VMware HA and DRS
Capacity Planning

Table of Contents

Overview

VMware is the leading provider of infrastructure virtualization products with a mature and robust suite of products available for nearly every aspect of IT infrastructure. VMware products have been in mainstream production scenarios for the last several years; however the most comprehensive enterprise production-use features were released in VMware Infrastructure 3 (VI3) in June 2006, and then further advanced with the release of VMware ESX Server 3.5 and VirtualCenter 2.5 in December of 2007. VI3 offers unsurpassed virtualization technology management, physical resource optimization, and operational automation capabilities within a single integrated platform.

Physical memory is a critical component of all VMware Infrastructure 3 deployments, particularly in environments where VMware HA and DRS will be leveraged. This paper will assist readers in understanding how to interpret the current virtualization workload in the VI3 environment and correlate that information into a capacity planning strategy for a successful VMware HA/DRS implementation, taking into account the memory needs when host failures occur, memory conditions when DRS is active, and improving HA response with DRS.

This paper is intended for VMware and Kingston partners, resellers, and customers who want to implement VI3 in production scenarios and want to proactively plan for physical memory requirements.

Introduction to VMware HA

VMware HA, which stands for *High Availability*, is an enterprise-level feature introduced as part of VMware Infrastructure 3 in 2006. VMware HA is a virtualization-based distributed infrastructure service that provides simple-to-use and cost-effective high availability without the added cost or complexity of traditional application clustering technologies, which typically require additional dedicated hardware or layered software. This layer of protection is invisible to the guest operating system and provides high availability features for applications and services that might otherwise go unprotected. VMware HA is included in VMware Infrastructure Standard and Enterprise editions.

A VMware HA cluster consists of a logical grouping of two or more VMware ESX hosts that enable high-availability services to a collection of virtual machines. In an HA cluster, each VMware ESX Server maintains an HA agent that continuously monitors a heartbeat with the other hosts in the cluster. Heartbeats are sent every five seconds over the Service Console network connection. If the heartbeat from a particular VMware ESX host is lost for three consecutive intervals, it is assumed the host has either failed or become isolated from the network; the virtual machines that were running on that host will be automatically restarted on other hosts in the cluster. Likewise, if a host loses all heartbeats from any particular host in the cluster, it begins an internal process to determine if it is isolated from the rest of the cluster and if so, will proactively power down any running virtual machines in anticipation of other hosts powering them on. VMware HA also protects against multiple ESX Server failures (up to four) within the cluster, provided there is ample capacity in the cluster to accommodate the post-failure load of the virtual machines failing over.

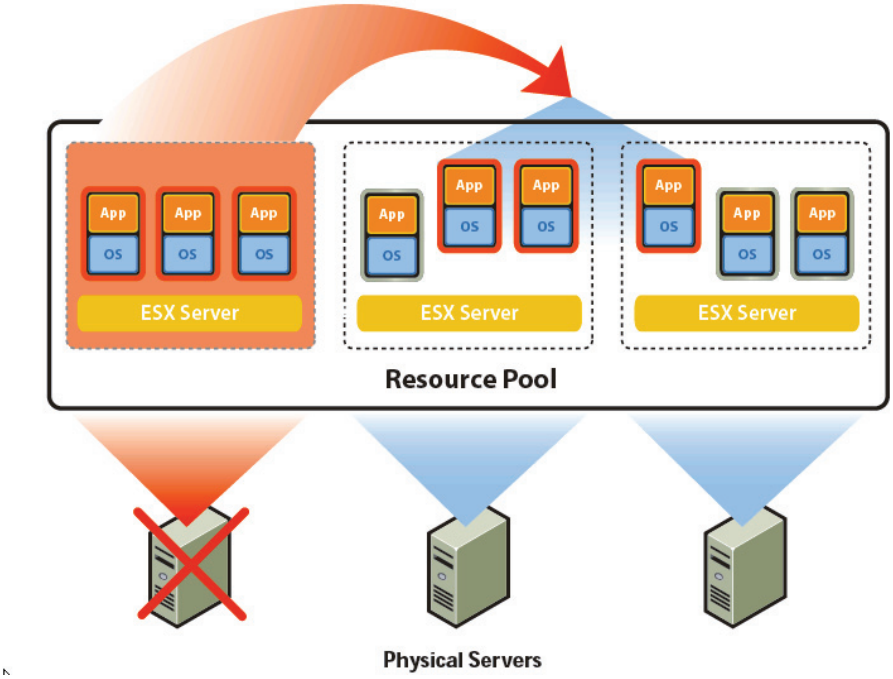

Figure 1, VMware High Availability (HA) Functionality

During a VMware HA failover event, the guest OS perceives the event as a restart from a hardware crash and not an orderly shutdown and therefore the recovery is not stateful; any transactions in progress in the virtual machine that were not yet committed will be lost. The guest OS will not detect any difference in hardware, even if the target ESX Server host hardware differs from the source ESX Server host hardware.

It is important to point out that VMware VMotion enables live migration of running virtual machines from one physical host to another and is considered a key enabler of business continuity by shuffling workloads around to ESX Server hosts with spare capacity. However, VMotion is not a component of HA because VMotion requires a coordinated handoff between two physical ESX Server hosts, and VMware HA applies only when one or more ESX Server hosts have failed and not shut down in an orderly fashion. When ESX Server hosts are shutdown orderly, the Distributed Resource Scheduler (DRS) service can leverage VMotion to relocate running virtual machines away from the ESX Servers that are shutting down.

