

DRS Performance and Best Practices

VMware® Infrastructure 3

VMware® Infrastructure 3 provides a set of distributed infrastructure services that make the entire IT environment more serviceable, available, and efficient. Working with VMware ESX 3, VMware VirtualCenter 2, and VMware VMotion, VMware Distributed Resource Scheduler (DRS) dynamically allocates resources to enforce resource management policies while balancing resource usage across multiple ESX hosts.

This paper is intended for readers who understand the architecture and basic concepts of DRS. See the paper “Resource Management with VMware DRS” for these details. (See “Resources” on page 19 for a link.) This paper identifies various scenarios in which you can benefit from DRS and explains how to configure your environment to take best advantage of DRS.

The paper covers the following topics:

- “Overview of DRS Performance” on page 1
- “Resource Allocation for Virtual Machines and Resource Pools” on page 3
- “Scaling the Number of Hosts and Virtual Machines” on page 9
- “DRS Aggressiveness” on page 13
- “Interval Between DRS Invocations” on page 14
- “Cost-Benefit Analysis” on page 15
- “Heterogeneous Host Environments” on page 15
- “Impact of Idle Virtual Machines” on page 16
- “DRS Performance Best Practices” on page 16
- “Monitoring DRS Cluster Health” on page 17
- “Conclusions” on page 19
- “Resources” on page 19

Overview of DRS Performance

VMware Distributed Resource Scheduler improves resource allocation across all hosts and resource pools in a cluster. When you enable a cluster for DRS, VirtualCenter continuously monitors the distribution of CPU and memory resource usage for all hosts and virtual machines in that cluster. DRS compares these metrics to ideal resource utilization—that is, the virtual machines’ entitlements. These entitlements are determined based on the resource policies of the resource pools and virtual machines in the cluster and their current demands. VirtualCenter uses this analysis to perform initial placement of virtual machines, virtual machine migration for load balancing, enforcement of rules and policies, and distributed power management, if distributed power management is enabled.

This study focuses on understanding the effectiveness and scalability of DRS algorithms.

Effectiveness of DRS Algorithms

Our goal in these tests was to see if DRS is effective in improving performance when there is resource contention and if DRS can allocate resources while balancing load in proportion to the resource policies allocated to virtual machines. We evaluated various configurations of the DRS algorithm and compared how effective they are in various use cases.

We examined whether the shares, reservations, and limits of a virtual machine or resource pool are effectively enforced by DRS and how the migrations recommended by DRS affect the performance of the affected cluster as a whole. We used virtual machine throughput (or the throughput of multiple virtual machines) as a measure of how effective DRS is, in the presence of various workloads with various policy settings.

The loads should achieve throughput that is approximately proportional to their CPU and memory entitlements. These resource entitlements are a measure of how many CPU cycles and how much memory the virtual machine or resource pool should receive and are calculated based on the shares, reservations, and limits—the resource policies—set in Virtual Center and the estimated CPU and memory resources demanded by the virtual machine or resource pool. By placing or migrating the virtual machines effectively based on these entitlements, DRS ensures that the virtual machine or resource pool gets what it deserves without violating the shares, reservations and limit policy.

Scalability of DRS Algorithms

DRS supports a certain number of hosts and virtual machines in a DRS cluster. The goal in our study was to see how DRS scaled as we increased the number of hosts and virtual machines. As the number of virtual machines and host increases, the possibilities for balancing with virtual machine migrations also increase. On the other hand, each balancing migration does have a resource overhead. This might effectively improve or worsen cluster throughput as we scaled the environment to include more hosts and virtual machines. Furthermore, throughput improvement depends heavily on the characteristics of the workload—for example, placement of the virtual machines and variability of workloads. We explored a number of different workloads, at various scales, to see how DRS handled each of them.

