Load Balancing Performance of DRS in vSphere 7.0

Performance Study - July 15, 2020

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com

Table of Contents

Executive Summary 3

Introduction 3

Faster Response to Load Imbalance 3

The Old DRS 3

The New DRS 5

No Load Imbalance Metric 6

Example 1: Outlier Host Workload with the Old DRS 6

The New DRS 7

Example 2: Uneven Host Workload 7

The Old DRS 7

The New DRS 8

VM Selection 8

New Cost Modeling 8

Network Load Balancing 8

Example 9

DRS Score 10

Example 11

Accounting vMotion Benefit 12

Example 13

Considering Granted Memory Instead of Active Memory to Compute Demand 14

Example 14

Tuning DRS Sensitivity (Migration Threshold) 16

VMs in Partially Automated Mode 16

Conclusion 17

References 18

Executive Summary

New features of DRS in VMware vSphere 7.0 have been introduced to make VMware's compute control plane robust and better handle both traditional and modern application needs.

Introduction

VMware vSphere® Distributed Resource Scheduler™ (DRS) is the resource management feature of vSphere. vSphere 7.0 includes a new DRS with improvements for resource scheduling—primarily, the feature now works at the VM level (ensuring better resource availability to all VMs) instead of at the cluster level (balancing load across the hosts). This gives finer control of resources to each VM and makes resource scheduling faster and more efficient.

DRS provides two key aspects of resource management: resource scheduling (in the form of VM placement) and load balancing. The VM placement aspect of the new DRS was introduced in vSphere 6.5. The new VM placement:

- Is faster and lighter
- Ensures more even placement of VMs across the cluster upon power-on
- Powers on VMs much faster, even with highly concurrent workloads

More information about the new VM placement can be found in the "DRS Performance of vShere 6.5" white paper [1].

The second aspect of resource management is load balancing. In this document, we cover the load balancing aspect of the new DRS, which is available in vSphere 7.0. Note that we refer to the older load balancing version of DRS (vSphere 6.7 and older) as the old DRS.

Faster Response to Load Imbalance

Traditionally, DRS load balancing took place once every five minutes. This worked well for most cases, because most of the traditional workloads like databases and other tier-1 applications were long-lived and changed less frequently. In general, the state of the cluster was stable for long periods of time. In contrast, datacenters these days are highly dynamic in nature, so reacting quickly to VM demand changes due to workload variation is crucial. As a result, in the new DRS, load balancing is done every minute, instead of every five minutes. Also, starting in vSphere 7.0, creating containers natively on vSphere is supported. With the arrival of container workloads in vSphere, the lifespan of these workloads is much shorter, resulting in potentially higher VM churn rates in the cluster. The new algorithm, which reacts every minute, is a step toward supporting these modern workloads.

In the next two sections, we look at how the performance of the old DRS compares to the new DRS.

The Old DRS

We ran a custom CPU stress workload to test the load balancing in the old DRS and compared it to that in the new DRS in vSphere 7.0.

Consider a 2-host, DRS-enabled cluster with 9 VMs and CPU contention as shown by the CPU readiness chart for VMs in Figure 1. In this example, the contention starts at 04:30pm.

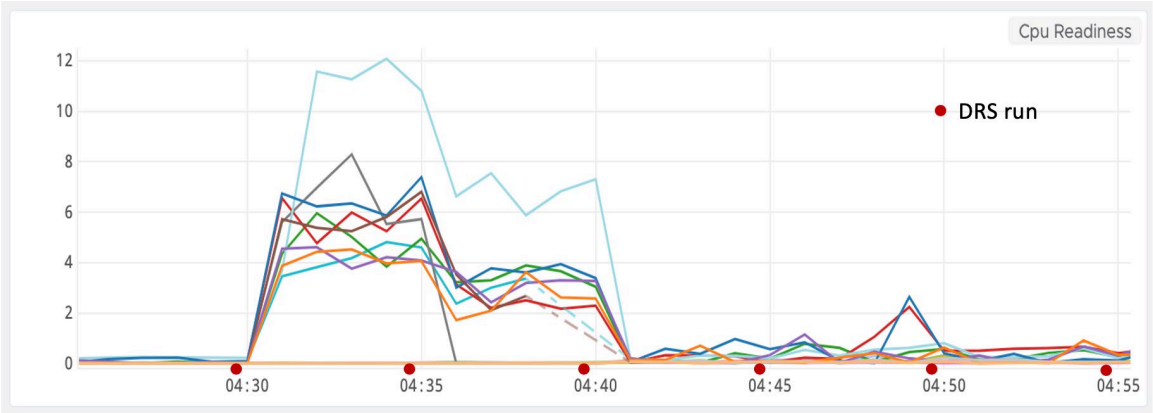

Figure 1. Cluster VM CPU Readiness (%)

This CPU contention resulted from a sudden increase in host load that started at 04:30pm and created an imbalance in the cluster, as shown in Figure 2.

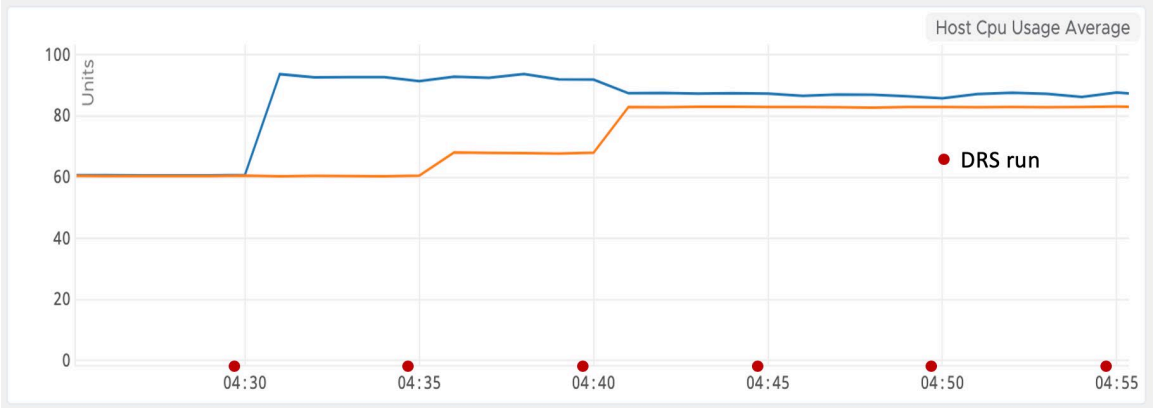

Figure 2. Host CPU Usage (%) Average

In this example, a DRS run invoked before 04:30pm. So, the next DRS run invoked just before 04:35pm. This means DRS reacted to the contention after only five minutes, and the VMs with CPU readiness continued to run that way for some more time.

