

UNDERSTANDING vSPHERE DRS PERFORMANCE

VMware vSphere 6

Table of Contents

Introduction	4
How DRS Works	4
Effective VM Placement	4
Efficient Load Balancing	5
Calculating VM Resource Demand	5
Detecting VM Demand Changes	5
Cost Benefit Analysis.....	5
Factors That Affect DRS Behavior	6
DRS Automation Levels.....	6
DRS Aggression Levels (a.k.a. Migration Threshold)	6
VM Overrides	7
VM/Host Rules.....	8
Reservation, Limit, and Shares	11
Reservation.....	11
Limit.....	11
Shares.....	11
Advanced Options.....	12
Troubleshooting and Monitoring	13
Recommendations.....	14
History.....	15
DRS Faults.....	15
CPU Utilization Summary	16
Memory Utilization Summary	16
Case Studies	16
Case Study 1 - DRS Provides Effective VM Initial Placement	16
Expectation	17
Observation	18
Case Study 2 - Memory Considerations for DRS Load Balancing (VM Consumed vs. Active Memory).....	18
Expectation	20
Observation	21
Case Study 3 - Cluster Balance Impacted by VM Overrides.....	22
Expectation	24
Observation.....	24

Case Study 4 - Guaranteeing Resources for VMs.....	24
Expectation	25
Observation	26
Case Study 5 - Resource Prioritization Using Shares.....	26
Expectation	27
Observation	30
Case Study 6 - How DRS Migration Threshold Impacts Cluster Balance	30
Observation	32
Case Study 7 - DRS Considers Prerequisite Moves for Placement.....	32
Expectation	33
Observation	33
Conclusion	34
References	34

Introduction

VMware vSphere® Distributed Resource Scheduler™ (DRS) is the resource scheduling and load balancing solution for vSphere. DRS works on a cluster of ESXi hosts and provides resource management capabilities like load balancing and virtual machine (VM) placement. DRS also enforces user-defined resource allocation policies at the cluster level, while working with system-level constraints.

Although DRS is widely deployed and generally understood, questions about "how" DRS does what it does are not uncommon. Not knowing exactly how DRS works often leads to confusion and improper expectations about DRS behavior and its performance.

In this paper, we demystify DRS. First, we give an overview of how DRS works. Next, we explain some of the factors that influence DRS behavior and how they can be monitored. Finally, we cover some common performance scenarios that customers raised as issues and show what we can learn about DRS from these scenarios.

How DRS Works

The main goal of DRS is to ensure that VMs and their applications are always getting the compute resources that they need to run efficiently. In other words, *DRS strives to keep your VMs happy*¹. It does this by ensuring that newly powered-on VMs get all the required resources soon after they are powered on, and the resource utilization is always balanced across the cluster.

From time to time, VMs' workloads may change, and with many VMs with changing workloads, there can be imbalance in the cluster. Each of these can degrade application performance. DRS solves these problems by regularly monitoring the cluster balance state once every five minutes, by default, and then takes the necessary actions to fix any imbalance. DRS automatically determines which virtual machines would benefit from a move to another host and live migrates the VM onto the new host using vMotion. In this way, DRS ensures each virtual machine in the cluster gets the host resources—like memory and CPU—that it needs.

Let's take a closer look at how DRS achieves its goal of ensuring VMs are happy, with effective placement and efficient load balancing.

Effective VM Placement

When a VM is being powered up in a DRS cluster, DRS runs its algorithm to determine the right ESXi host for it to be powered up on. This decision, also known as VM placement (or initial placement) is made based on the expected change in resource distribution (after ensuring that there are no constraint violations if the VM was placed on the host).

One of the first steps in ensuring good VM performance is to make sure that the VM gets all the resources it needs as soon as it is powered on. DRS considers the demand of a VM, so it will never be short of resources whenever it is started. A VM's **demand** includes the amount of resources it needs to run, and the way DRS calculates this is described in [Calculating VM Resource Demand](#).

¹ For VMs and their applications to perform well, they rely on DRS to provide the necessary resources. In reality, application performance depends on more than just the availability of resources. DRS can only ensure that lack of resource availability is not the reason for any application performance issues in your cluster.

VM placement is a key aspect of DRS performance and must be effective because this reduces the amount of load balancing that is required. When DRS chooses the optimal host with the closest match of resources, and it does not disturb the balance of resources across the cluster, this is effective VM placement.

Even with optimal placement, VM workloads may change over time, and there could be some imbalance in the cluster from time to time. This is when load balancing is required.

Efficient Load Balancing

When the host resources in a cluster are more or less evenly utilized, then the cluster is well balanced. DRS uses a cluster-level balance metric to make load-balancing decisions. This balance metric is calculated from the standard deviation of resource utilization data from hosts in the cluster. DRS runs its algorithm once every 5 minutes (by default) to study imbalance in the cluster. In each round, if it needs to balance the load, DRS uses vMotion to migrate running VMs from one ESXi host to another.

Calculating VM Resource Demand

In calculating the resource utilization, DRS looks for the demand for every running VM in the cluster. VM **demand** is the amount of resources that the VM currently needs to run. For CPU, demand is calculated based on the amount of CPU the VM is currently consuming. For memory, demand is calculated based on the following formula.

VM memory demand = Function(Active memory used, Swapped, Shared) + 25% (idle consumed memory)

In other words, by default, DRS balances memory workloads based mostly on a VM's active memory usage, while considering a small amount of its idle consumed memory as a cushion for any increase in workload. This behavior enables you to efficiently run memory workloads in your DRS clusters, even with over-commitment.

Detecting VM Demand Changes

During each round, along with resource usage data, DRS also collects resource availability data from each and every VM and host in the cluster. Data like VM CPU average and VM CPU max over the last collection interval depict the resource usage trend for a given VM, while availability data like VM CPU Ready Time² and VM Memory Swapped³ indicate resource shortage, if any, for the VM (availability data indicate if a VM is running short of resources). DRS then correlates the resource usage data with the availability data and runs its load-balancing algorithm before taking necessary vMotion actions in order to keep the cluster balanced and to ensure that VMs are always getting the resources they need to run.

Cost Benefit Analysis

vMotion of live VMs comes with a performance cost, which depends on the size of the VM being migrated. If the VM is large, it will use a lot of the current host's and target host's CPU and memory for vMotion. The benefit, however, is in terms of performance for VMs on the source host, the migrated VM on the destination host, and improved load balance across the cluster.

² CPU ready time is the amount of time a VM was in a state where it was ready to execute, but there were not any CPU resources available. CPU ready time is usually a very good indicator of CPU contention in a host.

³ VM memory swapped is the amount of idle consumed memory that has been swapped out of a VM by the host.

Moving a VM to a host with better availability of resources and freeing resources on the source host means having more room for workload changes. The DRS algorithm constantly evaluates the cost and benefit of each load balancing vMotion move.

Another factor that comes into play in load balancing is the VM workload behavior. If a VM's application tends to follow a spiky trend, DRS tries to detect such spikes by looking at VM resource usage trends, as mentioned in the section [Detecting VM Demand Changes](#). Without this behavior, DRS could end up migrating VMs more often than is preferred.

Factors That Affect DRS Behavior

While DRS constantly works to ensure that VMs are getting the resources they need, it also provides several useful customizations that work very well for a variety of cluster needs. By understanding these customizations, you can get the best performance out of DRS and have it meet your expectations. In this section, we discuss some of the customizations and factors that affect DRS and how to use them for best performance.

DRS Automation Levels

During initial placement and load balancing, DRS generates placement and vMotion recommendations, respectively. DRS can apply these recommendations automatically, or you can apply them manually.

DRS has three levels of automation:

- **Fully Automated** – DRS applies both initial placement and load balancing recommendations automatically.
- **Partially Automated** – DRS applies recommendations only for initial placement.
- **Manual** – You must apply both initial placement and load balancing recommendations.

As shown in [Figure 1](#), you can set the DRS automation level in the client from **Cluster → Manage → Settings → Edit Cluster Settings**.

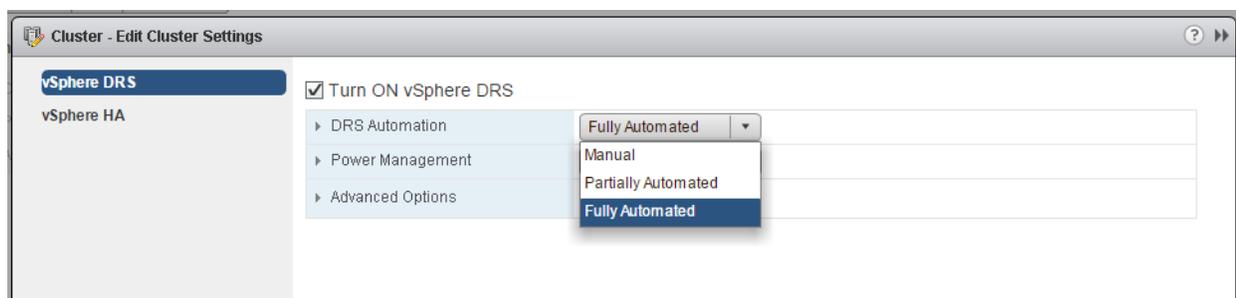


Figure 1 - DRS Automation settings

DRS Aggression Levels (a.k.a. Migration Threshold)

The DRS aggression level controls the amount of imbalance that will be tolerated in the cluster. DRS has five aggression levels ranging between 1 (most conservative) and 5 (most aggressive). The more aggressive the level, the less DRS tolerates imbalance in the cluster. The more conservative, the more DRS tolerates imbalance. As a result, you might see DRS initiate more migrations and generate a more even load distribution when you increase the aggression level. By default, DRS aggression level is set to 3. For most clusters, the default

aggression level of DRS works best. However, if you do need DRS to be more active in load balancing at the cost of increased live migrations, you can increase the DRS aggression level.

When DRS aggression is set to level 1, DRS will not load balance the VMs. DRS will only apply move recommendations that must be taken either to satisfy hard constraints, such as affinity or anti-affinity rules, or to evacuate VMs from a host entering maintenance or standby mode.

As shown in [Figure 2](#), you can set the migration threshold in **Cluster → Manage → Settings → Edit Settings → DRS Automation → Migration Threshold**.

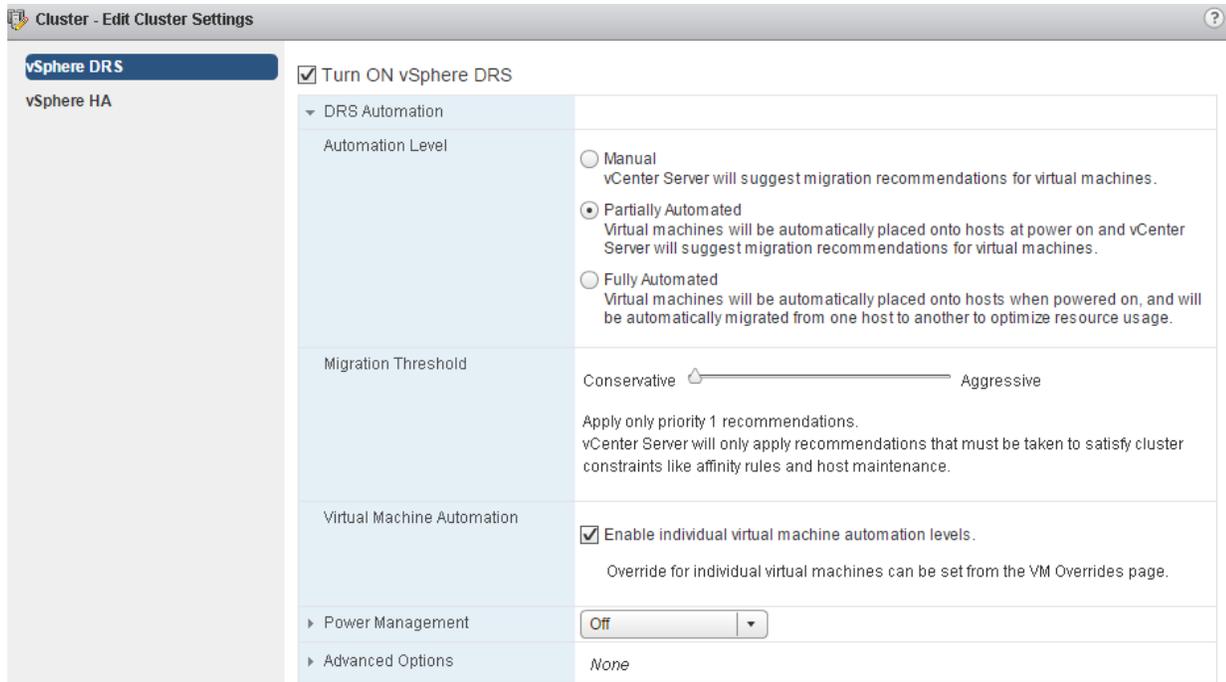


Figure 2 - Migration Threshold settings

You can find more details on how migration threshold affects DRS behavior in [Case Study 6 – How DRS Migration Threshold Impacts Cluster Balance](#).