VMware HA/DRS Clusters

VMware clusters are created in VirtualCenter and contain one or more VMware ESX hosts. The creation of the cluster is merely a container, and whether that cluster is providing HA services, DRS services, or both is based solely on the configured parameters of the cluster itself.

VMware HA Configuration

When VMware HA is enabled on a cluster in VirtualCenter, certain parameters are configured to identify capacity requirements and isolation response. Key criteria are described in more detail below:

- **Allow Violation of Resource Constraints** – This parameter determines whether VMware HA will continue to power on virtual machines if the cluster is no longer able to guarantee resources based on the configured per-virtual machine reservations.
- **Number of Host Failures Permitted** – This determines how many host failures are permitted before the VMware HA determines the cluster is no longer capable of satisfying resource requirements, based on the configured per-virtual machine reservations.
- **Restart Priority** — This setting determines the order that virtual machines are restarted upon ESX Server host failure. The priority applies on a per-host basis, meaning that if multiple hosts fail, VMware HA starts all virtual machines from the first host in order of priority, and then all virtual machines from the next host in order of priority.
- **Isolation Response** — When a host in a VMware HA cluster loses network connectivity but hasn't actually failed, the host is said to be in an 'isolated' state. By default, isolation response is triggered after missing 12 seconds worth of heartbeats. The default isolation response for the isolated host server is to power down all running virtual machines. Three seconds later, or a total of 15 seconds from the start of the isolation event, the other ESX Server hosts in the cluster assume that the isolated host has failed and begin powering on the virtual machines that were running on that isolated host.

> **Note:** Beginning with VMware VirtualCenter 2.0.2, the isolation response time can be modified from the default of 15 seconds using the *das.failuredetectiontime* setting in the *Advanced Options* of the VMware HA cluster configuration.

Key factors to VMware HA Operation

VMware HA has several prerequisites that must be satisfied before the HA agents on all hosts within the cluster start successfully. These prerequisites are as follows:

- **DNS Configuration** — All ESX Server hosts and the VirtualCenter server must have valid DNS A and PTR records that resolve to the configured host names exactly. The ESX Server hosts and the VirtualCenter servers must also be configured to resolve names via DNS (and not via host file)
- **Time Synchronization** — The system clock of the ESX Server hosts and VirtualCenter server must be synchronized by the same clock source or at least be freewheeling within one minute of each other
- **Network Configuration** — Establish redundant network connectivity for the Service Console, either by creating multiple vSwitches with Service Console interfaces or by adding additional physical NICs to the primary vSwitch hosting the Server Console interface. Since isolation response is partially validated by default gateway connectivity, the default gateway IP address must be reachable via PING; if not, the cluster's *das.isolationaddress* parameter must be configured to provide an alternate address for the logical connectivity check.

More information on VMware HA Architecture and best practices can be found in the following whitepaper on VMware's web site: http://www.vmware.com/resources/techresources/402.

Introduction to VMware DRS

VMware Distributed Resource Scheduler (DRS) was also released as part of VMware Infrastructure 3. DRS functions by continually analyzing and optionally rebalancing the running x86 workloads across the VMware ESX Servers running in the cluster. DRS clusters also allow for the use of Resource Pools, which aggregate resources from multiple VMware ESX hosts into a logical container for easier management.

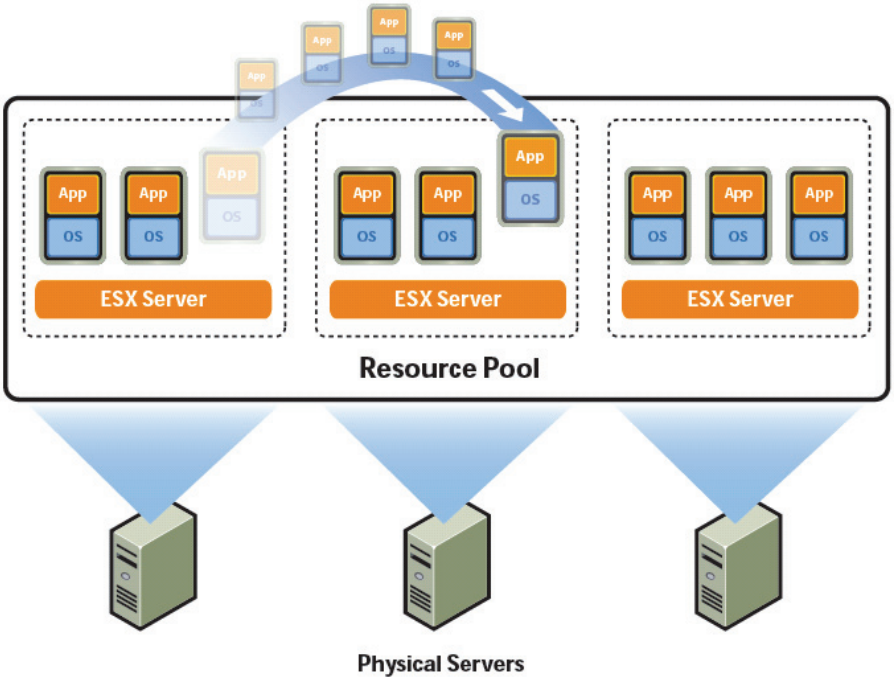

Figure 2, Dynamic Resource Scheduler (DRS) Functionality

DRS works by continuously monitoring physical resource utilization of virtual machines across resource pools and determining the optimal allocation of available resources among the ESX Server hosts in the VI3 cluster based on pre-defined rules that can be tied to business needs and changing priorities. In fully automated configuration, when a virtual machine (or a group of virtual machines) experiences an increased load, DRS automatically allocates additional resources by redistributing virtual machines among the physical ESX Servers in the resource pool. In environments that are stricter concerning dynamic changes, DRS can be configured to provide recommendations to VirtualCenter administrators on the redistribution of virtual machines as opposed to dynamically moving them.