In the next run, DRS recommended a few more vMotions, and the host CPU load became more evenly distributed around 04:42pm, as shown in Figure 2 (above). Around the same time, the CPU contention of VMs in the cluster was also cleared, as shown in Figure 1. In other words, CPU contention started around 04:30pm, and this contention was cleared around time 04:42pm (12 minutes later).

The New DRS

We then repeated the experiment with the new DRS. The initial state was very similar to the old DRS case. Some VMs in the cluster started to see CPU contention in the form of readiness at 03:54pm, as shown in Figure 3.

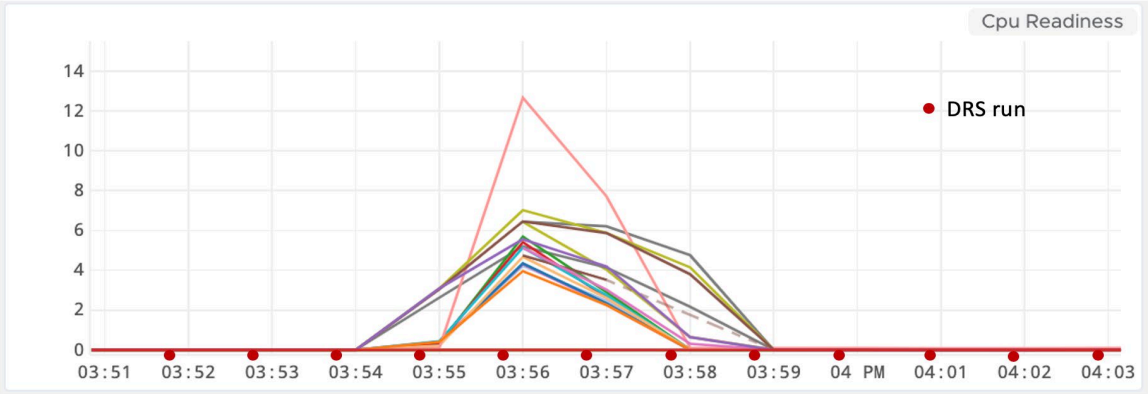

Figure 3. Cluster VM CPU Readiness

This CPU contention was correlated with the sudden increase in CPU load on one of the hosts in the cluster, as shown in Figure 4.

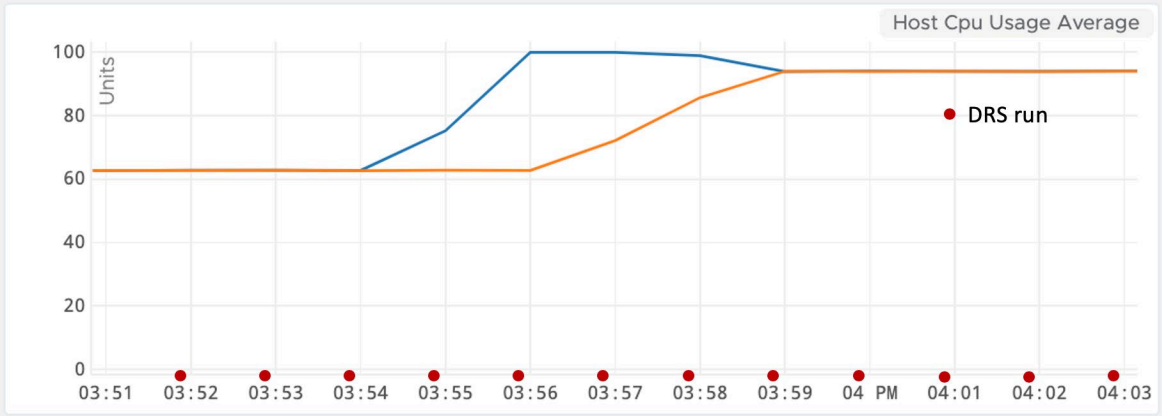

Figure 4. Host CPU Usage Average (%)

Now, DRS runs once every minute, so it starts issuing vMotions to balance the load in its next run at 03:55pm, as shown in Figure 5.

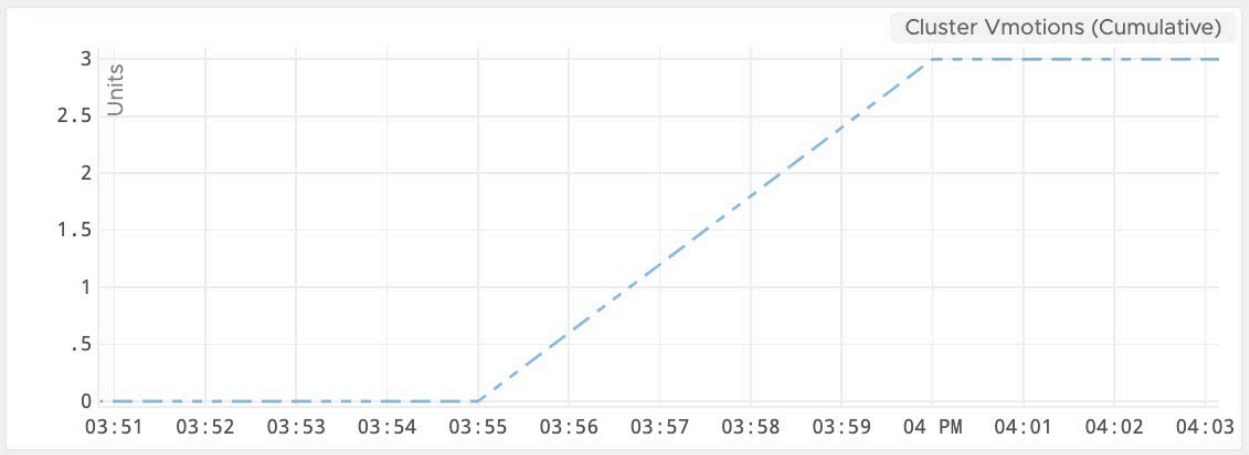

Figure 5. Cluster DRS vMotions (cumulative)

The CPU contention started at 03:54pm and the DRS vMotions helped balance the load and resolve CPU contention in the cluster by 03:59pm (in 5 minutes) as shown in Figure 3 and Figure 4 (above). Clearly the new DRS can resolve sudden resource contention in the cluster faster than the old DRS.