VM Overrides

DRS automation levels and migration threshold are normally applied at the cluster level. In some cases, you might require DRS to treat some VMs specially. For example, you might decide DRS should not consider a specific VM when generating its recommendations, or you might decide DRS should not migrate that VM at all.

You can set VM overrides under **Cluster → Manage → Settings → VM Overrides**, as shown in [Figure 3](#). Here you can set the automation or migration threshold for a VM to a value different than that at the cluster level, or even disable them.

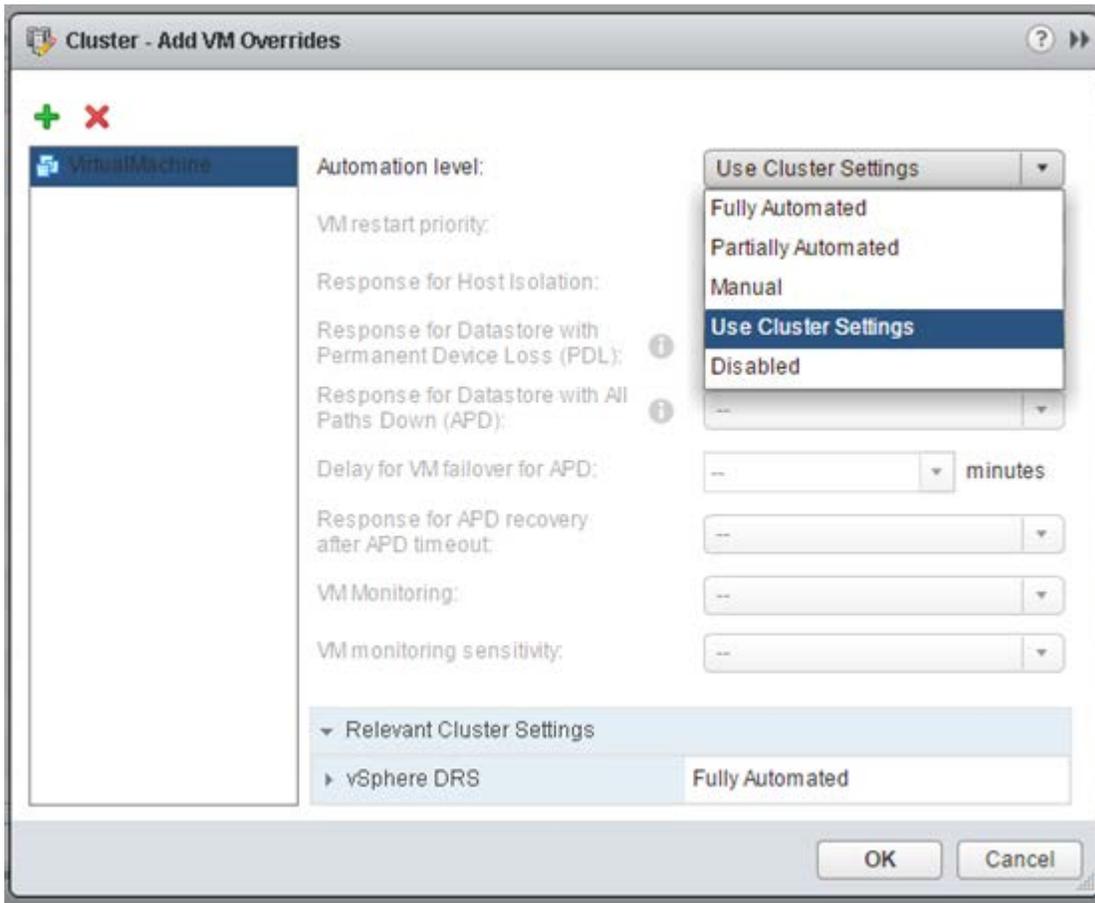


Figure 3 - VM Overrides

More details on how virtual machine overrides affect DRS behavior is explained in [Case Study 3 - Cluster Balance Impacted By VM Overrides](#).

VM/Host Rules

In a fully automated cluster, DRS generally has complete control over where every VM in the cluster is placed, and where it will continue to run. While this is ideal for DRS to work, there may be situations where you might want to run specific VMs on specific hosts, or always keep some specific VMs together. For example, you might want to keep an application server VM and its database VM together on the same host for performance reasons. Or if there are VMs that are running a specific type of application that runs best on a certain kind of hardware, you might want to make sure that such VMs always run on hosts that have that kind of hardware. This is when DRS VM/Host Rules (or just Rules) can be useful.

Rules help define special conditions on VMs and/or hosts in a DRS cluster. Once a rule is set, DRS has to honor it, and has to make recommendations in accordance to the rule, along with its placement and load balancing logic.

There are different types of rules that can be set:

1. **Keep Virtual Machines Together (VM-VM)**—This rule ensures that the VMs specified in the rule are always running on the same host.
2. **Separate Virtual Machines (VM-VM)**—This rule will keep the VMs in the rule always running on different hosts.
3. **Virtual Machines to Hosts (VM-Host)**—This type of rule is set on groups of one or more VMs and one or more hosts. A host or a VM group can be created in the web client, under **Cluster → Manage → Settings → VM/Host Groups**, as shown in [Figure 4](#) and [Figure 5](#). VMs/hosts can be part of multiple groups.

In VM-Host rules there are sub-rules of type **should** and **must**, as shown in [Figure 6](#). With these sub-rules, you can specify if a VM group should/must, or should not/must not run on a host group. Sub-rules of type **must** (mandatory) will always be honored by DRS under all circumstances. However, sub-rules of type **should** (preferential) are dropped if DRS determines that the imbalance in the cluster is very high.

VM-VM rules are always treated as mandatory moves, like the **must** type VM-Host rules. However, since these are DRS rules, vSphere HA (High Availability) [1] might violate these rules during failover. In such an event, DRS will fix the rule violation during its first invocation.