DRS Operating Modes

VMware DRS, like VMware HA, is a feature enabled on a VI3 cluster. When the DRS feature is enabled on a cluster, the memory and CPU resources of the individual ESX Servers are then managed as a single resource pool. DRS can be configured to operate in one of three modes: fully automatic, partially automated, or manual mode. When set for fully automatic mode, DRS determines the optimal distribution of virtual machines among the physical servers and uses VMotion to redistribute the running virtual machines. In partially automated mode, DRS determines the optimal physical server to power on a virtual machine, and then only provides redistribution recommendations, which the administrator can then execute leveraging VMotion. In manual mode, the administrator must select the desired ESX Server to start virtual machines, and must act upon migration recommendations administratively.

Understanding the Relationship of VMware HA and DRS

VMware HA and VMware DRS are two different technologies that operate independently. A common misconception is that these technologies work highly interactively in an HA failover event to ensure the cluster remains in balance as virtual machines are restarted; this most likely stems from the fact that you can have a single cluster support both VMware HA and VMware DRS simultaneously.

On the contrary, VMware HA and VMware DRS have no interaction with one another. Although both are configured via VMware VirtualCenter, DRS is a VirtualCenter-implemented feature while HA is a peer-model and although it is configured using VirtualCenter, it is implemented at the individual host level.

Advantage of using DRS with HA

Although DRS and HA are completely independent processes, they can complement each other to create a robust, automated recovery mechanism for the virtualization platform. When an HA failover occurs, VMware HA will restart virtual machines in priority order from each failed ESX Server in succession *on a single elected ESX Server host*. In other words all the virtual machines from one or several failed ESX Servers will start on a single host until resource constraints prevent any further virtual machines from starting or the ESX Server maximum running vCPU count is exceeded.

The advantage of using DRS with HA is that once the virtual machines from a failed ESX Server are restarted, the load is analyzed by DRS and redistributed automatically. The maximum advantage is gained when DRS is set for fully automated mode; in partially automated or manual mode, the administrator must be present at the time of failure to avoid application or service impact at the virtual machine level, a result of either failure to power on a virtual machine, or by poor performance the result of an oversubscribed ESX Server host.

Maximum and Recommended Cluster Size

The maximum number of ESX Server hosts in a DRS and HA enabled cluster is thirty-two and sixteen, respectively. Clusters enabled for both DRS and HA will be restricted to a maximum of sixteen server hosts as a result of the HA cluster size limitation. However, these are configurable maximums and not necessarily real-world recommended cluster sizes.

The actual cluster size will be dependent on a number of factors, including the number of virtual machines per LUN, the number of LUNs exposed per host and the number of hosts exposed to each LUN. Additional deciding factors include the anticipated cyclic variation in virtual machine load, SAN architecture, and network architecture, and host failure allowances.

- **Anticipated Cyclic Variation in Virtual Machine Load**—DRS by default is limited to executing two simultaneous VMotion tasks per ESX Server. DRS load balancing is not instantaneous, so balancing virtual machines with rapidly oscillating load with more consistent

workload characteristics will likely require a larger cluster to ensure a more rapid redistribution response.

- **Storage Architecture**—In order for VMotion and DRS to function properly, all ESX servers in the cluster must have shared access to the storage hosting the virtual machine disk files. Assuming the maximum DRS cluster size with an average of one LUN per ESX Server, each ESX Server could have 32 LUNs provisioned to each, an amount that may be challenging to manage, particularly when provisioning new virtual machines.

  There are also performance considerations as to the number of hosts that should concurrently access a single LUN, as well as storage IO considerations. Too many hosts with concurrent access to a single LUN can inflict a performance penalty due to LUN-level SCSI reservations associated with virtual machine file operations and LUN metadata updates.

- **Network Architecture**—The same VLANs must be provisioned to all ESX Servers within the HA/DRS cluster. Inconsistency in the network configuration may result in virtual machines losing network connectivity after redistribution, in the event they are moved to a physical host lacking the required VLAN.

- **Host Failure Allowance** – The number of hosts that can fail within the cluster and continue to meet resource demands will be a deciding factor in cluster size.

- **LUN Size and Quantity** – LUN sizing is a factor that is outside the scope of this paper, but it is worth mentioning that the number of hosts accessing a concurrent LUN may impact performance, dependent upon what workloads are hosted on those LUNs. Although an HA cluster can contain up to sixteen hosts, recommended cluster sizes generally fall somewhere between eight and twelve, based on the individual organization's needs.

Understanding the Role of Memory in VMware HA

Memory is a critical resource in the success of any VMware Infrastructure implementation. However, while capacity planning is important from a provisioning standpoint alone, it is even more critical to understand how VMware HA can affect memory utilization.

VMware ESX Server contains a robust memory management subsystem to optimize the physical memory used by running virtual machines and is comprised of three key mechanisms.

Transparent Page Sharing

This process ensures that duplicate memory blocks are removed and replaced with logical pointers to a single read-only copy; a copy-on write (COW) mechanism is leveraged to quickly create a private working copy of any shared memory blocks to which a particular virtual machine attempts to write.