Now that the new DRS runs every minute, more vMotions can occur compared to the old DRS, because the old and the new DRS recommend different types of VMs for vMotion during load balancing. The new DRS generally recommends lighter VMs (in terms of their memory usage), whereas the old DRS recommends heavier VMs in general. More details about VM selection are discussed in later sections.

No Load Imbalance Metric

The old DRS uses an *imbalance* metric to keep workloads balanced in the cluster. This metric is derived from the standard deviation of load across hosts in the cluster. If the imbalance metric is within a known threshold value, the cluster is considered balanced. The old DRS always tries to maintain the cluster load balance using this metric.

In the new DRS, the focus is on *VM happiness*—each VM will get the resources it needs—rather than targeting the load balance in the cluster. The new DRS ensures that VMs in the cluster always get the resources they need.

In order to ensure sufficient resource availability for all VMs, the new DRS tries to vMotion VMs if they can benefit from more resources in the destination host, and if the cost of such a vMotion is not higher than the resulting benefit. This workload-centric approach of DRS resource management also ensures the load is balanced across the cluster.

Example 1: Outlier Host Workload with the Old DRS

Consider a 4-host cluster with around 60% CPU usage on 3 hosts and 75% CPU usage on 1 host (outlier host), as shown in Figure 6. The old DRS considers the standard deviation of load and finds that the cluster load is balanced. As a result, it takes no further action in the cluster.

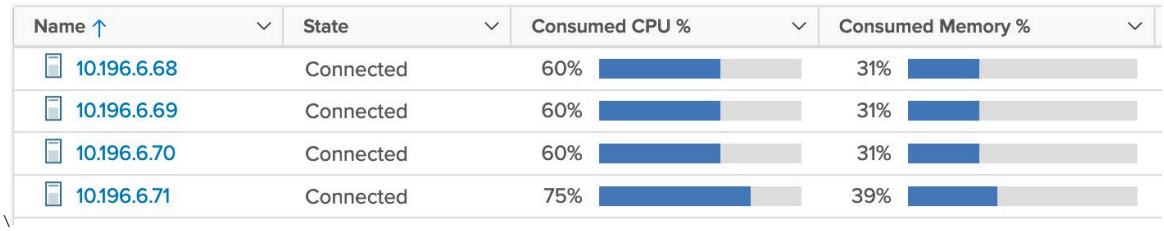

Figure 6. Cluster host distribution showing an outlier host (75% consumed CPU and 39% consumed memory)

The New DRS

Even though the cluster load is balanced per the imbalance metric, there are some VMs that could benefit in terms of better resource availability when moved to different hosts. In the new DRS, there is no imbalance metric and DRS continues to run its algorithm as long as it can find any vMotion opportunities that could improve the resource availability for VMs.

DRS checks to see if it can run a VM on a host with more available resources, compared to its current host. If so, it weighs the benefit of running the VM on the target host against the cost of moving it there (including, but not limited to the cost of vMotion). If the cost/benefit analysis passes, DRS issues the vMotion. As a result of these VM-centric decisions, the cluster load ends up being more evenly distributed, as shown in Figure 7.

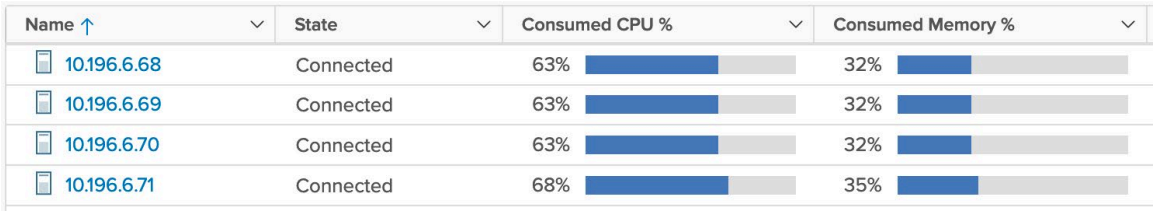

Figure 7. Cluster host distribution with the new DRS

Example 2: Uneven Host Workload

The Old DRS

Consider a 4-host cluster with 60% CPU usage on 2 hosts and around 40% usage on the other 2 hosts, as shown in Figure 8. DRS finds this cluster to be load balanced, since in this case, the imbalance metric is lower than the threshold. As a result, the cluster remains in this state.

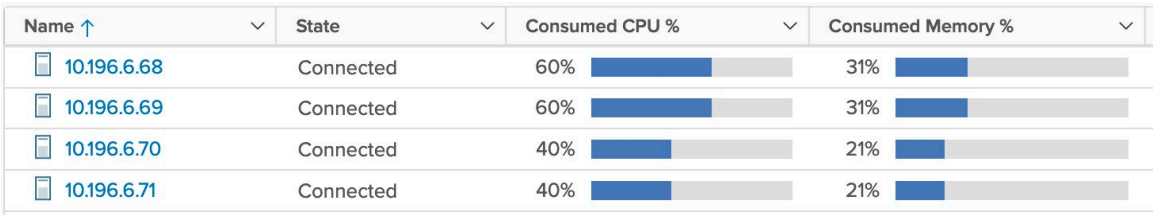

Figure 8. Uneven cluster host load distribution in a balanced cluster

The New DRS

For the same scenario, the new DRS finds that moving some VMs from the 60% loaded hosts to the 40% loaded ones can improve their resource availability. As a result, DRS recommends vMotions that lead to a more balanced cluster and guarantees better resource availability for the VMs. In this case, the resulting cluster host load distribution is shown in Figure 9.

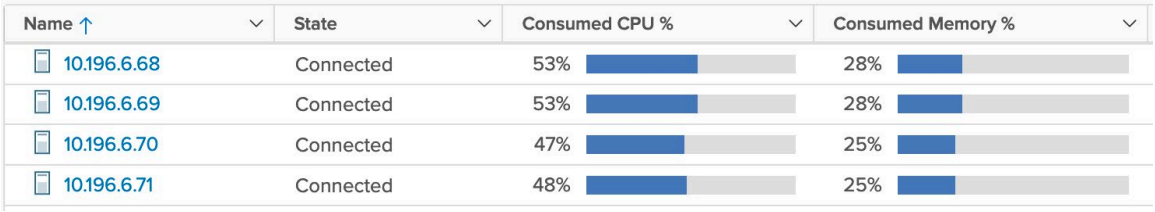

Figure 9. Cluster host load distribution with the new DRS

As we can see, the new DRS uses a VM-centric (aka workload-centric) approach during load balancing, as opposed to the cluster-centric approach (cluster imbalance metric) used by the old DRS. The new DRS tries to ensure better resource availability for all the VMs in the cluster and, as a result, distributes the cluster load more evenly.