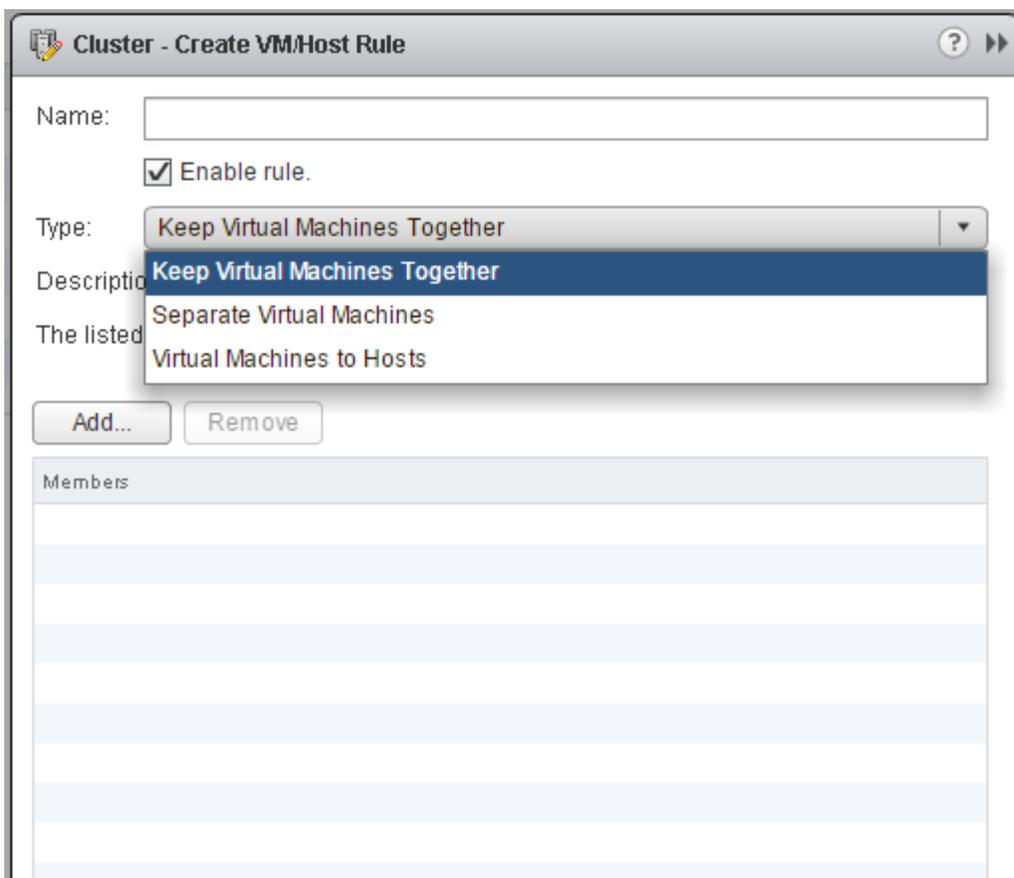


Figure 4 - Types of VM/Host rules

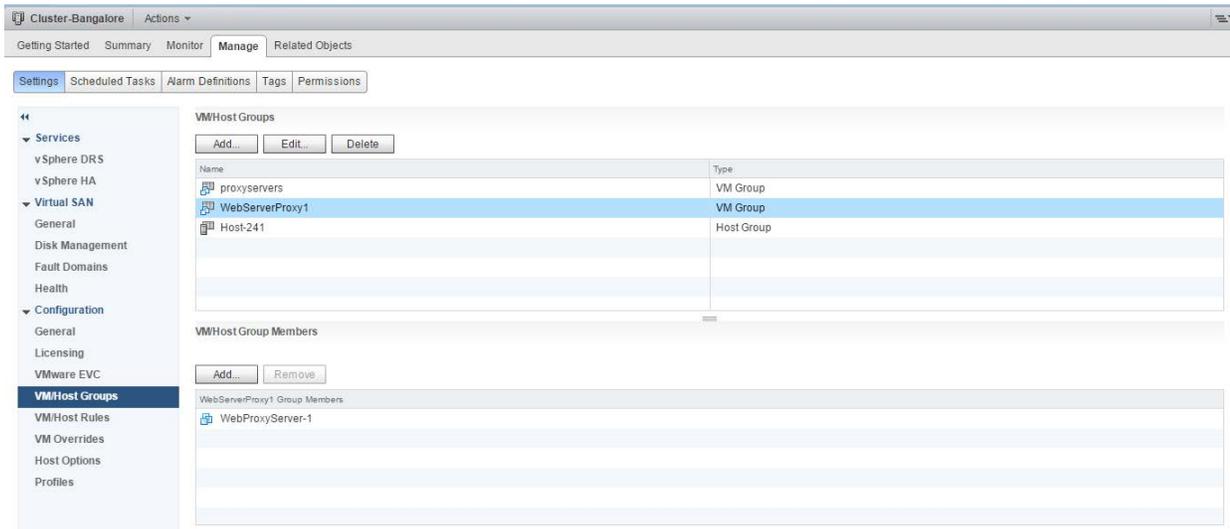


Figure 5 - VM/Host Groups settings

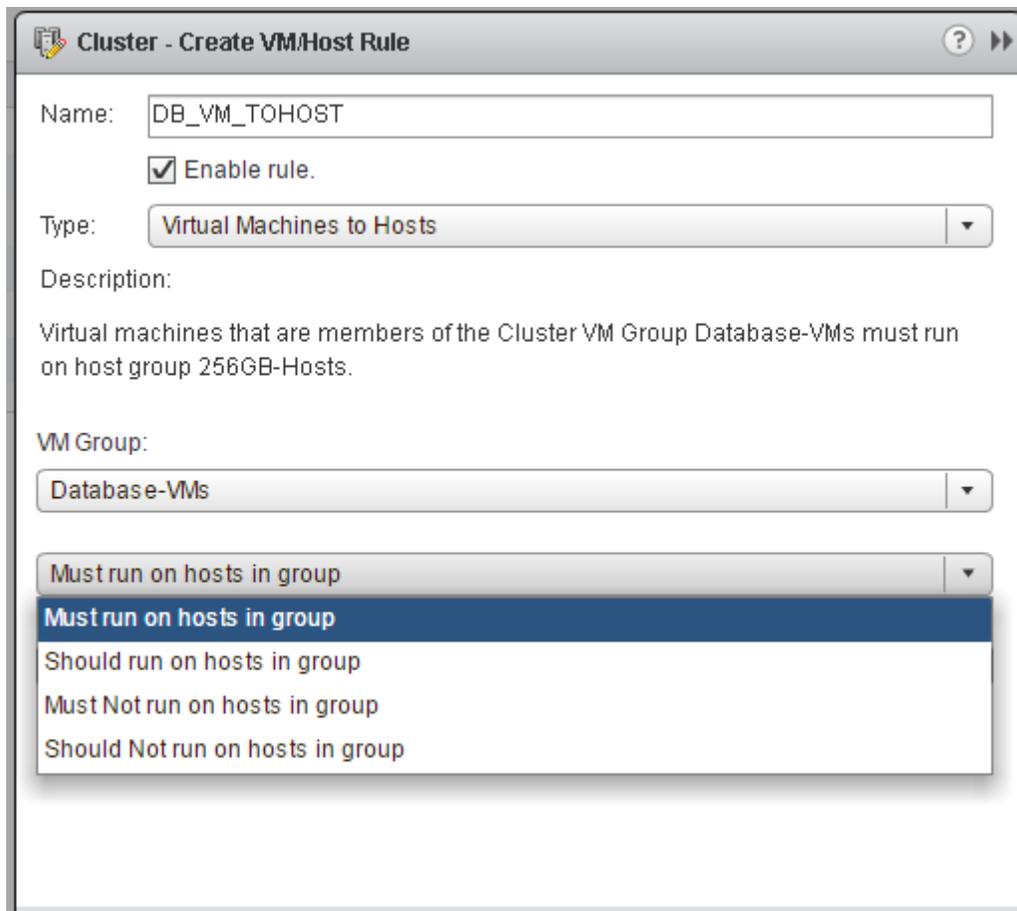


Figure 6 - VM/Host rules

Rules empower you to tune DRS behavior to work with critical virtual machines and workloads that require special settings. While rules are very useful, they tend to limit DRS to work with constraints, which can sometimes hinder its progress. For more information, see [Case Study 3 - Cluster Balance](#).

Reservation, Limit, and Shares

DRS provides many tools for you to customize your VMs and workloads according to specific use cases. Reservation, limit, and shares (as shown in [Figure 7](#)) are three such tools borrowed from ESXi's resource management paradigm.

Reservation

You might need to guarantee compute resources to some critical VMs in your clusters. This is often the case when running applications that cannot tolerate any type of resource shortage, or when running an application that is always expected to be up and serving requests from other parts of the infrastructure.

With the help of reservations, you can guarantee a specified amount of CPU or memory to your critical VMs.

Reservations can be made for an individual VM, or at the resource pool level. In a resource pool with several VMs, a reservation guarantees resources collectively for all the VMs in the pool.

Limit

In some cases, you might want to limit the resource usage of some VMs in their cluster, in order to prevent them from consuming resources from other VMs in the cluster. This can be useful, for example, when you want to ensure that when the load spikes in a non-critical VM, it does not end up consuming all the resources and thereby starving other critical VMs in the cluster.

Shares

Shares provide you a way to prioritize resources for VMs when there is competition in the cluster. They can be set at a VM or a resource pool level.

By default, a cluster has a resource pool hierarchy, with the root resource pool (the cluster itself) at the top, and all VMs as its children. Shares are defined as numbers for all the sibling VMs under this root resource pool. Shares are distributed equally, by default, on a per-resource basis (per-vCPU and per-unit of memory). This means that by default, a VM with more configured resources will get more shares than a VM with fewer resources. During resource contention, resources available at the root resource pool are shared among the children based on their shares' values.

DRS provides four types of shares for VMs and resource pools - **Low**, **Normal**, **High**, and **Custom** - to change their priority compared to their siblings. Normal shares are typically 2x Low, and High shares are typically 2x Normal. Custom can be used to set specific share values. When setting custom shares at a VM level, you need to account for all the vCPUs and memory of that VM, since shares are assigned based on the amount of configured resources of a VM.

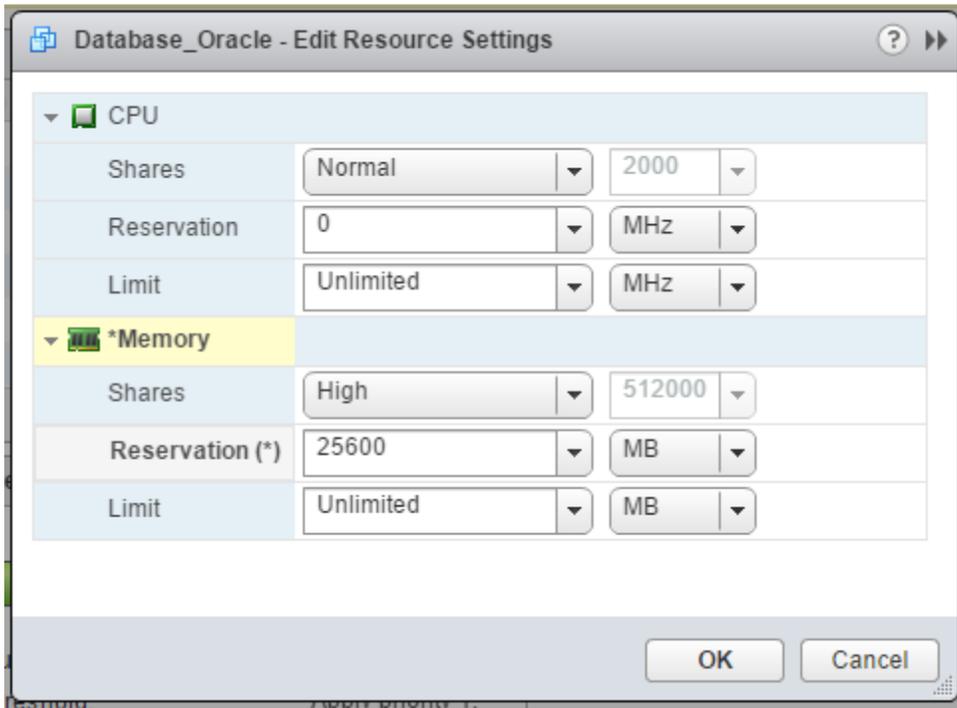


Figure 7 - VM-level resource settings

More details on how VM-level Resource Settings affects DRS behavior are explained in [Case Study 4 - Guaranteeing Resources for VMs](#) and [Case Study 5 - Resource Prioritization Using Shares](#).

Advanced Options

DRS generally works well with the default/recommended settings. However, not all clusters are the same, and some special cases might require specific customizations in DRS for best performance. DRS provides several advanced options to handle specific cluster requirements outside of recommended settings. Advanced options are specified as key-value pairs, as shown in [Figure 8](#).

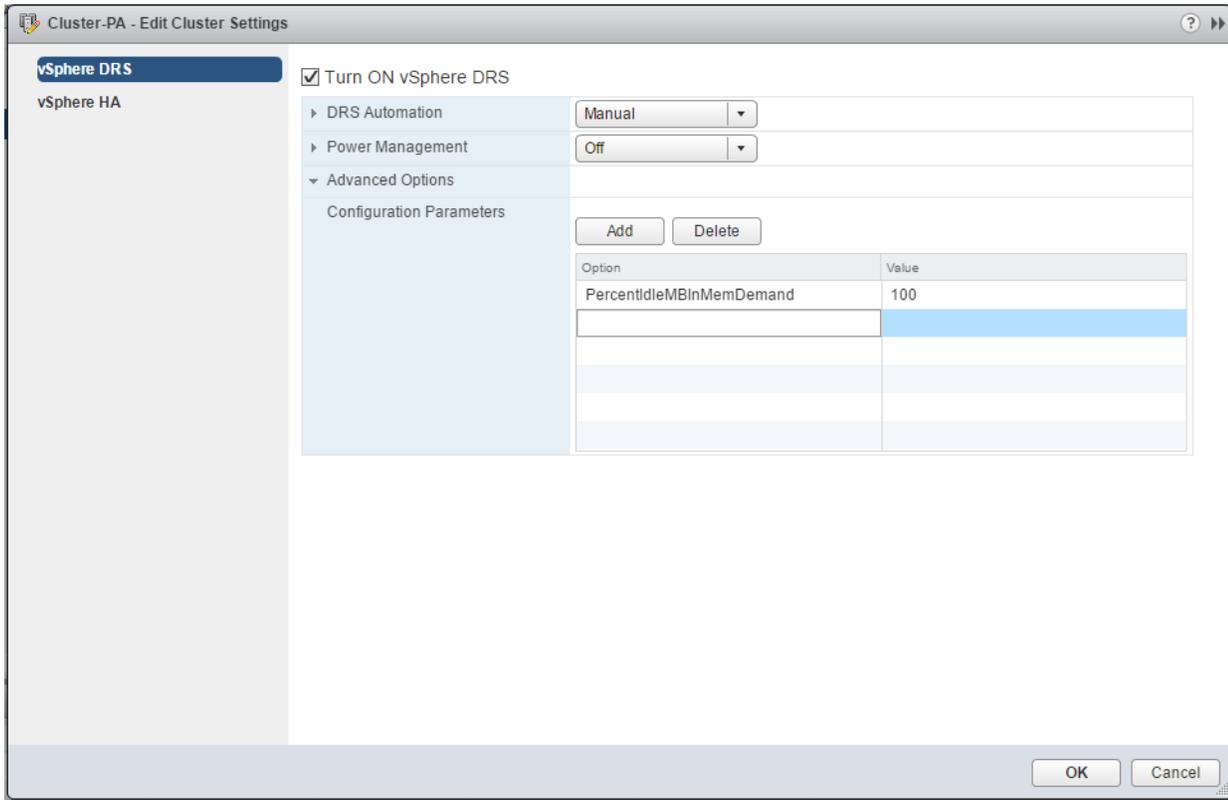


Figure 8 - DRS advanced settings

An example of how advanced settings can be used to change the default behavior of DRS is shown in [Case Study 2 - Memory Considerations for DRS Load Balancing \(VM Consumed vs. Active Memory\)](#).

Troubleshooting and Monitoring

The Summary tab shows the current state of any cluster imbalance, a summary of the DRS settings, and the number of recommendations and DRS faults generated.

The green bar (a bubble level) indicates cluster balance. As long as the bubble remains in the center of the bar, in between the vertical lines, the cluster is balanced. If it is not in the center and is outside of the two sets of vertical lines, the cluster is not balanced. Note that when the cluster is not balanced, the bubble always shifts to the right of the bar.

Hovering over the information (i) icon provides the Host Load Standard Deviation, which is the standard deviation of load across the cluster. Host Load Standard Deviation has two values: current and target. Current is the current standard deviation value, and target is the value at which the cluster is considered to be balanced (see [Figure 9](#)).

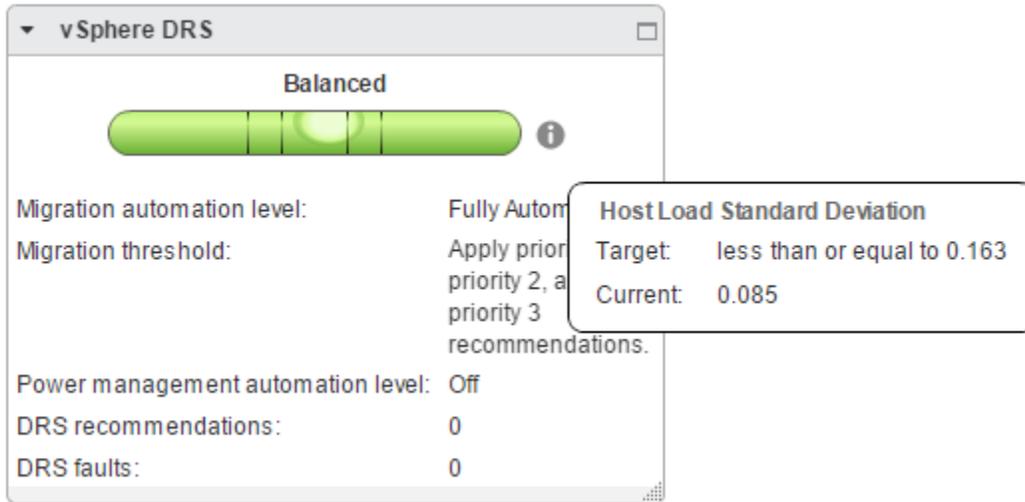


Figure 9 - DRS cluster status

Once we know the current status of the cluster in the Summary tab, we can look at the DRS Monitoring tab (**Cluster → Monitor → vSphere DRS**, as shown in [Figure 10](#)) to better understand the behavior of DRS and its decisions. This tab has multiple views for DRS recommendations, history, faults, and shows the CPU and memory utilization of the cluster, which can be used to troubleshoot DRS behavior.

