical pages.
Memory Compression

- Idea: If the swapped out pages can be compressed and stored in a compression cache located in the main memory, the next access to the page only causes a page decompression which can be an order of magnitude faster than the disk access.

- With memory compression, only a few uncompressible pages need to be swapped out if the compression cache is not full. This means the number of future synchronous swap-in operations will be reduced. Hence, it may improve application performance significantly when the host is in heavy memory pressure.

- In ESX 4.1, only the swap candidate pages will be compressed. This means ESX will not proactively compress guest pages when host swapping is not necessary. So memory compression does not affect workload performance when host memory is undercommitted.
Memory Compression

Figure: Host swapping vs. memory compression in ESX

Assuming ESX needs to reclaim two 4KB physical pages from a VM through host swapping, page A and B are the selected pages. With host swapping only, these two pages will be directly swapped to disk and two physical pages are reclaimed.
Memory Compression

- However, with memory compression, each swap candidate page will be compressed and stored using 2KB of space in a per-VM compression cache.

- Note that page compression would be much faster than the normal page swap out operation which involves a disk I/O.

- Page compression will fail if the compression ratio is less than 50% and the uncompressible pages will be swapped out. As a result, every successful page compression is accounted for reclaiming 2KB of physical memory.

- As illustrated in Figure c, pages A and B are compressed and stored as half-pages in the compression cache. Although both pages are removed from VM guest memory, the actual reclaimed memory size is one page.
Memory Compression

- If any of the subsequent memory access misses in the VM guest memory, the compression cache will be checked first using the host physical page number. If the page is found in the compression cache, it will be decompressed and pushed back to the guest memory. This page is then removed from the compression cache. Otherwise, the memory request is sent to the host swap device and the VM is blocked.

- In ESX 4.1, the default maximum compression per-VM cache size is conservatively set to 10% of configured VM memory size.
Reclaiming Memory

- ESX maintains four host free memory states: high, soft, hard, and low, which are reflected by four thresholds: 6%, 4%, 2%, and 1% of host memory respectively.

- When to use ballooning or swapping (which activates memory compression) to reclaim host memory is largely determined by the current host free memory state. In the high state, the aggregate virtual machine guest memory usage is smaller than the host memory size. Whether or not host memory is overcommitted, the hypervisor will not reclaim memory through ballooning or swapping.

- If host free memory drops towards the soft threshold, the hypervisor starts to reclaim memory using ballooning. Ballooning happens before free memory actually reaches the soft threshold because it takes time for the balloon driver to allocate and pin guest physical memory. Usually, the balloon driver is able to reclaim memory in a timely fashion so that the host free memory stays above the soft threshold.
Reclaiming Memory

- If ballooning is not sufficient to reclaim memory or the host free memory drops towards the hard threshold, the hypervisor starts to use swapping in addition to using ballooning. During swapping, memory compression is activated as well. With host swapping and memory compression, the hypervisor should be able to quickly reclaim memory and bring the host memory state back to the soft state.

- In a rare case where host free memory drops below the low threshold, the hypervisor continues to reclaim memory through swapping and memory compression, and additionally blocks the execution of all virtual machines that consume more memory than their target memory allocations.

- In certain scenarios, host memory reclamation happens regardless of the current host free memory state. For example, even if host free memory is in the high state, memory reclamation is still mandatory when a virtual machine’s memory usage exceeds its specified memory limit. If this happens, the hypervisor will employ ballooning and, if necessary, swapping and memory compression to reclaim memory from the virtual machine until the virtual machine’s host memory usage falls back to its specified limit.
ESX Memory Allocation Management

- ESX employs a share-based allocation algorithm to achieve efficient memory utilization for all virtual machines and to guarantee memory to those virtual machines which need it most.

- Each virtual machine consumes memory based on its configured size, plus additional overhead memory for virtualization.