Memory Ballooning

This process allows memory currently allocated to virtual machines to be recovered and used for virtual machines that have immediate demands for physical memory. The extent of ballooning is adjusted on the fly by the VMkernel based on the current memory utilization levels and the demand for memory resources. Memory ballooning is implemented as part of the VMtools running in each virtual machine. Interestingly enough, excessive memory ballooning can have a significant impact on the savings realized through Transparent Page Sharing due to the highly dynamic nature of ESX Server's ballooning techniques.

Memory Swapping

This mechanism provides a last resort to recover physical memory by moving the RAM contents of running x86 workloads not current being used to disk, making room for virtual machines that have an immediate demand. Each virtual machine is allocated an individual swap file, equal in size to the amount of memory configured for the virtual machine, minus any reservation in the virtual machine configuration.

Memory swapping can have a significant impact on a VMware ESX host server's performance as the VMkernel must dedicate CPU cycles to the swapping mechanism rather than virtual machine workload processing, adding significant overhead to the virtualization layer. In addition, the nanosecond access times for memory are replaced by millisecond access times for networked mass storage devices. Because ESX host servers have high memory utilization rates, memory swapping can have more impact than on a less utilized server. For this reason, swapping is used as a mechanism of last resort.

If not carefully planned, VMware HA can significantly impact the effectiveness of all three of these mechanisms.

DRS and HA Effects on Memory Utilization

VMware DRS and HA do not require additional memory or place additional resource constraints on the VMware ESX hosts. However, if not architected properly and monitored continually, performance can be significantly impacted, particularly when HA failover events occur.

As mentioned earlier, transparent page sharing is very effective at reclaiming redundant pages in physical memory. The process begins scanning physical memory for duplicate pages almost immediately but will take time to achieve maximum optimization. As time passes and more redundant pages are recovered, it is not unusual for organizations to see memory savings totaling 15%-25% or more. These savings can be effectively leveraged to service additional x86 workloads, allowing for efficient over-commitment of memory in the host. However, the challenge arises when outside influences affect transparent page sharing's effectiveness, particularly with DRS and HA actions.

For instance, let's say a particular virtual machine has been allocated 3GB or RAM, with 2GB actually being utilized by the guest operating system. VMware ESX has identified 500MB out of the 2GB of

physical memory allocated as redundant in-memory pages. That 500MB represents a 25% savings from TPS and can be leveraged to bring additional virtual machines online without adversely affecting performance by inflicting virtual machine memory ballooning or VMkernel page swapping. This nirvana is then interrupted when the ESX Server's workload changes and DRS has identified that the virtual machine would perform better if it were migrated to a different host. A VMotion event transfers the memory state of the virtual machine from one host to another and as part of this process, all memory contents are synchronized (duplicated) on the destination host; however the 500MB in memory savings realized through TPS is lost because the destination ESX host has no information on the incoming memory contents and how those memory blocks correlate to the existing physical memory contents on the host. Typical VMotion and DRS operations may not have a significant impact, but a large VMotion event, such as placing a host into Maintenance Mode and moving all running virtual machines to other hosts in the cluster, can have a more significant impact.

This challenge is even more significant in cases where a VMware HA isolation event causes a large number of virtual machines to be started on another host in the cluster that is already highly utilized. On a host with 32GB of RAM running many virtual machines with a running 20% memory savings from TPS, this can represent almost 6 GB of memory that will need to be accounted for on a host that must power on these virtual machines.

Memory Capacity Planning in VMware HA/DRS Environments

With today's multi-core CPUs, memory is often becoming the limiting factor in VMware Infrastructure 3 deployments. Many organizations are relying too heavily on memory over-commitment and are provisioning VMware ESX hosts will less RAM than is optimal. Therefore, it is important to understand the resource utilization of the virtual machines running in the environment, as well as properly planning for capacity needs and if necessary, add additional RAM to support the workload demands.

VMware HA Capacity Planning

As discussed previously, when an HA failover event occurs all running virtual machines will be restarted on a single host, beginning with the virtual machines with the highest restart priority. If the host that is chosen to restart the virtual machines is already tight on memory resources, the new virtual machines' demands will place additional load on the host and may significantly impact performance by severely over-committing memory.

Memory over-commitment is a frequently touted feature that can be leveraged to increase the ROI of VMware infrastructure. Although this is true, it is critical to understand how over-committing memory may affect situations when HA must restart a series of virtual machines on already-overcommitted hosts. The most effective way to extrapolate useful information on which to interpret HA failover scenarios is to leverage the resource utilization/performance counters in VirtualCenter and on the VMware ESX hosts.

Understanding Key Performance Counters Available in VirtualCenter

The following table lists some key memory performance counters in VirtualCenter that are helpful in determining cluster capacity requirements. These counters reference both per-virtual machine and per-ESX host counters available in VirtualCenter. Although many reference per-virtual machine examples, understand that the relevance for these counters is when viewed at the VMware ESX hosts level.