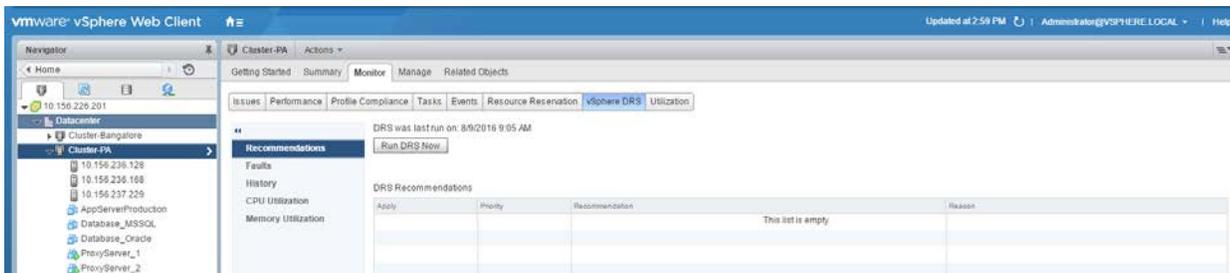


Figure 10 - DRS Monitor tab

The different views available and what information they show follows.

Recommendations

The Recommendations view ([Figure 11](#)) lists DRS recommendations that you must manually apply. When DRS is in partially automated mode or manual mode, you can see recommendations in this view. Here you can also see if there are recommendations for VMs with manual or partially automated VM override setting.

DRS was last run on: 8/9/2016 9:05 AM

Run DRS Now

Apply	Priority	Recommendation	Reason
<input checked="" type="checkbox"/>	3	Migrate ProxyServer_3 from 10.156.236.128 to 10.156.237.229	Balance average CPU loads
<input checked="" type="checkbox"/>	2	Migrate ProxyServer_2 from 10.156.236.128 to 10.156.236.168	Balance average CPU loads

Figure 11 - DRS Recommendations

History

The History view (Figure 12) of the DRS Monitor tab gives the history of DRS decisions. This page lists vMotion and initial placement actions taken by DRS. This view also shows actions like vMotion migrations that result when you apply manual recommendations.

Time	DRS Actions
Thursday, August 18, 2016 8:37:50 AM	Migrate ProxyServer_3 from 10.156.236.128 to 10.156.237.229
Thursday, August 18, 2016 8:37:48 AM	Migrate ProxyServer_2 from 10.156.236.128 to 10.156.236.168
Thursday, August 18, 2016 6:59:13 AM	Place ProxyServer_3 on host 10.156.236.168
Thursday, August 18, 2016 6:59:13 AM	Place ProxyServer_4 on host 10.156.237.229
Thursday, August 18, 2016 6:59:12 AM	Place ProxyServer_2 on host 10.156.236.128
Thursday, August 18, 2016 6:59:11 AM	Place ProxyServer_1 on host 10.156.236.128

Figure 12 - DRS History

DRS Faults

This tab (Figure 13) has information on DRS faults, which can be generated as a result of rules that DRS could not fix, or vMotion actions that DRS could not execute. Usually, faults indicate that DRS is unable to work efficiently due to reasons beyond its control.

Faults are not necessarily bad, but as a best practice, you should monitor this section and try to fix any DRS faults.

Issues Performance Profile Compliance Tasks Events Resource Reservation vSphere DRS Utilization

Time	Reason	Target Object
Friday, August 19, 2016 11:29:39 AM	Could not fix anti-affinity rule violation.	10.156.236.128

DRS Fault Details

Fault	Prevented Recommendation
This operation would violate a virtual machine affinity/anti-affinity rule.	Migrate ProxyServer_1 from 10.156.236.128 to 10.156.236.168
This operation would violate a virtual machine affinity/anti-affinity rule.	Migrate ProxyServer_4 from 10.156.236.128 to 10.156.236.168

Figure 13 - DRS Faults

CPU Utilization Summary

This view summarizes the CPU utilization of hosts in the cluster (Figure 14). For each host, you can see the total CPU consumed by all VMs running on that host. Hovering over the bar provides a tool tip with information on VM-level CPU utilization.

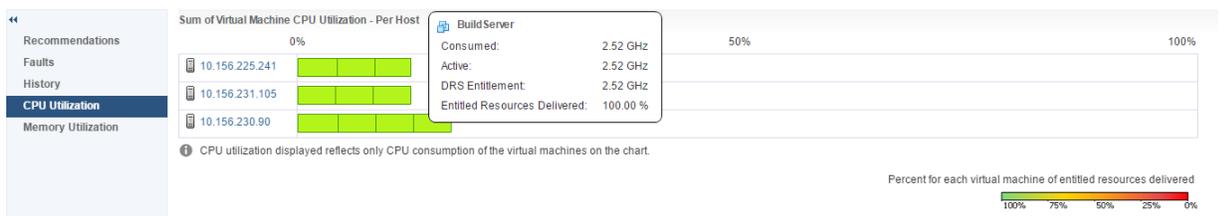


Figure 14 - DRS Cluster CPU Utilization

Memory Utilization Summary

This view summarizes the memory utilization of hosts in the cluster. For each host, you can see the total consumed memory⁴ from all VMs running on that host. As mentioned earlier, DRS load balances mainly using VM active memory⁵, and not consumed memory. This means that in the default setting, the cluster balance state in the DRS summary view might not have a direct correlation to the memory utilization across hosts shown in this view.

Hovering over the bar provides a tool tip with information on VM level memory utilization, as shown in Figure 15.

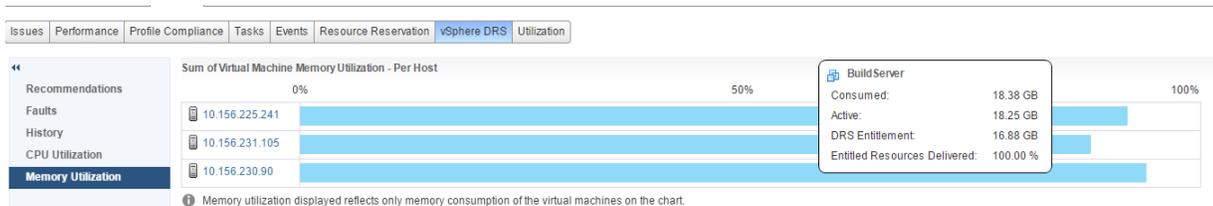


Figure 15 - DRS Memory Utilization summary

Case Studies

Case Study 1 – DRS Provides Effective VM Initial Placement

When a VM in a cluster is going to be powered on, it is the job of DRS to pick the right host to ensure that the VM gets all the resources it needs to run soon after being powered on. This study shows how DRS does effective initial placement and how this can affect the performance of the virtual machines.

⁴ Consumed memory is the amount of physical memory that has been allocated to a VM, but not necessarily being currently used.

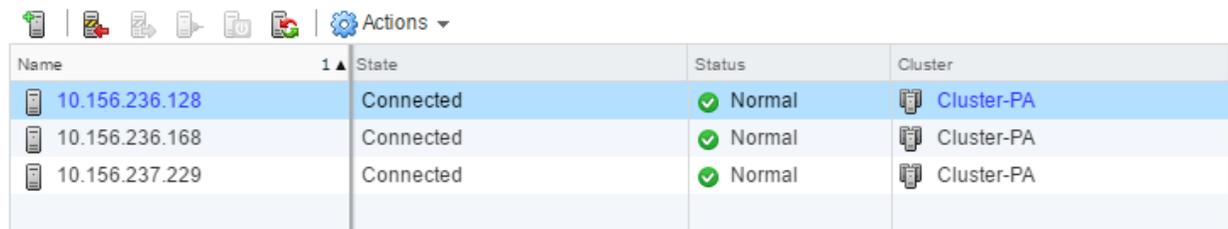
⁵ VM active memory is the amount of memory that is currently being actively used by the VM. This is an estimate provided by the ESXi host.

During initial placement, DRS first determines the amount of resources the VM needs soon after it powers on. For this, it considers the configured size (CPU + memory) of the VM, assuming that it will need all of its configured resources. DRS then powers on the VM in the best host in the cluster that can satisfy the VM's resource requirements. In this process, DRS might decide to unregister the VM from its current host and re-register it on a different host in the cluster, before powering it on.

In this case study, we have a cluster with 3 hosts, as shown in [Figure 16](#).

Expectation

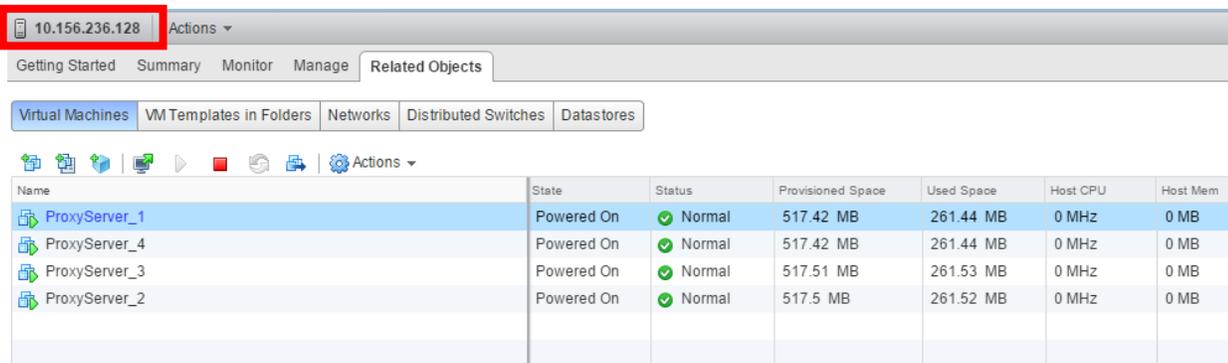
When powering on VMs in a cluster, the guest OS and its applications should have all the compute resources they need to boot up and start running in a timely manner. In this experiment, we power on VMs that are configured to run CPU-heavy applications. So, when the VMs are booted up, we do not expect to see any CPU ready time⁶ in any of the VMs.



Name	State	Status	Cluster
10.156.236.128	Connected	✓ Normal	Cluster-PA
10.156.236.168	Connected	✓ Normal	Cluster-PA
10.156.237.229	Connected	✓ Normal	Cluster-PA

Figure 16 - Cluster with three hosts

Initially, one of the hosts has 4 VMs that are CPU heavy, and the user starts these VMs without DRS. All of them are powered up on the same host ([Figure 17](#)).



Name	State	Status	Provisioned Space	Used Space	Host CPU	Host Mem
ProxyServer_1	Powered On	✓ Normal	517.42 MB	261.44 MB	0 MHz	0 MB
ProxyServer_4	Powered On	✓ Normal	517.42 MB	261.44 MB	0 MHz	0 MB
ProxyServer_3	Powered On	✓ Normal	517.51 MB	261.53 MB	0 MHz	0 MB
ProxyServer_2	Powered On	✓ Normal	517.5 MB	261.52 MB	0 MHz	0 MB

Figure 17 - No DRS: Host view showing all VMs powered up on the same host

Since all the VMs are powered up on the same host, and there is some resource contention, we see some CPU ready time in the VMs, as shown in [Figure 19](#).

We put the cluster back in the initial state and enable DRS in the cluster. With DRS enabled, when the VMs are powered on again, they are placed on different hosts. The History tab shows this initial placement of VMs ([Figure 18](#)).

⁶ CPU ready time is the amount of time a VM was in a state where it was ready to execute, but there weren't any CPU resources available. CPU ready time is usually a very good indicator of CPU contention in a host.



Figure 18 - DRS placements in the History view

DRS places two of the VMs on the same host, and the other two on different hosts. This time, we see no ready time.

Observation

DRS proactively calculates the resource demand of a VM while it powers on and places it on a host such that it does not run into any resource contention soon after it powers on.