Dimensions for DRS Performance Analysis

To better understand the performance of DRS, we characterized DRS along several dimensions. These dimensions include:

- Resource allocation for virtual machines and resource pools
- Number of hosts and virtual machines
- Degree of aggressiveness
- DRS periodic frequency
- Heterogeneous host environments
- Cost-benefit analysis of VMotion
- Impact of idle virtual machines

Resource Allocation for Virtual Machines and Resource Pools

DRS ensures that resource policies are satisfied and balances load across a cluster when appropriate. In order to evaluate its performance impact, we used throughput of various workloads as a means of comparison in various cases with a realistic load. DRS algorithms should respond dynamically to the resource allocation changes for virtual machines and resource pools. We set up several experiments to verify this behavior and evaluate DRS effectiveness, seeking to answer the following questions:

- Can DRS automatically reduce the load on an overcommitted host when additional resources are added or available in the cluster and consequently also improve the throughput for the workloads?
- How does DRS respond to a highly imbalanced cluster? How does DRS aggressiveness affect the placement of virtual machines?
- Can DRS effectively enforce resource allocation policies across hosts in a DRS cluster by setting different share values for individual virtual machines?

Test Methodology

This section describes the types of benchmark workloads used, resources demanded, and metrics used to test the impact of resource allocation on DRS.

We setup a homogenous cluster of hosts, each configured with two 2.8GHz Intel Xeon CPUs with hyperthreading enabled. Each host had 4GB of memory. We used a dedicated Gigabit Ethernet network for VMotion and configured DRS to run at the default migration threshold.

For this first set of tests, we used benchmarks to reflect realistic workloads. The stable period for these loads was an hour during which we took measurements. Table 1 lists these workloads.

Table 1. Workloads for Resource Allocation Tests

Virtual Machine Workload	Benchmark	Resource Demand	Metric
Database server	Swingbench	CPU: 850MHz Memory: 700MB	Transactions/sec
Web server	SPECweb 2005	Network I/O: 30Mb/s CPU: 1000MHz	Accesses/sec
File server	Dbench	Disk I/O: 120Mb/s CPU: 550MHz	MB/sec
Java server	SPECjbb 2005	Memory: 1.7GB CPU: 550MHz	New orders/sec
Idle virtual machine	None	None	None

Test Results

To verify that DRS can automatically reduce the loads on an overcommitted host, we ran the five workloads shown in Table 1, each in its own virtual machine, on one host. All virtual machines had equal shares and no reservations or limits. Figure 1 shows this case. Each circle represents one virtual machine, with different letters representing the different workloads, and the boxes representing separate physical hosts. The size of each circle represents how many shares are given to that virtual machine, not the CPU or memory resource demands of the virtual machine. Details on the resource demands are listed in Table 1.

Tests for Load Balancing with Increase in Available Resources

We used a mix of virtual machines that demanded more CPU resources than were available on the host on which we placed them. We measured the throughput for each virtual machine as we added or powered on a new host in the cluster.

The virtual machine workloads in Figure 1 and subsequent diagrams are identified using the following key:

- D—database server
- F—file server
- I—idle
- J—Java application server
- W—Web server

Figure 1. Initial Workload Distribution with DRS Disabled

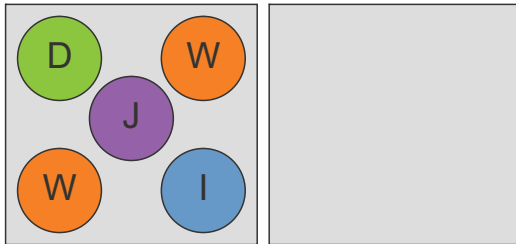
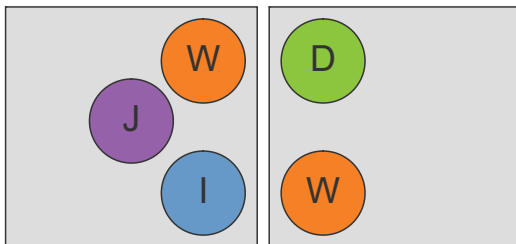