VirtualCenter Counter	Definition
Memory Unreserved	Total unreserved RAM in the host. Unreserved memory has not yet been specifically reserved for a virtual machine.
Memory Reserved Capacity	Amount committed to reservations at the VM-level directly. This counter does not reference *Memory Overhead*, even though memory overhead is held as a reservation by the VMkernel; this counter only reflects administratively-configured pre-virtual machine reservations. This counter is referenced by HA to determine how many virtual machines, based on their individual reservations, can be started on this host.
Memory Granted	The cumulative amount of RAM granted to all running virtual machines by the VMkernel. The amount of memory granted to a particular virtual machine by the VMkernel is based on several factors including the guest operating system type, configured memory reservation, working set, actual memory usage, etc. Memory granted to a virtual machine will typically equal the configured RAM for the VM plus virtualization overhead, provided memory is not overcommitted in the host.
Memory Consumed	The total amount of RAM used on this host. This is roughly derived as *(Memory Granted + Memory Overhead) - Memory Shared*
Memory Shared	This is the total NET savings of RAM on the host through transparent page sharing, including all zero-byte pages. Remember, transparent page sharing takes time to locate duplicate memory blocks so Memory Shared will be very low when a virtual machine or host starts and will gradually increase as transparent page sharing identifies sharing opportunities.
Memory Overhead	This is the amount of RAM allocated for virtualization overhead. Virtualization overhead is RAM that is required by the VMkernel to handle the actual hardware virtualization process. Overhead is a fixed size based on the guest operating system type and the memory configured in the virtual machine's settings. Memory Overhead is automatically held as a reservation on the host, although it is not reflected in *Memory Reserved Capacity*.
Memory Zero	This is the amount of RAM in zero pages; this number is included as part of *Memory Shared*. This number is a useful indication of a significant over-allocation of RAM to particular virtual machine, meaning the RAM contains no binary data and is not being used by the guest operating system.
Memory Balloon	This is the amount of memory actively recovered through the ballooning process.
Memory Swap Used	This is the amount of memory actively recovered through paging to the VM swap file (.vswp)
Memory Active	This is the amount of RAM that is actively used on the server, executing threads on host CPUs
Memory Usage (%)	This is a percentage representation of used RAM on the system. It is calculated by dividing *Memory Consumed* by the total amount of RAM in the host. This may not be an accurate representation of memory utilization as it is based on total RAM in the host, not the amount of RAM available to the virtualization of guests.

Table 1

There are a few other memory statistics that are of interest in Table 2, but they can only be obtained by viewing the VMware ESX host's configuration screen in the VMware Infrastructure Client.

Parameter	Definition
Total	This is the total physical RAM in the host as seen by VMware ESX.
System	This is the total amount of RAM allocated specifically to the VMkernel. This should not be confused with virtualization overhead, which is not included in this number. System memory is fixed, based on the total RAM in the host, ESX version, etc.
Service Console	This is the total RAM configured for the Service Console operating system.
Virtual Machines [Memory]	This is the remainder of the physical memory available to virtualization efforts. This is calculated by subtracting both *System* memory and *Service Console* memory from the *Total* RAM in the host.

Table 2

Armed with an understanding of the key counters, the next step is to insert some real-world statistics and observe how memory utilization, over-commitment and HA failover events can affect VMware ESX hosts.

Table 3 below contains actual real-world memory statistics from two VMware ESX hosts running in a production environment. The counter statistics were captured at a specific point in time on both hosts. Each of the hosts in this example were HP DL-585 G1 servers with (4) dual-core AMD processors and 32GB RAM.

ESX Host Parameter	Host A	Host B
Total [RAM in host]	32,604	32,604
System	2,458	2,458
Service Console	272	272
Virtual Machines	29,780	29,870

Table 3

As seen in the table, both VMware ESX hosts contain identical RAM, and because they are identical hardware, also share the same RAM commitment for the VMkernel (2,458 MB) and the same Service Console memory allocation (272 MB). There is approximately 29.5 GB of memory remaining for the virtualization of x86 workloads (29,870 MB).

Table 4 contains real-world examples from the same hosts using the counters listed in Table 2.

VirtualCenter Counter	Host A	Host B
Memory Unreserved	28,235	28,173
Memory Reserved Capacity	0	0
Memory Granted	23,332	21,248
Memory Consumed	20,336	16,344
Memory Shared	4,687	6,615
Memory Overhead	1,496	1,561
Memory Zero	2,613	4,116
Memory Balloon	0	0
Memory Swap Used	0	0
Memory Active	1,613	1,282
Memory Usage (%)	62.3	50.5

Table 4

The first interesting statistic is that Host A and Host B each have differing *Memory Unreserved* counters (28,235 MB and 28,173 MB, respectively), while they both have no memory reservations set on any running virtual machines, as noted in *Memory Reserved Capacity* (0 MB). This is because each host has a different commitment to *Memory Overhead* (1,496 MB and 1,561 MB, respectively) due to different virtual machines running on each host.

The next three counters of note are *Memory Granted*, *Memory Consumed* and *Memory Shared*. These three counters provide a good observation point of the actual resource requirements of the virtual machines running on the hosts.

Memory is granted to virtual machines by the VMkernel based on several factors, including the operating system type and version, amount of memory configured for the virtual machine and the virtual machine's working set. Typically, Memory Granted will equal the amount of RAM configured for the virtual machine plus the memory overhead, provided host memory isn't overcommitted.

Memory Consumed is the actual physical memory used in the host for virtual machines and is computed using three other counters; *Memory Granted*, *Memory Shared* and *Memory Overhead*. *Memory Consumed* is calculated by adding the memory granted to virtual machines to the memory overhead for the virtualization effort, and then subtracting the total memory recovered through transparent page sharing:

```
Memory Consumed = (Memory Granted + Memory Overhead) – Memory Shared
```

Calculating HA Failover Effects on Memory

The next step is to use this information to predict how the failure of a single VMware ESX host will affect the other hosts in the cluster.

The following table (Table 5) lists several statistics that are the result of some calculations performed on the VirtualCenter counters illustrated previously to determine how an HA failover will affect the memory resource needs of the other hosts in the cluster. These numbers are critical when identifying whether adequate capacity exists in the cluster to effectively handle the failure of a host and are explained in further detail below.