VM Selection

The old DRS runs once every 5 minutes and computes the cluster imbalance metric to recommend vMotions. In this process, it tends to pick heavier VMs (in terms of both CPU and memory size) to clear as much imbalance as possible at once.

In contrast, the new DRS runs once every 1 minute and prefers to move lighter VMs (in terms of memory size). Picking lighter VMs to vMotion reduces the potential impact of vMotions on application performance of VMs across the cluster. Due to this change in behavior, DRS may need to recommend multiple vMotions to avoid resource contention or to improve resource availability for the VMs. As a result, you may sometimes notice increased vMotion activity, but the cluster will be better balanced over time. Also, DRS now moves only one VM at a time between any given pair of hosts to reduce the imbalance incrementally; this means that vMotions will be spread over time.

New Cost Modeling

DRS considers various factors to understand the cost of a vMotion (move) for load balancing. In the old cost model, DRS accounted for metrics like VM CPU, VM memory, vMotion time, etc. The new DRS uses a more flexible cost model. This makes DRS ready for future extensibility, in which case it will be able to remove or add any new VMware managed resource cost to address changing requirements.

Network Load Balancing

The cost model in the old DRS considers only CPU and memory metrics and moves VMs to balance the load for these metrics across the hosts in the cluster. As mentioned in the previous section, in the new DRS, the cost model is very flexible, and it is now easy to extend the framework to add any new metrics into the model.

With the new flexible cost model, the new DRS can balance the network bandwidth usage along with CPU and memory usage. Workloads that are network intensive will benefit from this functionality in the new DRS. Let's look at an example to understand this detail.

Example

Consider a 2-host cluster with balanced CPU and memory load, but with an imbalanced network load distribution. In the following charts (Figure 10, Figure 11, and Figure 12), we observe that between 08:10am and 08:15am on the timeline, the CPU and memory load is evenly distributed, but the total network (bandwidth) load is high on one of the hosts and low on the other.