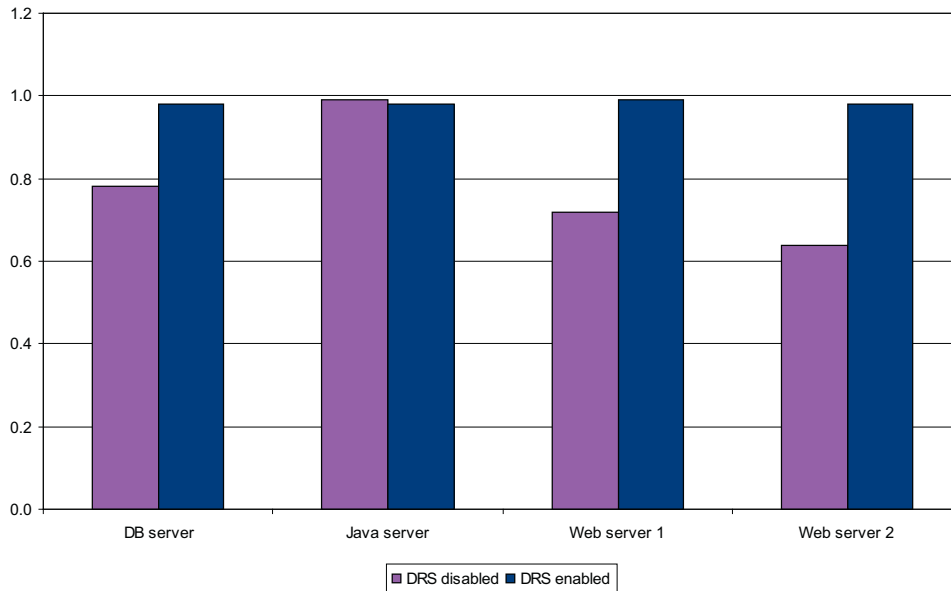
Figure 2. Workload Distribution with DRS Enabled



When we enabled DRS, it moved some of the load on the overcommitted host to the idle host as shown in Figure 2. Using the default migration threshold (moderate), DRS improved overall system throughput (geometric mean of the individual throughput improvements) by 27 percent. These results confirm that simple balancing provided by DRS can automatically bring significant improvements in throughput without the need to place virtual machines manually.

CPU-intensive loads improved significantly, because system CPU resources were overcommitted before we enabled DRS. Figure 3 shows the normalized throughput improvement for individual virtual machines used in this test. The database server throughput improved by 25 percent because more CPU resources were available after we enabled DRS. Web server throughput improved significantly after we enabled DRS. Because the Web server is a network-intensive load, it benefited from DRS because the network I/O causes increased CPU load. On the other hand, the Java application server throughput remained the same, because the Java server workload is memory intensive, and the system did not experience any memory contention during the test. We did not use a file server workload for this test, because file server demands a relatively low level of CPU resources. Instead, we used a second instance of the Web server workload for this test, which requires more CPU resources than the file server workload does.

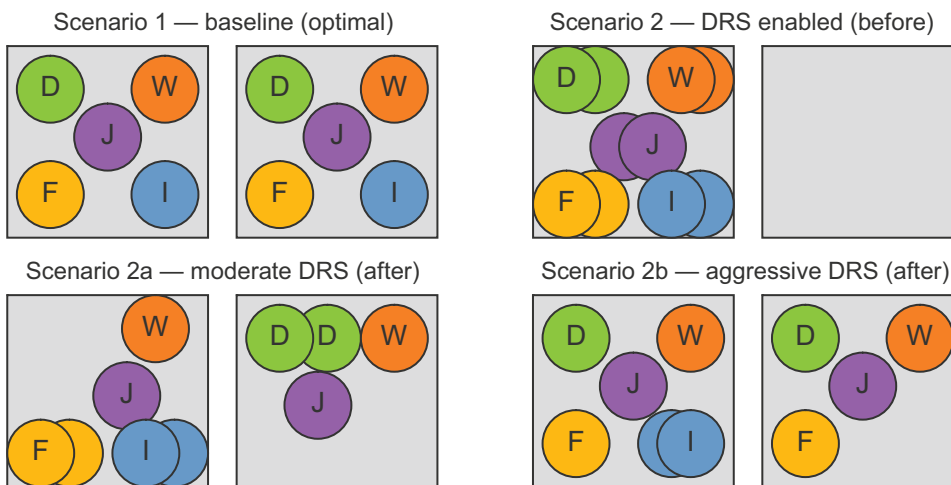
Figure 3. Throughput Normalized to Baseline Individual Workload Throughput



Tests for Robust Response to Highly Imbalanced Cluster

To further validate the robustness of the DRS algorithms, we then ran 10 workloads, two of each of the loads described in Table 1, on two separate ESX hosts. Once again, we gave all virtual machines equal shares and no reservations or limits. Figure 4 depicts the scenarios that we attempted to validate during this test.