Calculated Parameter	Host A	Host B
Total Memory Liability	24,829	22,809
Remaining Free Memory	5,041	7,061
Available RAM in Host	9,729	13,676
Available Capacity %	32.6%	45.8%
Total Cluster Capacity % (Optimized)	121.6%	
Opposing Host Failover Commitment	43,145	41,172
Over-Subscription %	44.4%	37.8%
TPS Recovery %	18.9%	29.0%

Table 5

Total Memory Liability

This first statistic is the cumulative amount of RAM consumed by the ESX host for the virtualization effort is calculated by taking *Memory Granted* and adding *Memory Overhead*. This represents how much RAM a host will need to effectively power on all of this host's running virtual machine on another host if an HA failover occurred. This number does not take into account memory saved through transparent page sharing as this process takes time to reclaim memory and will not be available for recently powered-on virtual machines.

Remaining Free Memory

This statistic represents how much free RAM is in the ESX host when transparent page sharing is not taken into consideration. This value is calculated by subtracting the *Total Memory Liability* from the *Virtual Machines* ESX Host memory parameter in Table 3.

Available RAM in Host

This statistic represents the available capacity of the ESX host to power on new virtual machines, without considering failover. It is calculated by taking the free RAM in the ESX host (from the previous statistic *Remaining Free Memory*) and adds in the memory recovered thought transparent page sharing (*Memory Shared*). For HA capacity calculations, this number is used to identify the anticipated VMware ESX host memory utilization of the other host once transparent page sharing has optimized physical memory, assuming the same memory savings.

Available Capacity %

This value takes the previous statistic *Available RAM in Host* value and divides it by the *Virtual Machines* ESX Host memory parameter in Table 3 to obtain a spare capacity percentage for this host only.

Total Cluster Capacity %

The total optimized cluster capacity calculates the memory utilization percentage of each host and adds them together. This identifies how subscribed the cluster is when "optimized" by taking into account transparent page sharing. If this value is over 100%, then most likely the cluster is oversubscribed, although this isn't necessarily an adverse condition. Many production hosts can perform adequately with moderate oversubscription.

Opposing Host Failover Commitment

This statistic represents the memory demands that will be placed on the host if a failure occurs and this host must power on all virtual machines running on the other host in the cluster. This number is derived by taking the *Memory Consumed* value and adding in the other host's *Total Memory Liability* value. Because transparent page sharing will not complete its optimization process immediately, memory saved through TPS on the opposing host cannot be considered.

If this value is larger than the *Virtual Machines* ESX Host memory parameter in Table 3, then memory ballooning will occur when these hosts come online, and possibly could even invoke swapping if the demand is high enough.

Over-Subscription %

This number indicates how oversubscribed, as a percentage, the host is in the event of an HA failover. The number here indicates host much RAM will need to be recovered by ballooning or swapping. It is calculated by taking the over-committed RAM amount and dividing it by the *Virtual Machines* ESX Host memory parameter in Table 3.

TPS Recovery %

This statistic is just for reference, and is a calculated by taking the amount of *Memory Shared* and dividing it by the sum of *Memory Granted* and *Memory Overhead*. It shows the percentage of RAM saved by transparent page sharing, in relation to the amount of RAM in use on the host.

Calculating Anticipated Memory Load

Once the counters and calculations are understood, the next step is to put them to use. Anticipating capacity in an HA cluster can be performed simply by reverse-calculating the cluster capacity based on the worst-case failure scenario. The following tables offer a few examples of multi-node clusters and the maximum normal memory utilization when all hosts are available.

Table 6 shows the maximum capacity that a single host should exhibit when the desired maximum load on any one host is no more than 100% when no more than one VMware ESX host in the cluster fails. Table 7 shows the same capacity but with an allowance of two host failures concurrently. Notice how the individual maximum memory utilization numbers change significantly as the number of host failures allowed rises.

Maximum Host Memory Utilization During Failure Condition 100% Maximum Number of Host Failures Permitted in the Cluster. 1									
No. Hosts in Cluster	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Individual Host Memory Use Maximum	**50.0%**	**66.7%**	**75.0%**	**80.0%**	**83.3%**	**85.7%**	**87.5%**	**88.9%**	**90.0%**

Table 6

Maximum Host Memory Utilization During Failure Condition 100% Maximum Number of Host Failures Permitted in the Cluster. 2									
No. Hosts in Cluster	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Individual Host Memory Use Maximum	**NA**	**33.3%**	**50.0%**	**60.0%**	**66.7%**	**71.4%**	**75.0%**	**77.8%**	**80.0%**

Table 7

Table 8 and Table 9 show the same maximum host failure threshold relationship, but when allowing a maximum memory utilization of 120% when the allowed hosts have failed.

Maximum Host Memory Utilization During Failure Condition 120% Maximum Number of Host Failures Permitted in the Cluster. 1									
No. Hosts in Cluster	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Individual Host Memory Use Maximum	**60.0%**	**80.0%**	**90.0%**	**96.0%**	**100.0%**	**102.9%**	**105.0%**	**106.7%**	**108.0%**

Table 8

Maximum Host Memory Utilization During Failure Condition 100% Maximum Number of Host Failures Permitted in the Cluster. 2									
No. Hosts in Cluster	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Individual Host Memory Use Maximum	**NA**	**40.0%**	**60.0%**	**72.0%**	**80.0%**	**85.7%**	**90.0%**	**93.3%**	**96.0%**

Table 9

In all of the above tables, notice how the increase in added memory capacity over the previous column diminishes as the host count in the cluster rises. For instance, in Table 8, the maximum allowed individual host utilization rises 20% from 60% to 80% when going from a two- to a three-node cluster. However, the difference in capacity increase from a seven-node cluster to an eight-node cluster is only 2.1%. This is because larger clusters are more efficient at handling the load from a failed host as the liability can be spread over more surviving hosts. Figure 3 shows a graphical representation of the cluster capacity increase as illustrated in Table 8.