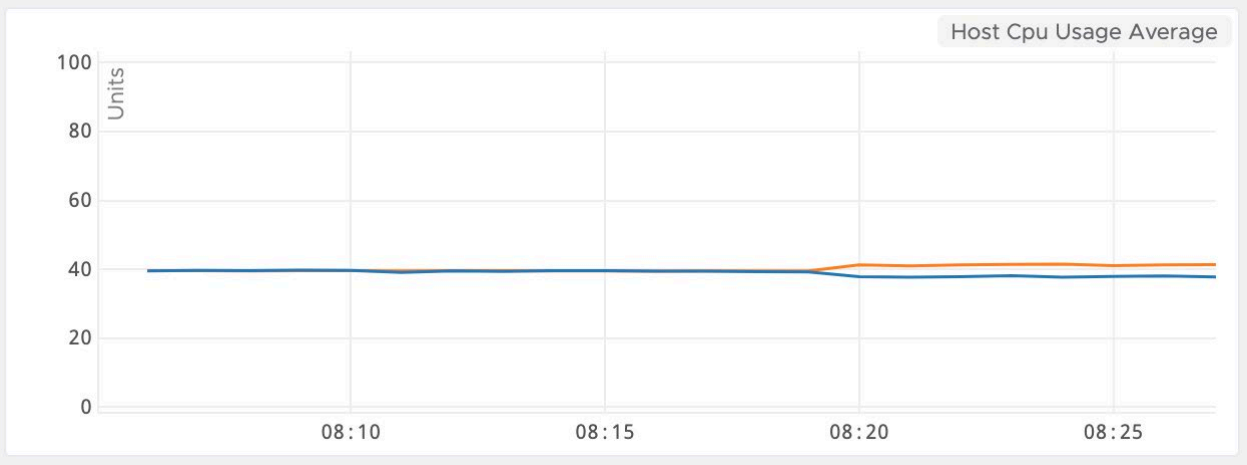

Figure 10. Balanced CPU load in the cluster

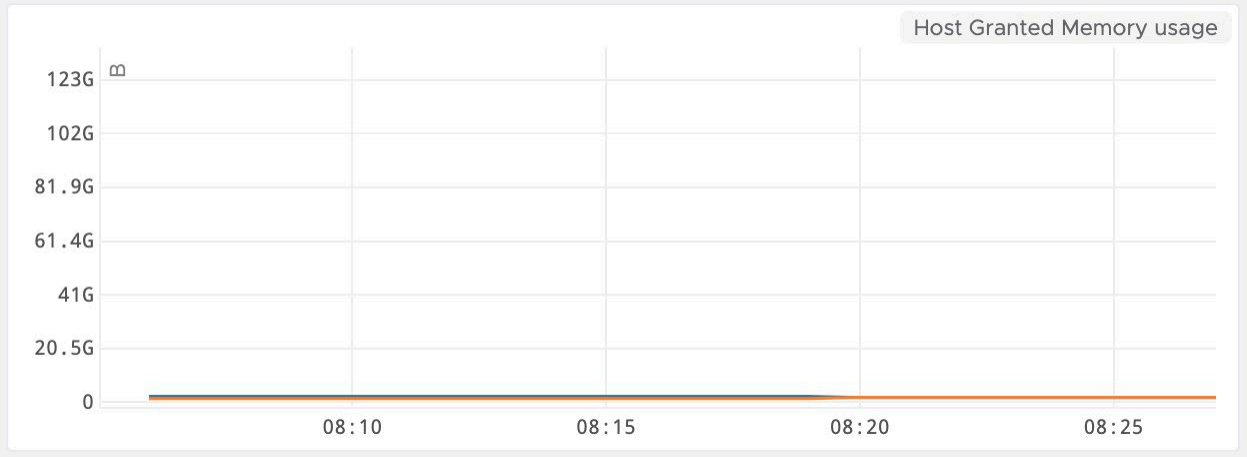

Figure 11. Balanced memory load in the cluster

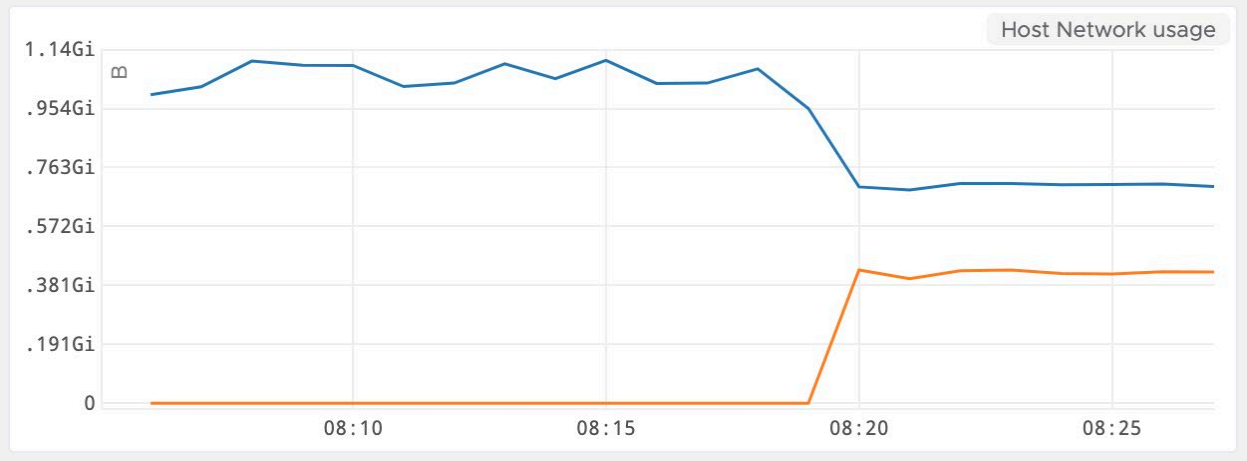

Figure 12. Total network load distribution in the cluster

In this example, the new DRS was enabled at 08:15am. As soon as it runs, DRS detects the network load imbalance. As shown in Figure 13, DRS begins to issue vMotions at 08:15am to reduce the network contention on the host with high network bandwidth utilization.

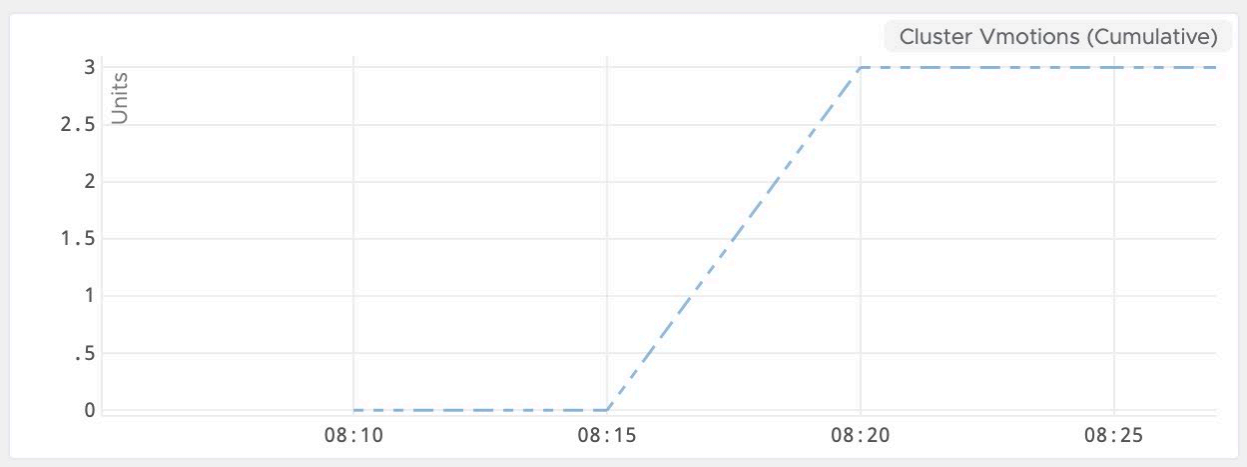

Figure 13. DRS vMotions to balance network bandwidth load distribution

As seen in Figure 12, the new DRS can detect and mitigate imbalance in the cluster related to network bandwidth utilization within minutes.

DRS Score

The new DRS is workload-centric and issues vMotions that improve the resource availability for VMs. One way to measure what DRS has achieved is by looking at a score that represents how *happy* a VM is—the VM DRS score, which is a measure of the resources available for consumption by the VM. The higher the DRS score for a VM, the better is the resource availability for that VM. DRS moves a VM from one host to another in order to improve this VM DRS score; that is, to improve the resource availability for that VM.

A cluster-level DRS score is also calculated to reflect the overall resource availability for VMs in the cluster. This score is a weighted sum of DRS scores of all the VMs in the cluster.

Example

Consider a 4-host cluster, where the overall resource utilization is around 30% as shown in Figure 14. In this scenario, all VMs are happy because there are enough resources in the cluster. Thus, all the VMs have high DRS scores and, as a result, the Cluster DRS Score is also high (Figure 15).

Name	State	Consumed CPU %	Consumed Memory %
10.196.6.68	Connected	29%	30%
10.196.6.69	Connected	29%	30%
10.196.6.70	Connected	29%	30%
10.196.6.71	Connected	29%	30%

Figure 14. Overall cluster level resource utilization

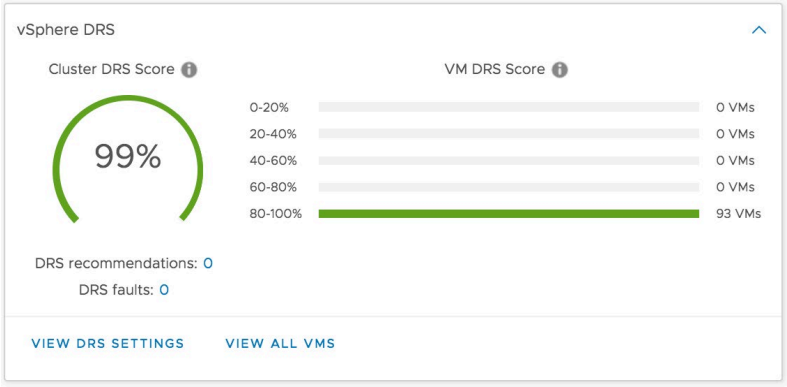

Figure 15. Cluster DRS Score showing a high (good) value

We then increase both the CPU and memory utilization on two of the four hosts, creating some resource contention on those hosts. Now, the VMs on those hosts do not have enough spare resources available to be used (Figure 16). As a result, the DRS score for these VMs drops, as does the overall cluster DRS score (Figure 17).