Figure 4. Distribution of Workloads When DRS Aggressiveness Is Adjusted



Scenario 1 shows the baseline or balanced throughput configuration. We chose this as the balanced arrangement because putting one of each virtual machine per host balances the CPU, memory, and I/O demands evenly while ignoring locality or caching effect. We used this as our baseline and we expected DRS to achieve a throughput close to this.

Scenario 2 depicts the worst possible configuration. We start with this configuration to see how DRS balances the load of these virtual machines.

We performed the test using default (moderate) and aggressive migration threshold settings to test whether the resource allocation that DRS generates can match the baseline configuration. The worst possible configuration could not be sustained on one host without failures in the benchmark metrics because of resource constraints. So we inserted a delay between the start of one virtual machine and the next to allow DRS to balance resources as we added each new load to the first host.

Our results showed that the virtual machine placement as well as system throughput with DRS was very close to that of the baseline configuration. The overall system throughput was 95 percent of that of the baseline configuration for the DRS configuration with a moderate threshold and 99 percent of that of the baseline configuration with the aggressive threshold setting. This includes the impact on throughput caused by the virtual machine migrations needed to get to the final state of the cluster. Because the loads were constant, the final state was achieved quickly after the loads reached stable state and no further migrations were made.

For the test with the default threshold setting (moderate), DRS performed four migrations—that is, it moved one Web server, one Java server, and the two database servers.

The test using the aggressive threshold yielded an almost baseline configuration. This test performed more migrations—a total of eight—to get to this state. The number of migrations reflects the fact that DRS load balancing occurs while the load is ramping up, and hence in the more aggressive case, some migrations were reverted after all loads reached steady state. The idle virtual machine was not migrated because the cost of migrating the virtual machine could not be justified, given that there was no load. Because the loads are relatively stable over time, we obtained better throughput with an aggressive threshold. However, the default setting in DRS is moderate in order to accommodate changing workloads.

Figure 5 compares the system throughput for individual workloads in the baseline case, using DRS with a moderate threshold, and using DRS with an aggressive threshold. We normalized all throughput results to the baseline case for easier comparison and measured the results over a stable period of an hour.

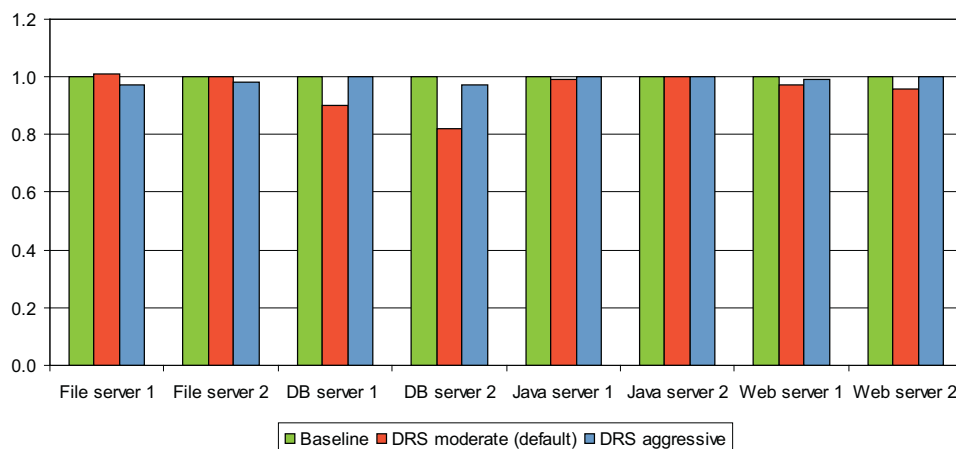
In the moderate case, all individual throughputs are within 80 percent of baseline and overall within 95 percent (geometric mean of all the relative throughputs) of baseline. This is true despite the facts that DRS has an opportunity to rebalance only once every five minutes (default) and that it is affected by VMotion overhead. When we used DRS with a moderate threshold, the database servers were the most affected because DRS placed both database virtual machines on the same host. Migrating one of the database loads to a different host was not justified under the moderate threshold because the imbalance across hosts was not great enough to justify it. Some workloads achieved better throughput than the baseline because additional resources were available on the host on which the workload was running.

In the more aggressive case, the results are different. Because each workload was fairly constant, over time DRS aggressive rebalancing produced better throughput.