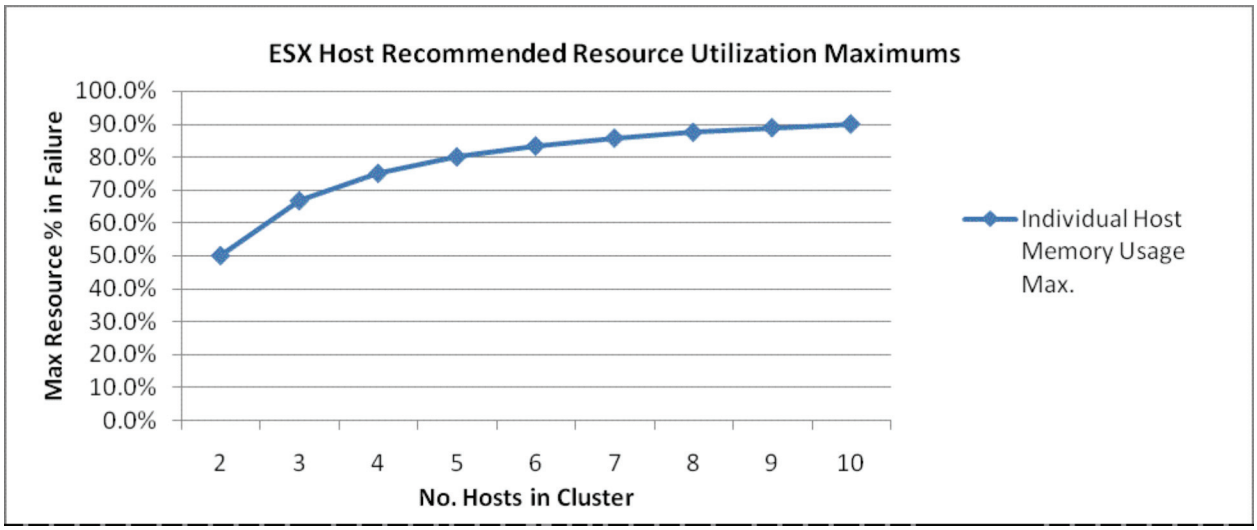

Figure 3

Leveraging Reservations

A key factor in HA capacity planning is in understanding how the cluster will react when a failure occurs and what impact there may be to the existing x86 workloads.

When VMware HA determines that a host has been isolated from the rest of the cluster, it determines on which host to begin restarting the virtual machines that were running on the failed or isolated host. Once chosen, all virtual machines will be started on that host, up to the host's ability to satisfy resource reservations. Once the host can no longer guarantee reservations, another host will be selected to continue the power up process.

Without defining explicit memory resource reservations at the virtual machine-level, VMware HA has no way to ascertain what the demands of the virtual machines currently are or will be when restarted on another host; remember, there is no inter-communication between DRS and HA, so HA has no concept of Resource Pool reservations. If reservations are not defined on individual virtual machines, when a host in an HA cluster fails ***all*** virtual machines from that host will be restarted on a ***single*** VMware ESX host. HA does not "load balance" the process of starting virtual machines across hosts. This can leave the cluster in a highly unbalanced state and can severely impact the performance of virtual machines that were running on the selected failover host prior to the failure. The only way VMware ESX can gauge capacity is through the use of per-virtual machine reservations.

Virtual Machine Reservations

The purpose of reservations is to guarantee a designated amount of resources to a given workload. That workload may be either a logical grouping of virtual machines, such as a DRS Resource Pool, or a specific virtual machine. VMware Infrastructure 3 supports two different kinds of reservations; those configured on a resource pool and those assigned to individual virtual machines.

To maximize the effectiveness of VMware HA, memory reservations should be configured on a per-virtual machine basis as this is the only way VMware HA can determine if adequate resources exist in the cluster to support the workload demands and alert administrative personnel when resource demands can no longer be satisfied.

Determining Memory Reservation Needs

When setting memory reservations at the virtual machine level, the reservation should be configured based on the amount of memory required for the workload to perform satisfactorily. This number may be determined by a number of factors specific to the environment, including the following:

Minimum Memory Required

Most likely, this will be determined by observing statistics either within the virtual machine itself, such as Windows Performance Monitor, or observing specific counters in VirtualCenter about the behavior of the virtual machine. For instance, a high *Memory Zero* counter in VirtualCenter may indicate that the virtual machine has been configured with far too much RAM. Likewise, if excessive paging is observed within the guest operating system, the virtual machine may not be performing optimally. Finding what the correct workload demand is for the virtual machine will allow the assignment of an adequate reservation.

Service-Level Agreements

Some organizations may choose employ reservations as a means to guarantee specific SLAs, or *Service-Level Agreements*. An SLA may be configured on an assumed performance or tier level as follows:

- **Tier 1** – Reservations set at 100% of configured memory
- **Tier 2** – Reservations set at 50% of configured memory
- **Tier 3** – No memory reservations set.

The reservation number is derived by determining what the minimum amount of RAM is required to run the workload in accordance with performance guarantees or SLA. If using percentages as a rule, it is important to provision the correct amount of memory for the virtual machine. Arbitrarily configuring too much RAM on a VM will only contribute to an inefficient platform, as will under-sizing RAM requirements or reservations.