Name	State	Consumed CPU %	Consumed Memory %
10.196.6.68	Connected	99%	59%
10.196.6.69	Connected	73%	71%
10.196.6.70	Connected	29%	30%
10.196.6.71	Connected	29%	30%

Figure 16. Resource close to contention on two hosts

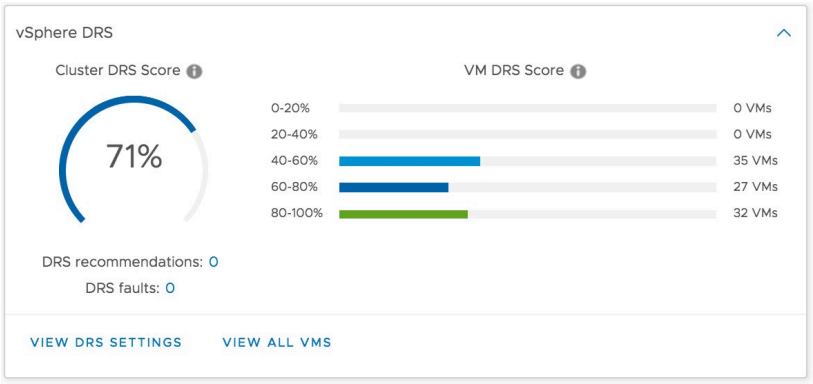

Figure 17. Reduced Cluster DRS Score

The next time that DRS runs, it moves VMs from a highly utilized host to other hosts where there are enough resources available (Figure 18). This increases the resource availability for the VMs moved, and thus the overall cluster DRS score improves (Figure 19).

Name	State	Consumed CPU %	Consumed Memory %
10.196.6.68	Connected	73%	41%
10.196.6.69	Connected	58%	59%
10.196.6.70	Connected	66%	43%
10.196.6.71	Connected	65%	47%

Figure 18. Resource distribution after DRS run

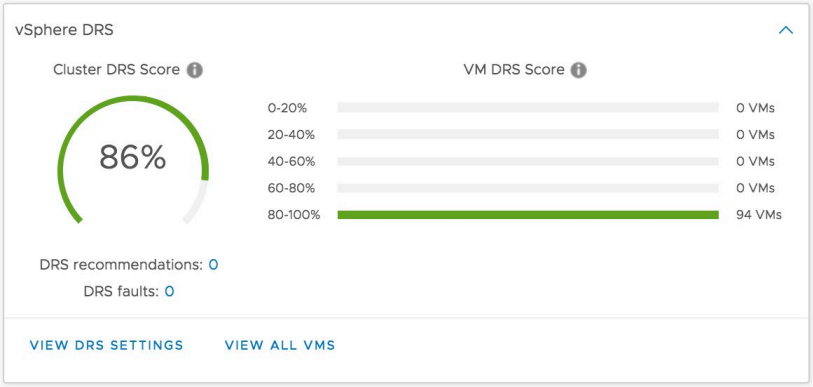

Figure 19. Improved cluster DRS score after DRS run

Accounting vMotion Benefit

vMotion can be an expensive operation, depending on the amount of VM memory pages to be copied onto the destination host. vMotion often requires considerable amounts of CPU, memory, and network resources on both the source and destination hosts. In a cluster where VMs are frequently powered on, or in a cluster that has VMs with volatile workloads, the cluster workload state changes often, and DRS might end up

continuously migrating VMs, with vMotion, in the cluster. In such cases, continued vMotions will not help in balancing the load in the long term. To avoid continuous vMotions, DRS must compute how long the gain of live migrating a VM is going to last (also known as *gain duration*) and decide how to vMotion VMs based on that.

When a VM is being powered on, DRS is invoked for a placement recommendation for the VM, to pick the right host for the newly powered-on VM based on the resource availability on hosts in the cluster. If a cluster has VMs that are frequently being powered on, then DRS should let the power-on placement distribute the cluster load, instead of issuing vMotions, each of which is a costly operation. Also, if there are continuous changes in VM workloads, DRS should avoid ping-pong vMotions in the cluster. The new DRS is aware of this and issues vMotions only after considering the gain duration for each vMotion.

Example

Consider a 2-host cluster with a balanced CPU workload (initially), and a negligible memory workload. As seen in Figure 20, the CPU load in both hosts is very similar until 07am. The VMs on one of the hosts have a highly unstable CPU workload. In this scenario, initially whenever there is a load change, DRS issues vMotions to balance the cluster load. As seen in Figure 20, the CPU load on one of the hosts goes up significantly for the first time around 07:10am. This is followed by the first set of DRS vMotions at the same time, as seen in Figure 21.

Until about 08am, DRS issues vMotions in response to workload changes in the cluster. During this time, DRS is also gradually tuning the gain duration to reduce continuous vMotions, because the workload continues to be unstable. After 08am, DRS does not issue any more vMotions in reaction to workload changes in this cluster to avoid ping-pong vMotions.