If the loads vary, DRS with a moderate threshold may achieve better throughput in the long run because it would recommend fewer migrations and hence incur a lower VMotion cost compared to DRS with an aggressive threshold.

When we increased the number of hosts, as described in [“Scaling the Number of Hosts and Virtual Machines”](#) on page 9, DRS with a moderate threshold achieved throughput comparable to that of DRS with an aggressive threshold and did so with fewer migrations.

Figure 5. Relative Change in Throughput of Workloads

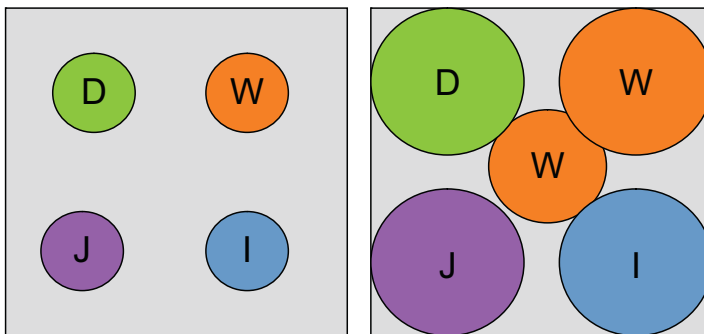


Enforcing Resource Allocation Policies

In a VMware Infrastructure 3 environment, you can allocate resources using absolute memory and CPU bounds (reservation and limit), proportional values (shares), or a combination of the two for the virtual machines and resource pools in the cluster. A reservation enforces a guaranteed minimum for resources and a limit enforces a hard maximum. Share settings can be low (1×), normal (2×), high (4×), or custom. You can assign different share values for CPU, memory, or both to ensure proportional resource allocation. ESX provides resources while ensuring the allocation does not violate the assigned reservations and limits (in Hz or bytes).

To test the effectiveness of DRS in combination with resource allocation features, we ran various types of workload virtual machines with different share values. Figure 6 shows the nine virtual machines running on two hosts before DRS is enabled. The size of each circle in this figure reflects the number of static shares assigned to the workload, not the actual resource usage of each workload. Hence, workloads with different resource usage could have circles of the same size. The shares are assigned in the ratio of 1:2:4. We located all low (1×) share virtual machines on the same host at the start of the test and located the virtual machines with normal (2×) and high (4×) shares on a different host.

Figure 6. Initial Distribution of Virtual Machines with High Resource Contention



We assigned the three Web server virtual machines high, normal, and low shares. We placed the two Web servers with high and normal shares on the same host, hence they share resource from one host. The virtual machine with the low share setting has no other Web server to contend with, hence it gets relatively more CPU resources because it encounters less contention.

After we enabled it, DRS balanced the workloads to give the final configuration as shown in Figure 7. DRS detected that the Web server with high shares was not receiving the resources it was entitled to receive. In fact, as shown in Figure 8, it was receiving a lower allocation than the virtual machine configured for a low share was receiving. DRS swapped the Web server virtual machines with low and high shares. This allowed the Web server configured for a high share to take advantage of the resources available on the other host. The swap also freed resources for the other virtual machines with high shares. Thus, DRS balanced the Web server loads across the two hosts and satisfied the overall system shares for all nine virtual machines.

Figure 7. Balanced Distribution of Virtual Machines After DRS Swaps Two Web Servers