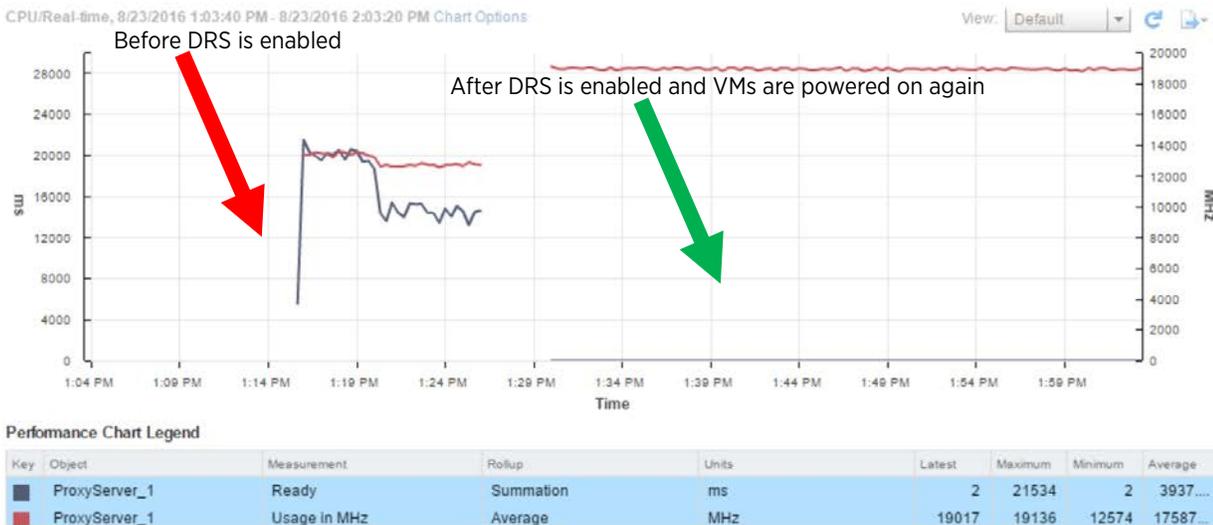


Figure 19 - CPU Ready Time before and after DRS is enabled

Case Study 2 – Memory Considerations for DRS Load Balancing (VM Consumed vs. Active Memory)

In the [Calculating VM Resource Demand](#) section, we learned that DRS balances memory workloads based on VM active memory usage. This case study explores how the consideration of active memory vs. consumed memory usage can influence DRS decisions.

In this case study, the cluster has 3 hosts, all with similar VMs (in terms of their memory configuration), as shown in [Figure 20](#).

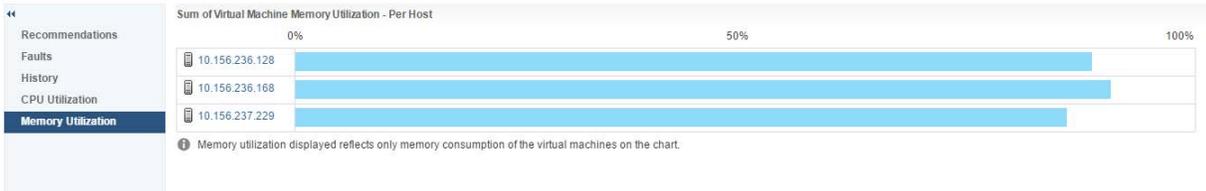


Figure 20 - Cluster Memory Utilization (consumed memory)

Only one of the hosts has VMs with low active memory, as shown in [Table 1](#). Since DRS considers active memory during load balancing, when the active memory is low on this host, but is higher and remains constant on the other hosts such that it causes an imbalance in active memory load distribution, it marks the cluster as imbalanced as shown in [Figure 21](#).

Host	Active Memory (as % of configured)	Consumed Memory (as % of configured)
10.156.236.128	16	85
10.156.236.168	85	90
10.156.237.229	78	80

Table 1 - Table summarizing the active and consumed memory distribution across hosts in the cluster

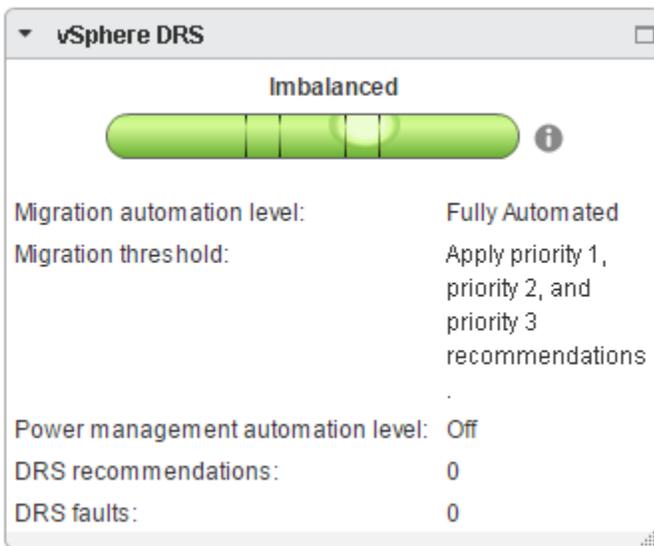


Figure 21 - Cluster Status shows cluster state is imbalanced

This results in the migration of one of the memory-heavy VMs onto the host with low active memory. Note that almost all of the physical memory has already been mapped to VMs on all three hosts (total consumed memory is close to or equal to total capacity on all hosts). To accommodate the additional VM being migrated with vMotion, the host with low active memory now has to reclaim memory from an existing running VM. This causes memory contention on one of the running VMs on the host, and eventually results in swapping in this case, as shown in [Figure 22](#).

Expectation

When we see that the (consumed) memory is more-or-less equally balanced across all hosts in the cluster, we do not expect DRS to migrate any VMs. Even if migrations do happen, we don't expect to see any ballooning or swapping in any VMs. Swapping could potentially lead to performance problems in the VM.

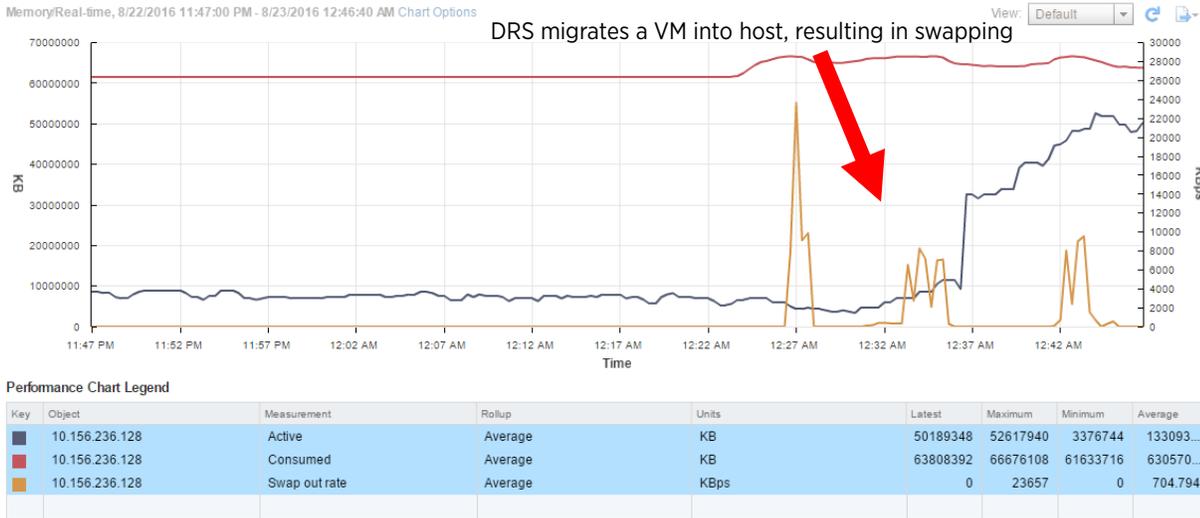


Figure 22 - Host memory graph shows some memory swapping because of VM migration

We learned in [Calculating VM Resource Demand](#) that during load balancing, by default DRS considers active memory with a small portion (25%) of the idle consumed memory (**idle consumed memory = consumed memory - active memory**) as the demand. When variations in active memory usage on the hosts lead to cluster imbalance, by default DRS will try to resolve this by migrating VMs with vMotion, irrespective of the consumed memory load distribution. In clusters where there is very high consumed, but very low active memory on the hosts, this behavior could cause some amount of memory pressure, as in this example.

This scenario can be handled using the advanced option **PercentIdleMBInMemDemand**. This option can be used to specify the percentage of idle consumed memory that DRS should consider when balancing memory load.

After setting **PercentIdleMBInMemDemand** to **100** (consider entire consumed memory), we see that the cluster is in a balanced state as shown in [Figure 23](#), and there is no memory swapping activity observed on the host, as shown in [Figure 24](#).

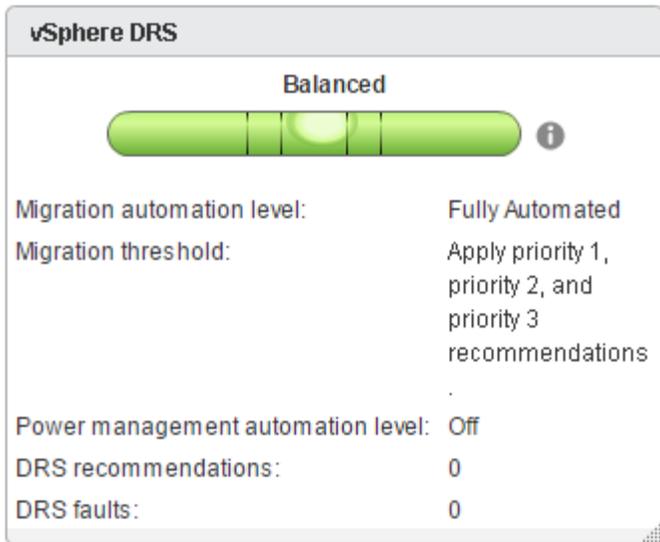


Figure 23 - Cluster Status is balanced

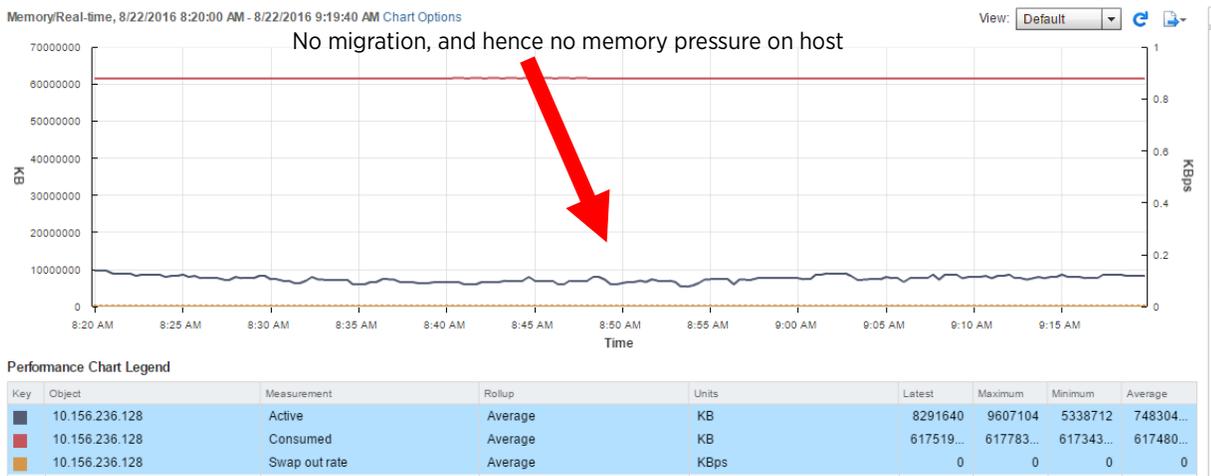


Figure 24 - Host memory graph with PercentIdleMBInMemDemand set to 100

Observation

With the advanced option **PercentIdleMBInMemDemand**, DRS considers all of a VM's idle consumed memory, along with its active memory (in other words, considering its total consumed memory) for load balancing. As a result, DRS finds the memory load to be balanced across the cluster and finds no need to migrate any VMs. This option can help in cases where over-committing memory resources might not be desirable. In general, if there is no memory over-commitment in the cluster, and the VM workloads are sensitive to availability of memory, we recommend you set the advanced option **PercentIdleMBInMemDemand** to **100**.

Case Study 3 - Cluster Balance Impacted by VM Overrides

As mentioned earlier, rules are an important customization feature in DRS. Rules empower you to tune DRS behavior to work with critical VMs and workloads that require special settings. While rules are very useful, they tend to limit DRS to work with constraints, which can sometimes hinder its progress.

The following example is based on an issue observed by a customer in the field. We have a DRS cluster with 3 hosts. Some of the VMs in the cluster are CPU intensive and others are memory intensive. All the CPU intensive VMs have a VM override setting with DRS manual automation level, and are spread across two different hosts as shown in [Figure 25](#).