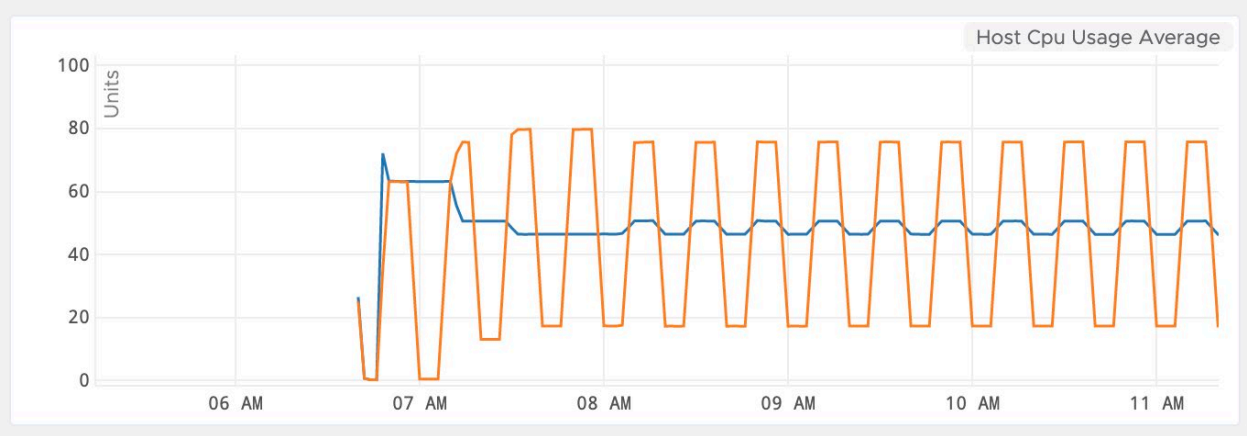

Figure 20. CPU usage (%) distribution in the cluster

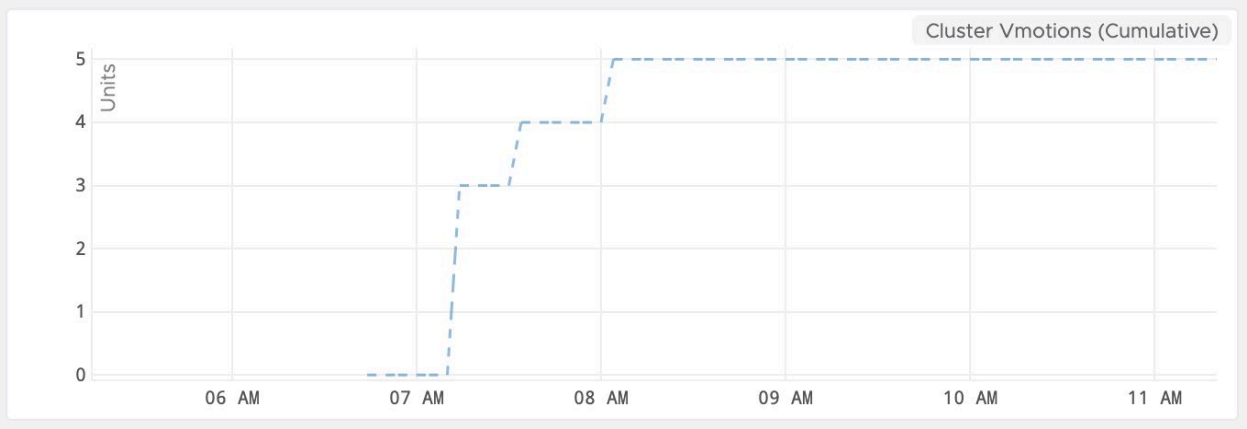

Figure 21. DRS vMotions (cumulative) in the cluster

Considering Granted Memory Instead of Active Memory to Compute Demand

By default, the old DRS considers *Active Memory*—memory actively being used by the VM —to compute host memory utilization. For example, if a VM consumes 4GB of memory at some point, say for booting an application, and once the application starts running, it uses only 1GB of memory pages, then DRS considers only 1GB memory usage for this VM while computing host memory utilization. Considering Active Memory to compute host memory utilization may result in DRS vMotions whenever the Active Memory usage of VMs change. Based on feedback from our customers, we understand that it's fairly uncommon to over-commit memory in a DRS cluster. When there is no memory over-commitment, there is no need to balance the cluster based on Active Memory. For this reason, a cluster-level DRS option *Memory Metric for Load Balancing* was introduced in vSphere 6.5.

To completely avoid these vMotions based on Active Memory fluctuations, the new DRS considers *Granted Memory*—host physical memory mapped to the VM—to compute host memory utilization. Also, there is no option to revert back to using Active Memory for host memory computation.

For more information on Active and Granted Memory, see the vSphere 7.0 product documentation: "Measuring and Differentiating Types of Memory Usage." [2]

Example

Consider a 2-host cluster with balanced CPU and a Granted Memory load, but imbalanced Active Memory (Figure 22). In this scenario, the old DRS migrates some VMs with vMotion to balance the Active Memory distribution (Figure 23).

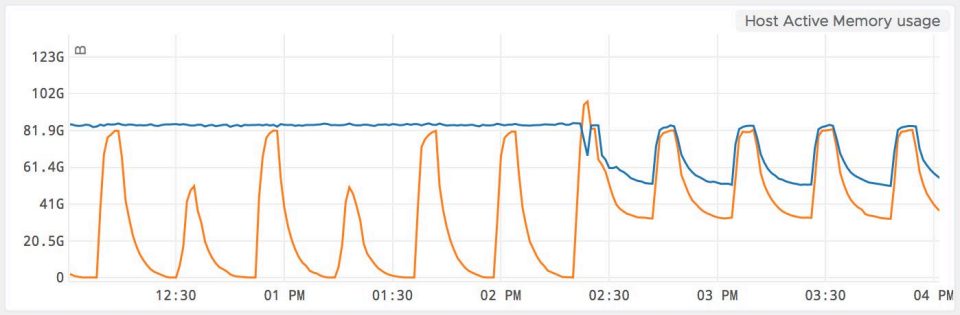

Figure 22. Varying Active Memory usage

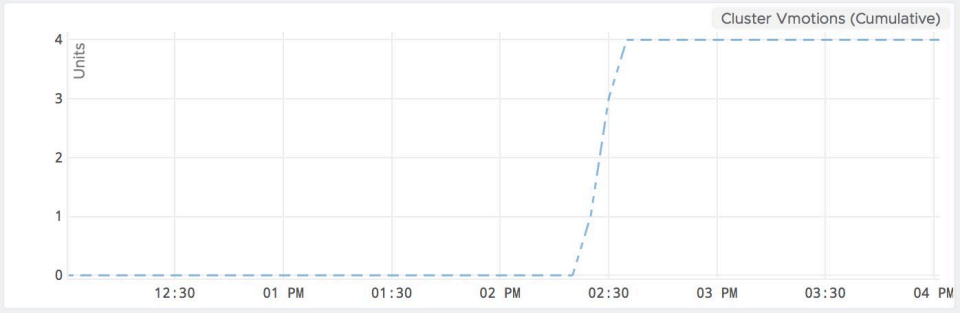

Figure 23. DRS vMotions issued due to Active Memory fluctuations

In a similar scenario, in a cluster running the new DRS, we don't see any DRS vMotions (Figure 24 and Figure 25), because DRS considers Granted Memory for host-memory-load estimation, and not Active Memory.