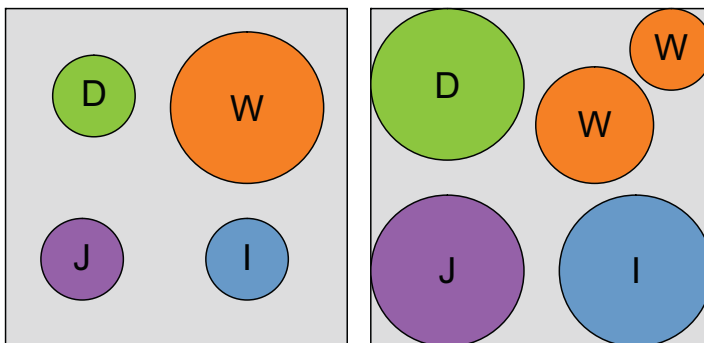


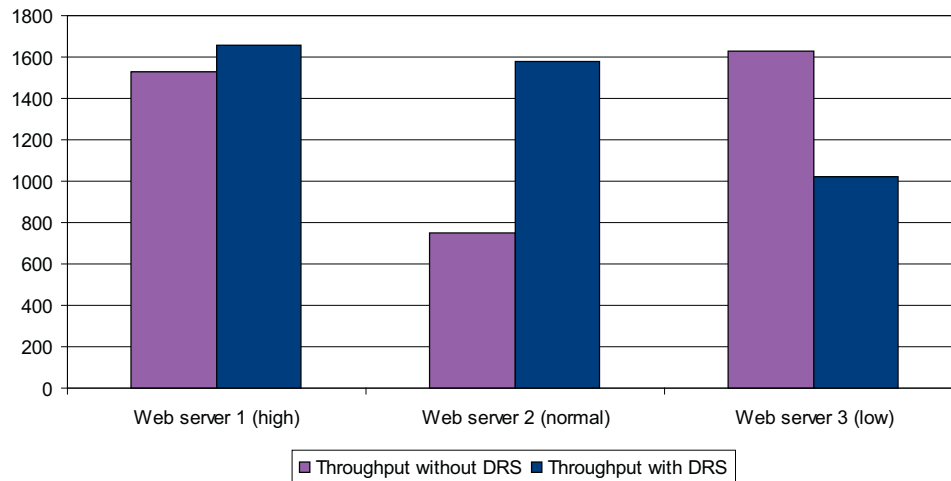
Figure 8. Web Server Throughput

Figure 8 shows the change in Web server virtual machine throughput before and after we enabled DRS. Before we enabled DRS, the Web server virtual machine with low shares achieved twice the throughput of the Web server virtual machine with normal shares. Without DRS, the scope of resource sharing is limited to a single host. Because the Web server virtual machines with high and normal shares ran on the same host, their respective throughput numbers reflected their relative shares as managed by the local ESX resource scheduler, which cannot schedule resources across hosts. However, after we enabled DRS, the virtual machine throughput numbers reflected the relative virtual machine shares for all virtual machines. DRS enforces resource sharing across hosts by controlling the host-level resource settings.

The idle and Java application workloads did not change because they do not have any significant CPU usage and there is no memory overcommitment or imbalance. Because CPU is the resource facing contention in this scenario, the Web server and database workloads were good candidates for migration. DRS migrated the Web server with high shares to the host where virtual machines were entitled to fewer resources because DRS estimated that the migration would bring better balance at minimal migration cost. Later, DRS balanced CPU usage by moving the low shares Web server virtual machine to the second host to further balance the entitlements.

Resource Allocation Recommendations

The following recommendations can help you improve resource allocation in your DRS clusters.

- Shares take effect only when there is contention for resources. Hence, if CPU resources on a host are not fully committed, each virtual machine or resource pool on that host gets all the CPU resources it demands. Similarly, if resources on a DRS cluster are not fully committed, virtual machines or resource pools receive the resources they demand, so long as the following conditions are met:
 - The DRS cluster has no unapplied manual recommendations.
 - No virtual machine or resource pool limits are set to values below what the virtual machine workload demands.
 - No virtual machine or resource pool reservations are set in such a way that they prevent any balancing migration from an overcommitted host.
 - No affinity and anti-affinity rules are set in such a way that they prevent any balancing migration from an overcommitted host.
 - No host VMotion incompatibilities such as CPU incompatibilities, lack of a shared virtual machine or VMotion network, or lack of a shared virtual machine datastore prevent any balancing migration from an overcommitted host.