Environment Type

Specific environment types, such as Production, Development, Test, QA, etc. may dictate specific needs for reservations. For instance, Production virtual machines may require a certain reservation, but development machines can run in a reduced capacity and may not have any reservations configured.

Chargeback Models

Another common model for reservation use is through business unit or departmental chargeback modeling. In these cases, the amount charged back to a specific business unit or department may be based on the ***guaranteed resources available*** (a.k.a., the configured reservation). A typical example might be a base chargeback for the virtual machine, plus a premium for specific reservations guarantees.

Regardless of the model used, reservations should be set at the virtual machine level to allow VMware HA to properly understand the workload demands of the cluster.

VMware HA and DRS Best Practices

When planning to leverage VMware HA, it's critical to ensure there is complete network redundancy in all paths between hosts in the cluster. The greatest risk with VMware HA implementations is a false isolation event where an ESX host is incorrectly identified as being offline, triggering an isolation response. To mitigate the risk, ensure that Service Console network connection has more than one path, or that more than one Service Console connection is available to transmit/receive HA heartbeats. Also ensure that the default gateway IP address is a live, reachable address, or set the cluster's *das.isolationaddress* to a reachable IP address. When a host believes it is isolated, it will check connectivity to these addresses to confirm network isolation.

Another option to mitigate false isolation detection due to intermittent network connectivity issues is to increase the amount of time for the isolation response to initiate. This is controlled through a new VMware HA advanced configuration setting that was enabled in VirtualCenter 2.0.2 called *das.failuredetectiontime*. Setting this to 60 seconds may prevent slight intermittent network connectivity issued from triggering an isolation response prematurely; however it will extend the recovery time of the virtual machines by the configured time. Both *das.isolationaddress* and *das.failuredetectiontime* are configured in the *Advanced Options* of the VMware HA cluster settings.

Furthermore, if the physical network switches providing network connectivity to the Service Console interface(s) support *PortFast* or equivalent, enable it such that connectivity is immediately available to avoid isolation events triggered by spanning tree loop detection.

Also remember that VMware HA requires properly functioning FQDN name resolution. Make sure DNS is configured properly on all VMware ESX hosts in the cluster and that host records are created in the appropriate DNS zone for all VMware ESX servers.

Avoid using Strict Admission Control in two-node HA clusters. When one of the two nodes fail, the cluster is automatically out of compliance and although HA failed-over virtual machines will power up automatically on the surviving host, an administrator will not be able to manually power on virtual machines while the other host in the cluster is down.

Ensure adequate memory has been provisioned on the hosts in the cluster. Monitor performance counters and overall physical memory utilization on the hosts to ensure there is adequate spare memory available to handle the demands when a host fails.

Leverage per-virtual machine memory reservations to enable the HA cluster to alert administrative personnel when the demands of the workloads can no longer be satisfied by available capacity. Also, configuring strict admission control will prevent new virtual machines from coming online that might otherwise prevent the cluster from maintaining appropriate SLAs.

If possible, configure DRS for *fully-automated mode* when used in combination with VMware HA. Remember, although DRS and HA are two independent services in VI3 and do not directly interoperate, DRS can quickly rebalance the cluster if an HA failover event occurs and the cluster is left out of balance. In addition, DRS will automatically migrate virtual machines to honor affinity/anti-affinity rules that the HA failover may have violated. For specific use cases, individual virtual machines can be configured so they are not automatically migrated in the cluster by setting them to either partially-automated or manual mode in the cluster configuration.

Use DRS affinity rules to keep appropriate virtual machines together on the same host and to separate others, as required. Some virtual machines may benefit from running on the same ESX host, such as VMs with high inter-virtual-machine network traffic. Likewise, DRS affinity rules can keep certain virtual machines apart, such as infrastructure servers that are redundant of one another, or particularly CPU or memory-intensive VMs. Furthermore, understand that DRS is unable to monitor disk and network utilization. If certain virtual machines are particularly disk- or network-intensive, such as backup servers, affinity rules can be used to separate similar I/O-intensive workloads.

Finally, VMware ESX host hardware in the cluster should be as homogeneous as possible, specifically in terms of memory capacity, CPU clock speed, and core count. Because DRS relies on VMotion to migrate running workloads, all hosts must be VMotion compatible.

Summary

VMware HA and DRS are features that enable a truly enterprise-ready virtualization platform. As the deployment scale grows and matures, it is critical to both understand and monitor resource utilization needs across the cluster and to be able to properly plan and maintain capacity requirements for the deployed x86 workloads.

Memory is a key enabling factor in VMware Infrastructure environments, specifically where DRS and HA are leveraged. As part of any sound capacity planning strategy, memory plays a key part in the overall health of the virtual infrastructure and must be monitored continually to ensure adequate resources are available to support the workload demands. Finally, ensuring there is adequate capacity to sustain

individual host failures will prevent undue loss of time, money and productivity and enable the organization to leverage virtualization to the highest levels of efficiency.

About the Environment Used

The information in this paper was derived from a production environment consisting of 60 ESX 3.0.1 Servers running on HP DL-685c blade hardware with (4) dual-core AMD Opteron processors running at 2.4 GHz and 32GB of RAM each. Each host was running between sixteen and eighteen virtual machines with an overall host processor utilization of 30%.

About the Author

Michael Burke, VCP, MCP, is the Virtualization Practice Director for a leading Virtual Infrastructure Services company headquartered in the Northeast providing virtualization design and integration services nationwide. Burke has over five years experience working closely with VMware products. He wrote several technical article and whitepapers in addition to being a technical editor for the book: "VMware ESX Server: Advanced Technical Design Guide."

Printed in the USA MKMS – 1061