VM Overrides

Add ... Edit ... Delete

Name	vSphere DRS Automation Level	Host
ProxyServer_1	Manual	10.156.230.90
ProxyServer_2	Manual	10.156.225.241
ProxyServer_3	Manual	10.156.225.241
ProxyServer_4	Manual	10.156.230.90

Figure 25 - VM Overrides settings

There is also an affinity rule that is added later that requires all the CPU intensive VMs to be together on the same host, as shown in [Figure 26](#).

VMHost Rules

Add ... Edit ... Delete

Name	Type	Enabled	Conflicts	Defined By
proxyServers	Keep Virtual Machines Together	Yes	0	User

VMHost Rule Details

The listed 4 Virtual Machines must run on the same host.

Add ... Details ... Remove

Rule Members	Conflicts
ProxyServer_1	0
ProxyServer_4	0
ProxyServer_3	0
ProxyServer_2	0

Conflicts

Figure 26 - VM affinity rule setting to keep VMs together on the same host

Initially the cluster is in an imbalanced state (Figure 28). We see DRS has made two manual recommendations to move the CPU intensive VMs (Figure 27), which must be applied manually due to the VM override settings.

DRS was last run on: 8/10/2016 1:21 AM

DRS Recommendations

Apply	Priority	Recommendation	Reason
<input checked="" type="checkbox"/>	1	Migrate ProxyServer_3 from 10.156.225.241 to 10.156.230.90	Apply affinity rule
<input type="checkbox"/>	1	Migrate ProxyServer_2 from 10.156.225.241 to 10.156.230.90	Apply affinity rule

Figure 27 - DRS Recommendations to apply affinity rules

vSphere DRS

Imbalanced

Migration automation level: Fully Automated

Migration threshold: Apply priority 1, priority 2, and priority 3 recommendations

Power management automation level: Off

DRS recommendations: 2

DRS faults: 0

Figure 28 - DRS status is imbalanced

However, there is no visible CPU pressure on the VMs (Figure 29), so you might ignore these recommendations.

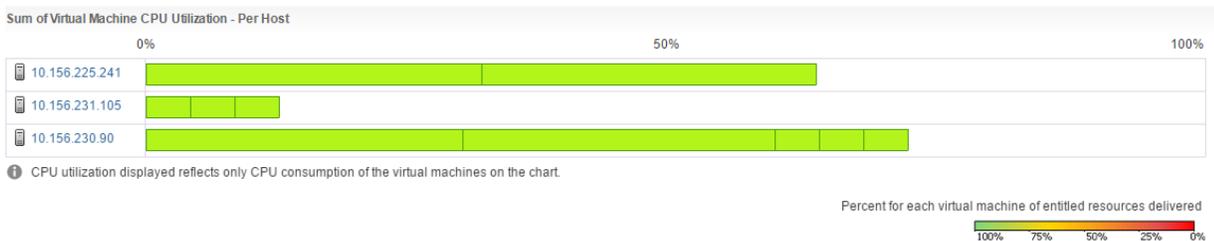


Figure 29 - Cluster CPU Utilization

When we look at the memory utilization across the hosts (Figure 30), there seems to be an imbalance. Even after waiting for a few rounds of DRS load balancing runs, DRS is not able to clear this memory imbalance.

Expectation

DRS is expected to work, even in the presence of custom VM Rules and Override settings, and is expected to clear out the cluster imbalance.

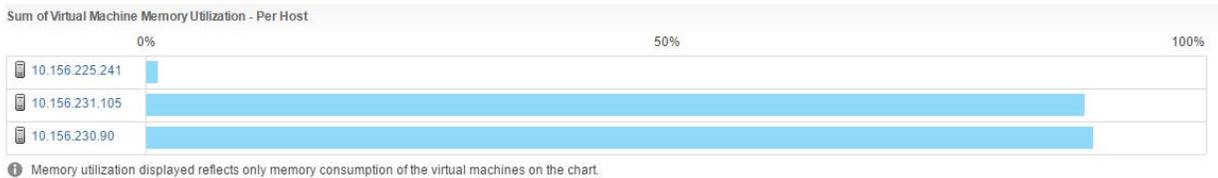


Figure 30 - Cluster Memory Utilization shows one of the hosts has less memory consumed

Upon closer inspection of the DRS recommendations in the Recommendations section, it appears they were generated to apply the newly introduced affinity rules. This information is specified in each of the generated recommendations (under the column Reason). DRS is unable to execute them on its own, since the VMs in these recommendations have VM overrides set to a manual automation level. As long as these recommendations are ignored, DRS will not be able to make any recommendations for load balancing that involves the hosts in these recommendations as the source. In this case, this was the reason why DRS was unable to clear the cluster imbalance. Once we apply these manual recommendations, DRS runs three more recommendations, as shown in Figure 31, and clears out the memory imbalance.

Recommendations	Time	DRS Actions
Faults	Thursday, August 11, 2016 2:51:44 AM	Migrate Database_MSSQL from 10.156.230.90 to 10.156.225.241
History	Thursday, August 11, 2016 2:50:37 AM	Migrate BuildServer2 from 10.156.231.105 to 10.156.225.241
CPU Utilization	Thursday, August 11, 2016 2:36:45 AM	Migrate JenkinsServer from 10.156.230.90 to 10.156.231.105
Memory Utilization	Thursday, August 11, 2016 2:27:00 AM	Migrate ProxyServer_2 from 10.156.225.241 to 10.156.230.90
	Thursday, August 11, 2016 2:27:00 AM	Migrate ProxyServer_3 from 10.156.225.241 to 10.156.230.90

Figure 31 - DRS History shows moves to clear memory imbalance, after the manual recommendations are applied

NOTE: Ignore timestamps; they indicate task completion times.

Observation

When there are rules and VM overrides set in a DRS cluster, you should be extra careful to watch out for DRS recommendations and apply them whenever possible. This can help avoid situations where DRS is not able to do its job because of waiting for manual recommendations to be applied.

Case Study 4 – Guaranteeing Resources for VMs

Whenever a VM begins to experience resource contention due to cluster load imbalance, DRS detects it and remediates the situation in the next load balancing round by moving VMs around to rebalance the cluster.

However, there can be VMs in the cluster that might be running workloads that are highly critical and that require resources to be guaranteed to them at all times. This can be achieved using Reservation. Reservation guarantees a certain amount of resources (CPU or memory) at all times for a VM or a resource pool.

In this experiment, we have a DRS cluster that has memory resources over-committed. We have a critical VM running a memory workload in the cluster. When there is no reservation set, at some point we start to see some ballooning and swapping (idle consumed memory being swapped-out) in this VM, as shown in Figure 32. This is

not necessarily a big problem in the VM, since many applications can tolerate this. However, this is not desirable in a VM that is running a critical workload.

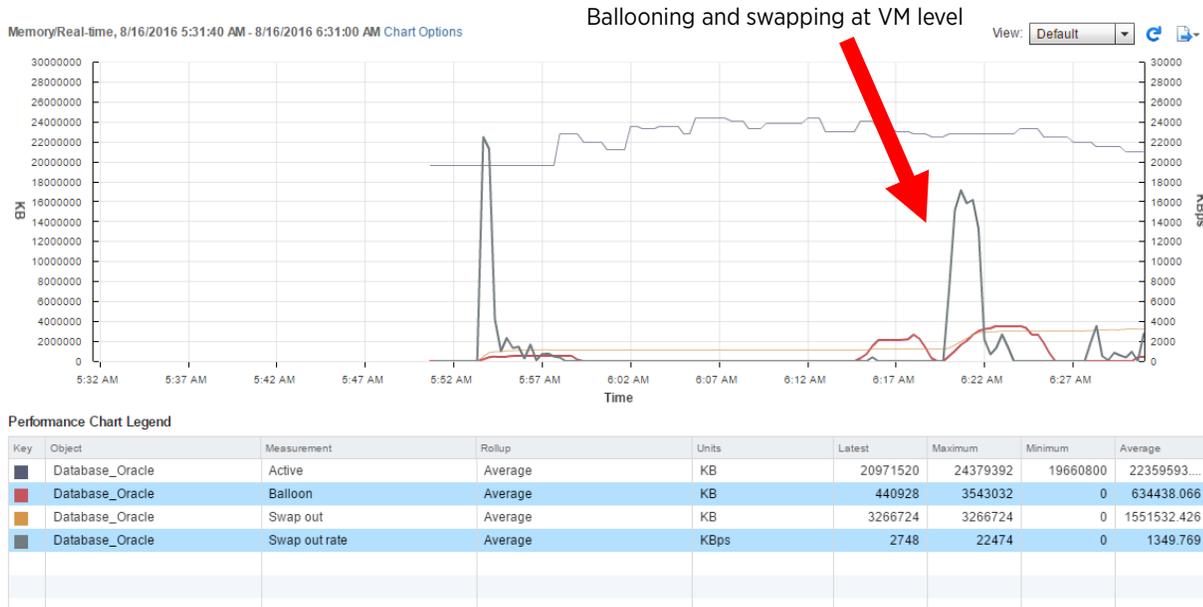


Figure 32 - VM Memory Graph showing ballooning and swap-out with no Reservation set

Expectation

We need to have a way to guarantee a certain amount of resources for critical VMs, such that those resources are always available to them.

Once the VM is given sufficient reservation, we no longer see any ballooning or swapping in the VM (Figure 33).

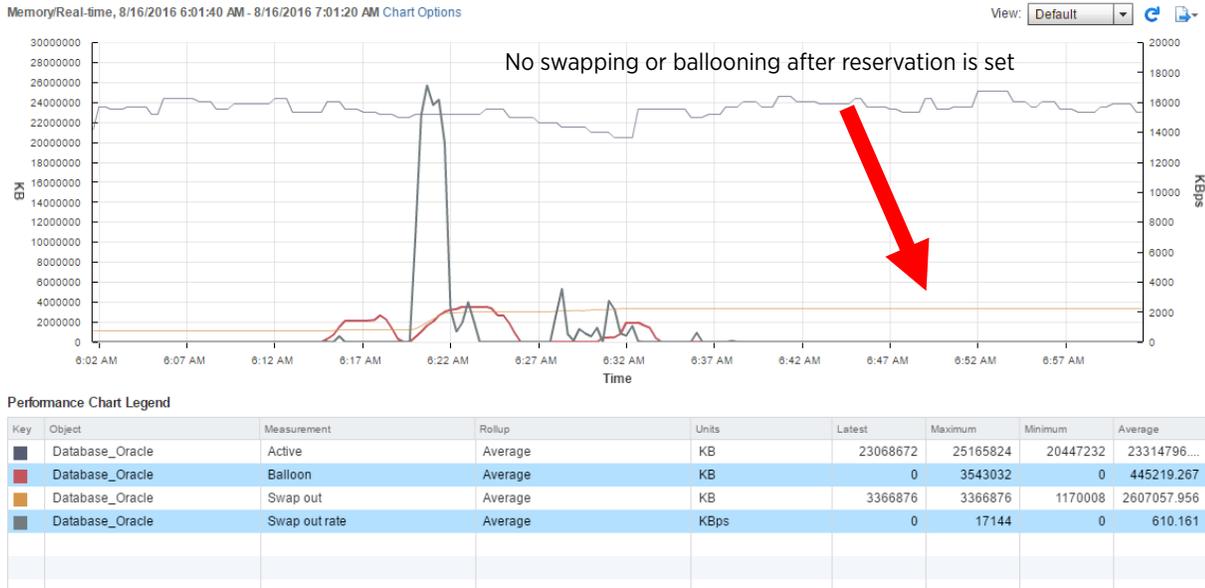


Figure 33 - VM Memory Graph showing no ballooning and swapping after reservation is set

Observation

Reservations can be used to ensure CPU and memory resources are allocated to critical VMs in a DRS cluster and are always available to them. VMs with reservations are guaranteed to not fall short of the reserved resources at any time. If there are several critical VMs, you could put them all in a resource pool and assign a sufficient amount of reservation for the resource pool, instead of assigning it to each VM.

Case Study 5 - Resource Prioritization Using Shares

Shares are one of the most commonly misunderstood resource management features. As explained in the section [Reservation, Limit, and Shares](#), shares are used to prioritize resources for VMs individually, or in a pool. Shares come into effect only when there is resource contention.