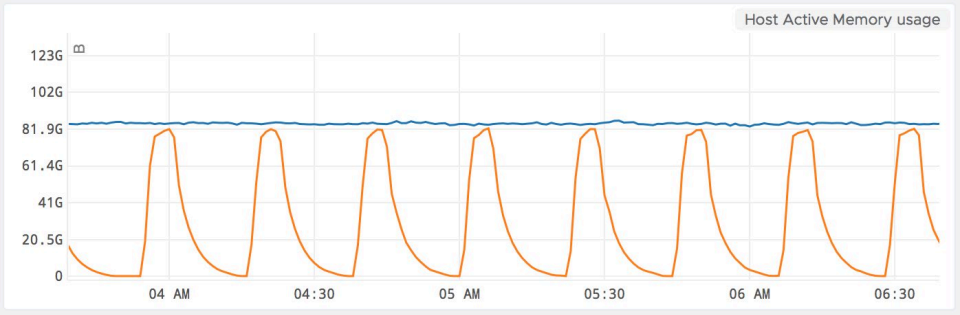

Figure 24. Varying Active memory usage

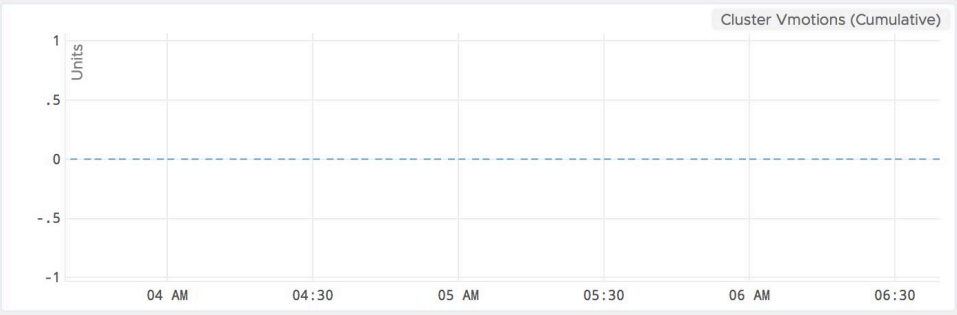

Figure 25. No DRS vMotions due to Active Memory fluctuations

Tuning DRS Sensitivity (Migration Threshold)

Migration threshold is one of the most widely used DRS settings. Based on the cluster balance requirement, in the old DRS, the migration threshold can be tuned to make DRS more aggressive or more conservative. The higher the threshold, the more aggressive DRS works to achieve better cluster balance. As explained earlier, this happens because the old DRS recommends vMotions only if the cluster load imbalance is above a certain target threshold, which is computed based on the migration threshold value that the user has set.

In contrast, the new DRS does not consider cluster standard deviation for load balancing. It is designed to be more VM-centric and workload-centric, rather than cluster-load-centric. As a result, the migration threshold in the new DRS is attuned to cater to different types of workloads, as explained below.

Level 1 – At this level, the behavior is like the old DRS. DRS recommends only vMotions that are mandatory to fix rule violations or those required for a host entering maintenance mode. DRS does not recommend any vMotions to improve VM happiness or cluster load distribution at this level.

Level 2 - This level is for highly stable workloads. When the Migration Threshold is set to this level, DRS will recommend vMotions only in situations that are at, or close to, resource contention. DRS does not recommend vMotions just to improve VM happiness or cluster load distribution. This level is recommended for workloads that are highly stable and are not expected to vary much in terms of resource usage. Even occasional episodes of resource contention could be a cause for concern in such clusters.

Level 3 – This is the default level and is meant for mostly stable workloads. At this level, DRS will recommend vMotions to improve VM happiness and cluster load distribution as well.

Level 4 - This level is for bursty workloads. It is useful when there are occasional bursts in the workload and DRS needs to react to the sudden load changes.

Level 5 - This level is for dynamic workloads. It is useful when the workloads vary a lot. In such cases, DRS will react to the workload changes every time.

VMs in Partially Automated Mode

DRS generally provides 3 different types of operational modes:

- **Manual** – In manual mode, DRS displays the list of recommendations (if any), both when a VM is being powered on (placement) and during load balancing.
- **Partially Automated** – In this mode, DRS will automatically place a VM when it is being powered on, but during load balancing, DRS will display the list of vMotion recommendations for users to select and apply.

- **Fully Automated** – In this mode, DRS automatically places a VM on the best-suited host when it is being powered on, and automatically performs vMotions during load balancing.

These modes can be set both at the cluster and at the VM level. The VM-level setting has a higher priority over the cluster-level setting. For example, if DRS is set to fully automated mode at the cluster level, then it applies the recommendations by default. If any VM has an explicit DRS level set to manual or partially automated, then DRS will only generate recommendations for that VM, but it will not apply those recommendations.

In the old DRS, irrespective of the cluster- or VM-level DRS automation settings, DRS will consider all the VMs in the cluster to generate vMotion recommendations. For example, when DRS is in fully automated mode, it considers all the VMs in the cluster to run its load-balancing algorithm. If one of the recommendations is for a VM that is in partially automated or manual mode, then DRS will just display that specific recommendation and let the user apply the recommendation.

In the new DRS, during load balancing, if DRS is in fully automated mode, it only considers VMs that are also in fully automated mode. This is to avoid overriding VM-level partial/manual automation when the cluster is set at a fully automated mode. So now, if a VM is in partially automated or manual mode, DRS will not consider this VM while computing recommendations, unless DRS is also in partially automated or manual mode.

By default, all the VMs take cluster-level DRS automation setting, unless explicitly specified for each VM.

Conclusion

DRS has been upgraded to adapt to a workload-centric resource management model that can support and address modern application needs. In this paper, we have discussed many of the new features and functionalities in DRS and how they impact your workloads, with relevant examples.

We would like to conclude by saying that with new features like network load balancing and faster reaction times, DRS in vSphere 7.0 has become more robust and flexible to cater to the needs of both traditional and modern workloads.

References

[1] Sai Manohar Inabattini, Vikas Madhusudana, and Adarsh Jagadeeshwaran. (2016, November) DRS Performance in VMware vSphere 6.5. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/drs-vsphere65-perf.pdf

[2] VMware, Inc. (2019, March) vSphere 7.0 product documentation. https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.resmgmt.doc/GUID-BFDC988B-F53D-4E97-9793-A002445AFAE1.html

About the Authors

Priyanka Gayam is a performance engineer with the vCenter Server performance group. She works on vCenter Server performance and scalability, with special focus on the Distributed Resource Scheduling (DRS) algorithm. Priyanka holds a bachelor's degree in computer science from the National Institute of Technology at Warangal, 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.

Acknowledgments

The authors would like to specially thank Sai Inabattini, VMware alumnus, for his guidance and co-authoring the paper. The authors would also like to thank Frank Denneman, Ravi Soundarajan, Abhijit Marulaiah Prabhudev, Jay Suresh, and Venu Uppalapati for their invaluable reviews and support of this paper, and Julie Brodeur for her help in compiling the paper.