- **Configured Size**: is a construct maintained by the virtualization layer for the virtual machine. It is the amount of memory that is presented to the guest operating system, but it is independent of the amount of physical RAM that is allocated to the virtual machine, which depends on the resource settings (shares, reservation, limit) explained in the next slide.
ESX Memory Allocation Management

- ESX provides three configurable parameters to control the host memory allocation for a virtual machine: Shares, Reservation, and Limit.
- **Reservation**: is a guaranteed lower bound on the amount of host physical memory the host reserves for a virtual machine even when host memory is overcommitted.
- **Limit**: is the upper bound of the amount of host physical memory allocated for a virtual machine. The virtual machine’s memory allocation is also implicitly limited by its configured size.
- **Shares**: entitle a virtual machine to a fraction of available host physical memory, based on a proportional-share allocation policy. For example, a virtual machine with twice as many shares as another is generally entitled to consume twice as much memory, subject to its limit and reservation constraints.
ESX Memory Allocation Management

- When host memory is overcommitted, a virtual machine’s allocation target is somewhere between its specified reservation and specified limit depending on the virtual machine’s shares and the system load.

- A guideline when using reservations and limits together, is to set the reservation for each VM to 50% of its limit.

- When a VM starts up a swap file which has a size of the limit minus the reservation is created. For example we have a VM with a 1024MB limit and a reservation of 512MB. The swap file created will be 1024MB – 512MB = 512MB.
ESX Memory Allocation Management

- Shares play an important role in determining the allocation targets when memory is overcommitted. When the hypervisor needs memory, it reclaims memory from the virtual machine that owns the fewest shares-per-allocated page.

- Shares are set in the VM’s settings and can be set to “low”, “normal”, “high” or a custom value.
  - low = 5 shares per 1MB allocated to the VM
  - normal = 10 shares per 1MB allocated to the VM
  - high = 20 shares per 1MB allocated to the VM

- It is important to note that the more memory assigned to a VM, the more shares it receives.

- Example: Say there are 5 VMs each with 2,000MB memory allocated and their share values set to “normal”. The ESX host only has 4,000MB of physical machine memory available for virtual machines. Each VM receives 20,000 shares according to the “normal” setting (10 * 2,000). The sum of all shares is 5 * 20,000 = 100,000. Every VM will receive an equal share of 20,000/100,000 = 1/5th of the resources available = 4,000/5 = 800MB.
Example: Now say the shares setting on 1 VM is changed to “High”, which results in this VM receiving 40,000 shares (20 * 2,000) instead of 20,000. The sum of all shares is now increased to 120,000. This VM will receive 40,000/120,000 = 1/3rd of the resources available. Thus 4,000/3 = 1333 MB. All the other 4 VMs will receive only 20,000/120,000 = 1/6th of the available resources = 4,000/6 = 666 MB each.

A significant limitation of the pure proportional-share algorithm is that it does not incorporate any information about the actual memory usage of the virtual machine. As a result, some idle virtual machines with high shares can retain idle memory unproductively, while some active virtual machines with fewer shares suffer from lack of memory.

ESX resolves this problem by estimating a virtual machine’s working set size and charging a virtual machine more for the idle memory than for the actively used memory through an idle tax.

A virtual machine’s shares-per-allocated page ratio is adjusted to be lower if a fraction of the virtual machine’s memory is idle. Hence, memory will be reclaimed preferentially from the virtual machines that are not fully utilizing their allocated memory.
ESX Memory Management: A Comparison

- vSphere lets users provide critical VMs with guaranteed memory. The memory “Shares” and “Reservation” settings prioritize memory allocated to each VM and ensure enough host RAM is reserved for the active working memory of each guest OS.

- ESXi efficiently reclaims memory from less busy virtual machines when needed by more active virtual machines using four techniques: transparent page sharing; in-guest ballooning; memory compression; and, hypervisor level swapping. Those technologies permit aggressive memory oversubscription with minimal performance impact using any supported ESXi guest operating system.

- Citrix XenServer and Microsoft Hyper-V rely solely on in-guest ballooning – they call it “Dynamic Memory” – to reclaim memory and permit memory oversubscription.
ESX Memory Management: A Comparison

- Figures shown demonstrate that ballooning alone cannot respond fast enough and provide enough memory savings to prevent performance slowdowns when memory is oversubscribed.
ESX Memory Management: A Comparison

Figure 4. ESXi delivers 18.9% higher aggregate performance than Microsoft Hyper-V at high virtual machine densities.