This use case explains how shares can be used to serve the resource demand of high priority VMs. There are two VMs in a DRS cluster with the same configuration of 25GB memory. Both the VMs are running a memory-heavy workload, but one of them is considered critical, and the other is not. They are both part of a resource pool ([Figure 34](#)), which has reserved memory of 42GB ([Figure 35](#)).

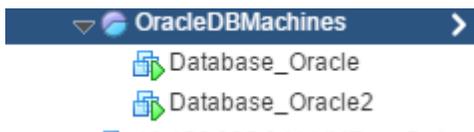


Figure 34 - Resource Pool

The screenshot displays two panels from the vSphere interface. The top panel, titled 'Resource Settings', shows the following configuration:

Resource Settings	
CPU	Normal (4000)
Memory	
Shares	Normal (163840)
Reservation	Expandable (42000 MB)
Limit	Unlimited
Worst Case Allocation	38.39 GB
Edit Settings...	

The bottom panel, titled 'Resource Consumers', shows the following counts:

Resource Consumers	
Virtual Machines and Templates	2
Powered On Virtual Machines	2
Child Resource Pools	0

Figure 35 - Resource Pool settings

This means that the resource pool is always guaranteed 42GB of memory for its VMs. When there is memory contention in the cluster, and the resource pool cannot get more than 42GB, the two VMs within this resource pool have to compete for this limited amount of memory.

Expectation

The VM with higher memory shares is expected to get more memory compared to the other VM when there is memory contention in the resource pool.

In the normal case (with normal shares for both VMs in the resource pool), with resource contention, we start to see an increase in ballooning and swapping in the VMs, affecting the performance of the VM running the critical workload, as shown in [Figure 36](#).

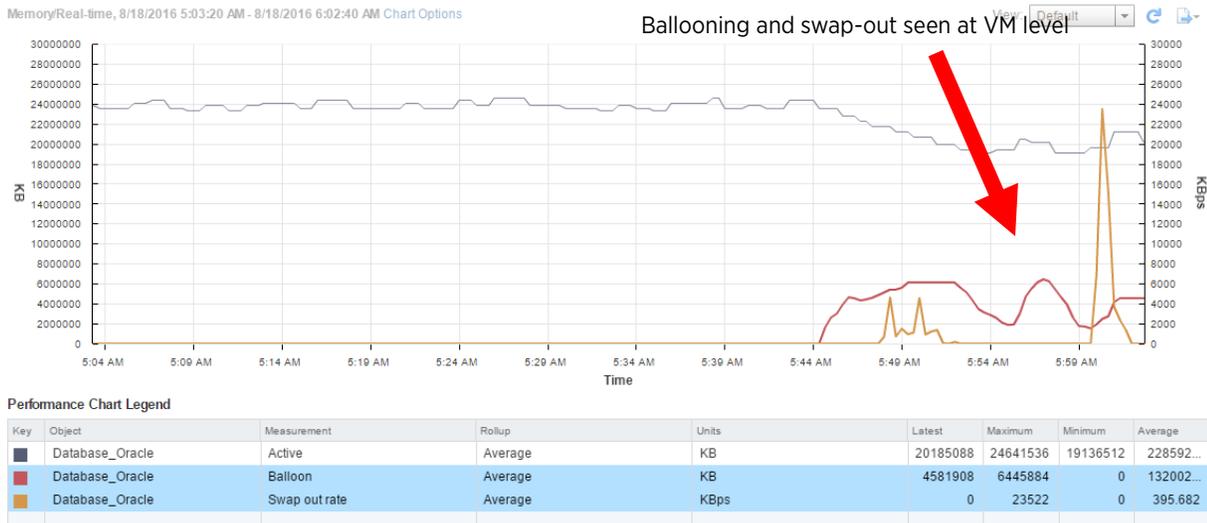


Figure 36 - VM Memory graph shows ballooning and swap-out in the high priority VM

This situation can be avoided by having proper shares set at the VM level such that the higher priority VM takes priority over the other. The memory shares of the high priority VM is modified to make it **High**, while shares for the other VM remain unchanged, as shown in [Figure 37](#) and [Figure 38](#).

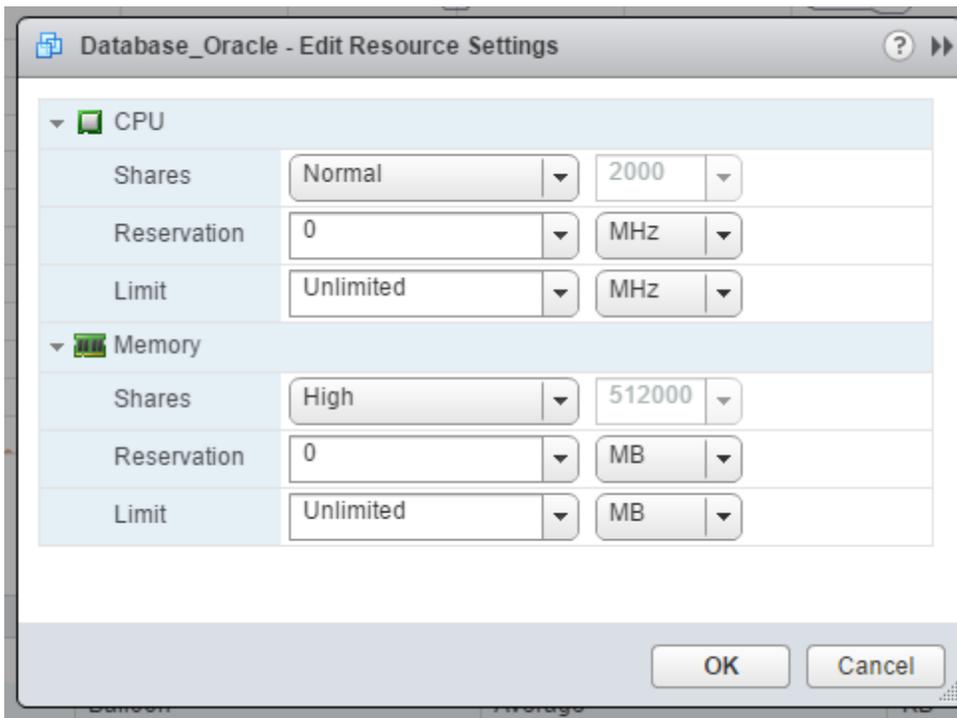


Figure 37 - VM with high memory shares

VM2 Configuration normal shares:

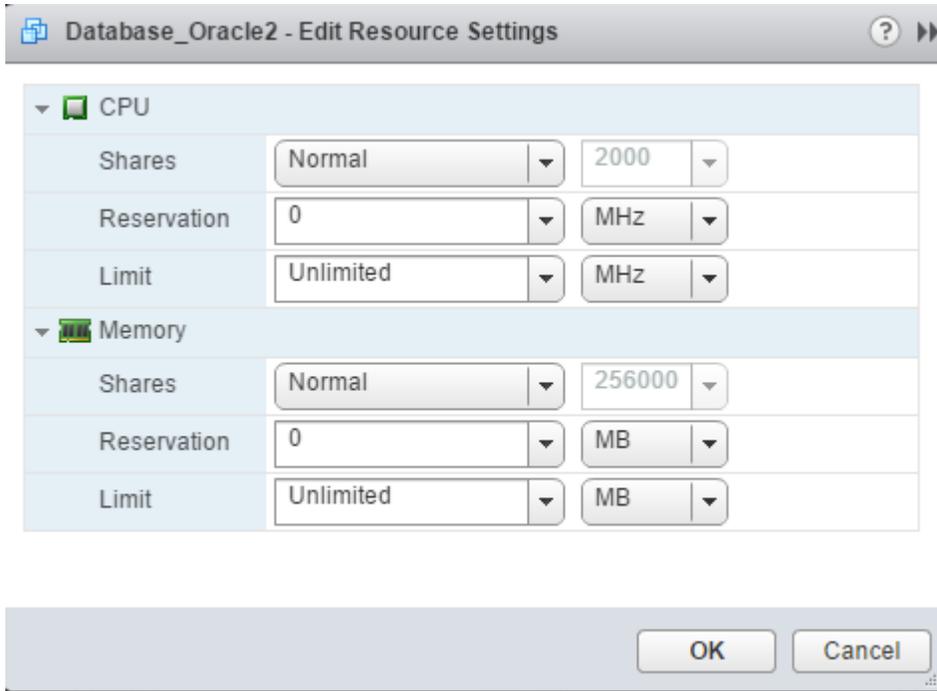


Figure 38 - VM with normal memory shares

After the VM shares are changed, there is no ballooning or swapping in the VM with higher shares, as shown in Figure 39.

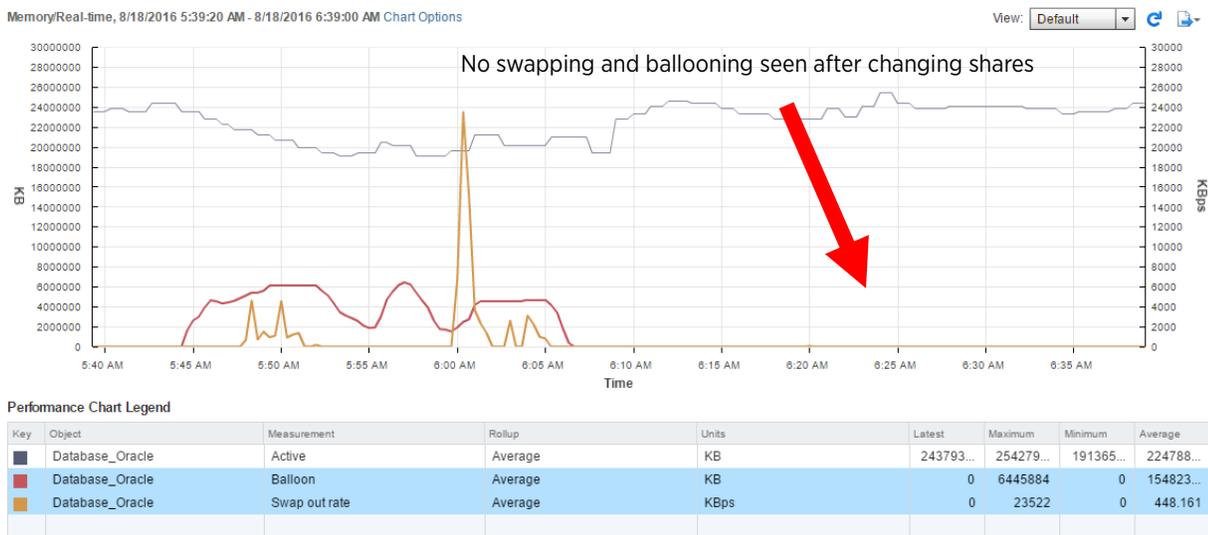


Figure 39 - Chart showing VM memory pressure reduced for high priority VM

For the VM with lower (normal) shares, the memory usage is further restricted and we see ballooning and high swap out rate, as shown in [Figure 40](#).

Again, this restriction is only when the resources are scarce. If there are enough resources, both VMs will get the resources they need.



Figure 40 – Memory performance chart for VM with normal shares

Observation

In clusters where resource contention is expected, or when working with over-committed resources, resources can be prioritized for critical VMs using shares. Shares come into play only when there is contention, so they can be used when you do not want to, or cannot allocate resources for some VMs all the time (like in the case of reservations).

Case Study 6 – How DRS Migration Threshold Impacts Cluster Balance

We learned in the section [DRS Aggression Levels \(a.k.a. Migration Threshold\)](#), that the migration threshold can be used to control the amount of imbalance that DRS can tolerate in a cluster. This directly impacts the number of vMotions that DRS does to balance out resources in the cluster. Very often, customers have observed that DRS is not issuing any vMotions, even though the load in their cluster seems to be unevenly distributed. In most cases, this is not a problem, since an uneven distribution of load may not necessarily indicate performance issues. However, if it is important to have the resource utilization balanced as much as possible, a simple change in the migration threshold will make DRS try to achieve better cluster balance.

In the following example, there is a cluster with three hosts, and the DRS migration threshold is at the default level (3). There are 60 VMs in the cluster, all running some CPU workload.

After DRS load balancing, one of the hosts has much higher CPU utilization compared to the other two, as shown in [Figure 41](#). DRS considers the standard deviation of the load to be below its target at the default level of migration threshold as shown in [Figure 42](#), so it does not initiate any more vMotions, and the cluster is balanced.

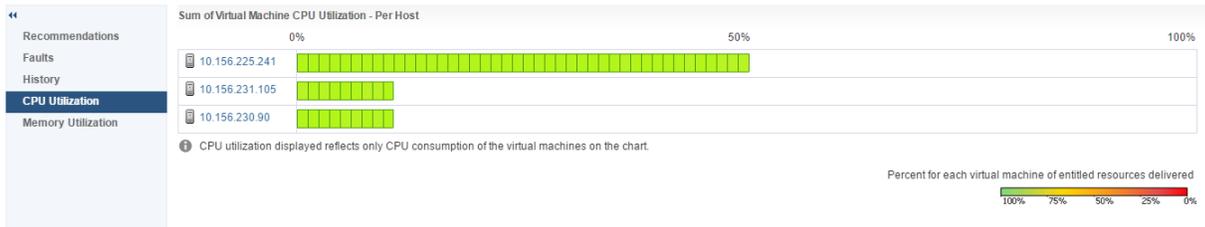


Figure 41 - Host CPU Utilization with default DRS migration threshold

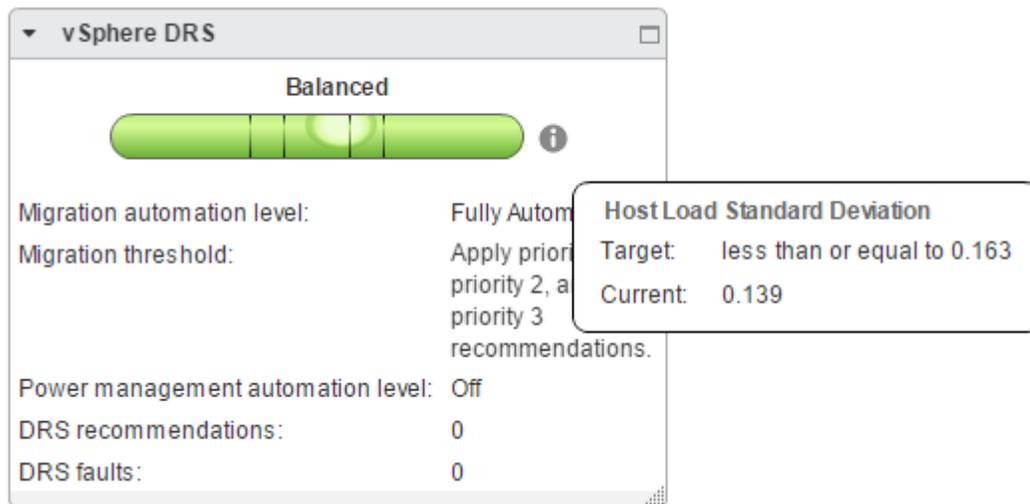


Figure 42 - Cluster balance Host Load Standard Deviation with default DRS migration threshold

When the migration threshold is changed to level 5 (most aggressive), DRS lowers the balance target, as shown in [Figure 43](#), and the current standard deviation of load is no longer under the target value. The cluster becomes imbalanced, and as a result, DRS initiates migrations with vMotion to further balance the load. Finally, the CPU utilization is moved and evenly distributed across the cluster, as shown in [Figure 44](#), and the cluster is balanced, as shown in [Figure 45](#).

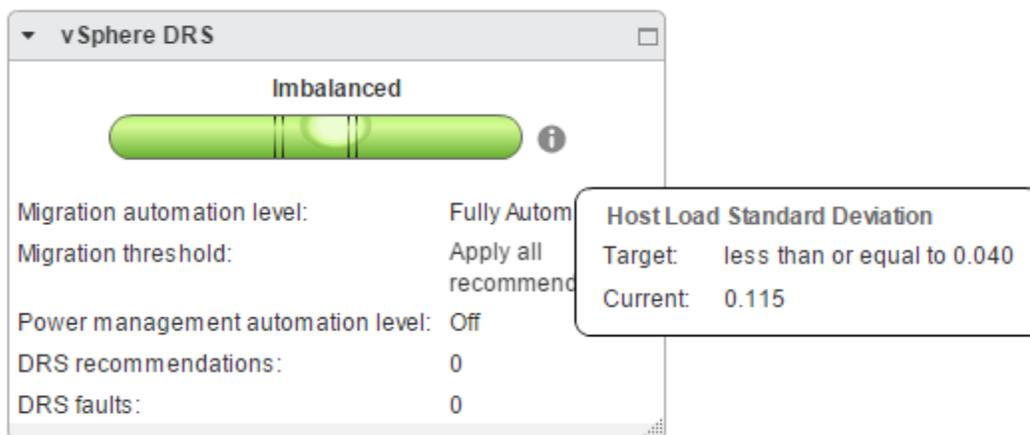


Figure 43 - Cluster balance Host Load Standard Deviation when DRS Threshold is set to 5

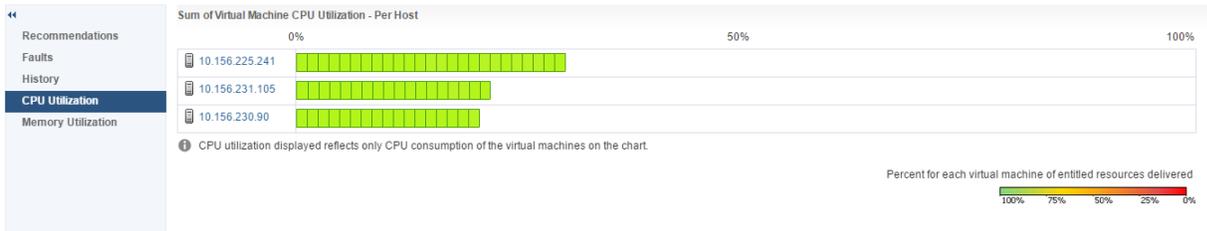


Figure 44 - Host CPU Utilization with DRS migration threshold set to 5

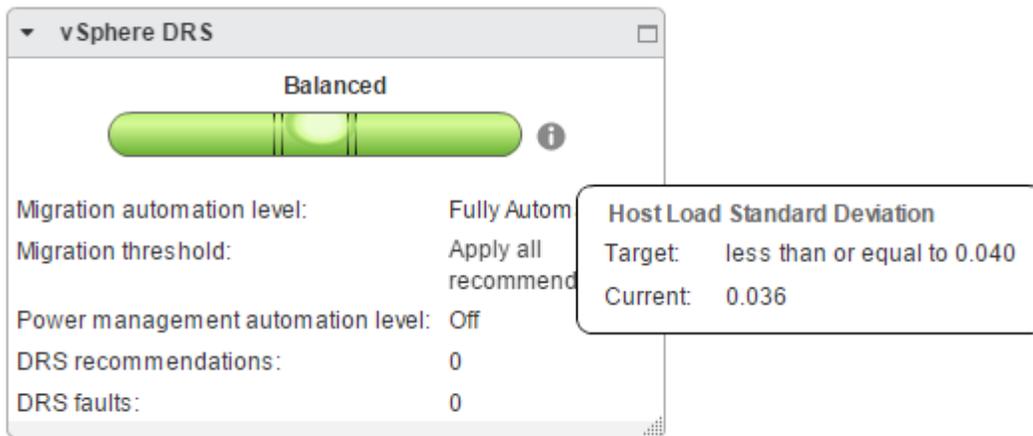


Figure 45 - Cluster Balance Host Load Standard Deviation after DRS balances cluster

Observation

DRS strives to ensure that VMs and their applications are getting the resources that they demand, while keeping the cluster resource utilization balanced as much as possible. In doing so, DRS considers many factors, like the cost of migrations, and the benefit of each vMotion in terms of the resulting cluster balance state. By default, DRS is not very aggressive due to all these considerations, and as a result, the cluster might not always be perfectly balanced.

When configuring your cluster, if one of your goals for DRS is to ensure that the cluster is balanced as much as possible, with a more even distribution of resources, at a potential cost of increased migrations with vMotion, you should increase the migration threshold of DRS.

Case Study 7 - DRS Considers Prerequisite Moves for Placement

DRS is very efficient in the way it does initial placement and load balancing. It includes several intelligent features that ultimately help make the best use of cluster resources with minimum manual intervention. One such feature is **prerequisite moves**.

This case study explains how DRS can make prerequisite moves before placing a VM if there are any affinity/anti-affinity rule violations. There is a cluster with three hosts. One of the hosts in the cluster has a VM

that is in the powered off state (WebProxyServer-1). This VM is part of three anti affinity rules with three different VMs, as shown in [Figure 46](#).

VMHost Rules

Name	Type	Enabled	Conflicts	Defined By
SeperateVMS	Separate Virtual Machines	Yes	0	User
SeperateVms2	Separate Virtual Machines	Yes	0	User
SeperateVms3	Separate Virtual Machines	Yes	0	User

VMHost Rule Details

The listed 2 Virtual Machines must run on different hosts.

Rule Members	Conflicts
WebProxyServer-1	0
DataBase1	0

Figure 46 - Anti-affinity rules

The three other VMs that are part of these affinity rules are disturbed across all three hosts (one on each host). Now when the VM (WebProxyServer-1) is powered on, DRS cannot place it on any of the hosts because of the anti-affinity rule violations.

Expectation

When a user issues a VM power-on in a DRS cluster, the expectation is that DRS will power up the VM on one of the hosts in the cluster, as long as the host has enough physical resources to support the newly powered-on VM. If there are any rules that might get affected due to starting the new VM, DRS should try to correct them on its own, or throw a fault indicating its inability to do so.

In our case, DRS first moves a VM with which WebProxyServer-1 has an anti-affinity rule out of one of the hosts and then places WebProxyServer-1 on that host, as shown in [Figure 47](#).

Time	DRS Actions
Tuesday, August 23, 2016 5:42:00 AM	Place WebProxyServer-1 on host 10.156.225.241
Tuesday, August 23, 2016 5:41:38 AM	Migrate 10.156.225.241:VMZero-ProxyServer-4 from 10.156.225.241 to 10.156.231.105

Figure 47 - DRS History indicating prerequisite moves before VM placement

Observation

DRS automatically takes care of prerequisite moves that are necessary for carrying out VM initial placement actions.

Conclusion

The key driver for DRS is VM/Application happiness, and it achieves this by effective VM placement and efficient load balancing. DRS also includes a host of features that are aimed at providing a robust, flexible, and smart solution for you in managing your cloud infrastructure.

Knowledge about the factors that affect DRS behavior can help you better understand how DRS does what it does. This knowledge, along with monitoring and troubleshooting tips, including real case studies, enables you to tune your DRS clusters for optimum performance.

In this paper, we have described case studies (using vCenter Server version 6.0) that were based on real situations that customers have observed in their DRS Clusters, to showcase the features of DRS.

References

- [1] VMware, Inc. (2016) Data Center Consolidation and Business Continuity.
<http://www.vmware.com/ca/products/vsphere/data-center-consolidation.html>
- [2] VMware, Inc. (2015) vSphere 6 Resource Management Guide.
<https://pubs.vmware.com/vsphere-60/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-60-resource-management-guide.pdf>
- [3] Aashish Parikh and Ajay Gulati. (2012) DRS: Advanced Concepts, Best Practices, and Future Directions.
<http://download3.vmware.com/vmworld/2012/top10/vsp2825.pdf>
- [4] Ajay Gulati, Ganesha Shanmuganathan, Ann Holler, and others. (2012) VMware Distributed Resource Management: Design, Implementation, and Lessons Learned.
<https://labs.vmware.com/vmtj/vmware-distributed-resource-management-design-implementation-and-lessons-learned>
- [5] VMware, Inc. (2016) Performance Best Practices for VMware vSphere 6.0.
<http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vmware-perfbest-practices-vsphere6-0-white-paper.pdf>

About the Authors

Vikas Madhusudana is a senior performance engineer with vCenter Server performance group. He works on vCenter server performance and scalability with special focus on development of tools for performance testing/automation. Vikas holds a bachelor's degree in computer science from University Visvesvaraya College of Engineering, Bangalore, India.

Adarsh Jagadeeshwaran works in the vCenter Server performance group in VMware, leading a team that is focused on vCenter Server resource management and scalability. Adarsh holds a master's degree in Computer Science from Syracuse University, Syracuse, USA.

Sai Manohar Inabattini is a senior performance engineer with the vCenter Server Performance group. He works on vCenter Server performance and scalability, with special focus on the Distributed Resource Scheduling (DRS) algorithm. Sai holds a bachelor's degree in computer science from the National Institute of Technology at Warangal, India.

Acknowledgments

The authors would like to thank Priyanka Gayam for her help with the case studies, Fei Guo, and Lan Gao for their help with discussing the case studies and paper review, and Ravi Soundararajan, Vikas Shitole, Sahan Gamage, and Pranshu Jain for their invaluable reviews and support of this paper.