- You might waste idle resources if you specify a resource limit for one or more virtual machines. The system does not allow virtual machines to use more resources than the limit, even when the system is underutilized and idle resources are available. Specify a limit only if you have specific reasons for doing so.
- For resource pools, resource allocations are distributed among their sibling virtual machines or resource pools based on their relative shares, reservations, and limits.
- The expandable reservation setting for resource pools allows the reservation to go above its initial value if the reservation of the child virtual machines or resource pools increase. Setting a fixed reservation prevents any increase in the child virtual machine or resource pool reservations.
- Virtual machines have some overhead memory that the system must account for in addition to the memory used by the guest operating system. This memory is charged as additional reservation on top of the user-configured virtual machine reservation. Hence, when you set the reservations and limits for resource pools, you should keep a buffer for the extra overhead reservation needed by the virtual machines. See the *Resource Management Guide* for estimated memory overhead. (See “Resources” on page 19 for a link.) The amount depends on virtual machine memory size, the number of virtual CPUs, and whether the virtual machine is running a 32-bit or 64-bit guest operating system. For instance, a virtual machine with one virtual CPU and 1024MB of memory has a virtual machine overhead of approximately 98MB for a 32-bit guest operating system or approximately 118MB for 64-bit guest operating system. Also, the overhead reservation may grow with time depending on the workload running in the virtual machine. This overhead growth is designed to ensure optimal performance within the guest operating system.
- Resource pools make managing resource allocations much easier by grouping virtual machines, resource pools, or both within each resource pool. This allows administrators to apply settings across many resource entities easily and to build a hierarchical structure that makes it easy to allocate their resources. This paper does not address the use of resource pool hierarchies because they do not have any impact on virtual machine performance.

Scaling the Number of Hosts and Virtual Machines

Scaling of the number of hosts and virtual machines affects the resource distribution and load balancing opportunities available to DRS. Our testing examined the impact of size of the cluster on the throughput of the system. Increasing the number of hosts in a cluster gives DRS more locations where it can place virtual machines, and consequently affects the output of DRS algorithms. Furthermore, a system with more hosts normally has more virtual machines, as well. Increasing the number of virtual machines means DRS can move more entities around. Each of these virtual machines might be running different workloads at different times.

Increasing the number of hosts and virtual machines does increase the complexity of various DRS computations, but at the same time, the increase provides opportunity for more efficient resource distribution. However, as these opportunities increase, the number of migrations may increase and as a result may incur additional overhead.

We accounted for this overhead in our experiments in order to evaluate the scaling. Resource pools can simplify the task of allocating resources across the DRS cluster, especially when dealing with a large number of virtual machines and hosts. However, scaling the number of resource pools does not affect performance of the virtual machines so long as the resource policies are effectively allocated identically. Hence, this section does not consider resource pool scaling.

Test Methodology

The objective of the test was to measure the effectiveness of the DRS algorithms as we scaled to larger environments, while ensuring that the overall cluster utilization remained constant—that is, that the ratio of overall virtual machine demand to total cluster resources remained constant.

We set up a homogenous cluster of hosts each configured with two 2.8GHz Intel Xeon CPUs with hyperthreading enabled. Each host had 4GB of memory. We dedicated a Gigabit Ethernet network for VMotion and configured DRS to run at the default migration threshold (moderate). Each host had three workload virtual machines, each with different workloads.

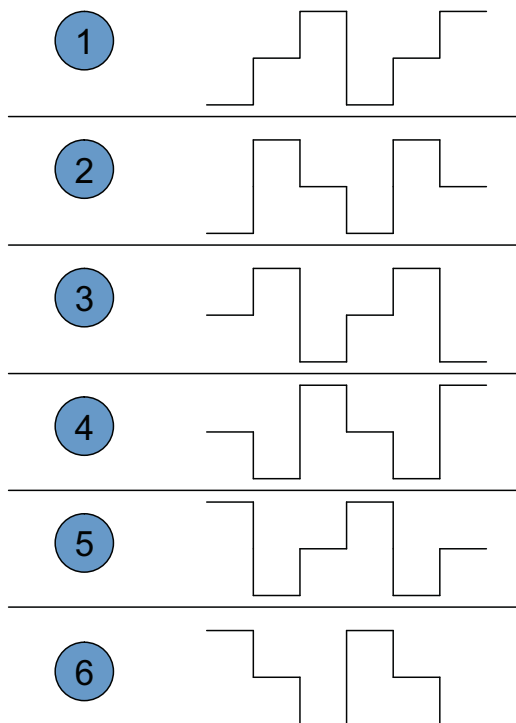
Table 2. Test Environment for Scaling Tests

Number of Hosts	Number of Virtual Machines	Number of Each Constant Workload	Number of Each Varying Workload
2	6	2	1
4	12	4	2
8	24	8	4
16	48	16	8

We configured each virtual machine with a single virtual CPU and 1GB of RAM and installed Red Hat Enterprise Linux 3 as the guest operating system. For simplicity, we gave each virtual machine the same shares and set no reservations or limits. We ran one of the following categories of workloads inside each virtual machine:

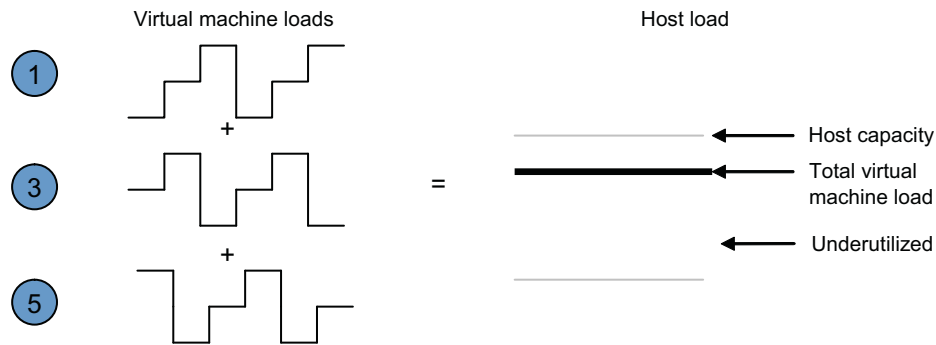
- Constant workload—used a fixed amount of CPU resources of 10 percent, 50 percent, or 90 percent of CPU (of one virtual CPU).
- Varying workload—used a varying amount of CPU resources, changing with time. For easy analysis, the workload varied among three steps, using 10 percent, 50 percent, or 90 percent of CPU, with each step lasting five minutes. We used all possible combinations of the three-step load to generate six different workload types, as shown in Figure 9. Our results with these varying workloads demonstrate that DRS performs effectively even if loads constantly change over time and no stable state is reached. As our tests showed, this is one of the main advantages of using DRS.

Figure 9. Patterns of Varying Workloads During Tests



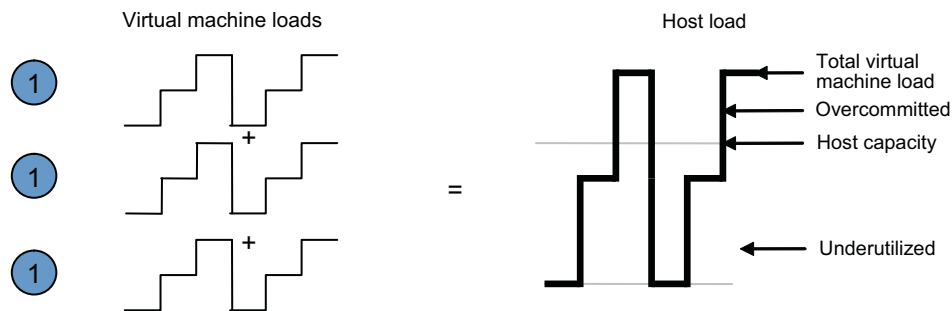
We distributed various combinations of the varying workload virtual machines across hosts to achieve different patterns that reflect three host usage scenarios.

Figure 10. Workload on a Host with Optimal Placement



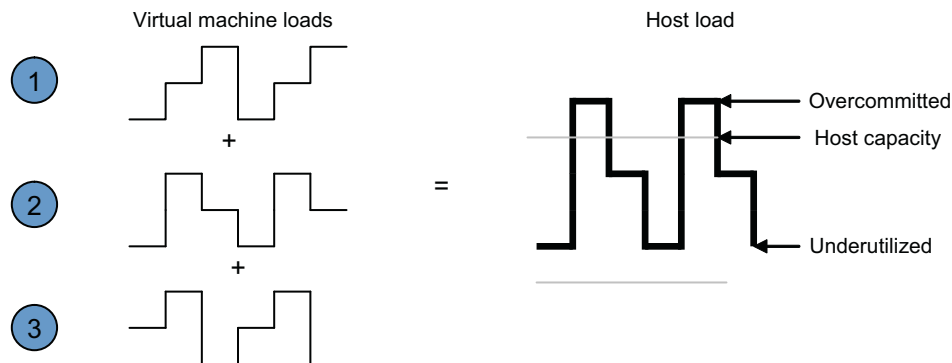
In the optimal placement scenario, shown in figure 10, we placed compensating loads together on each host. With this placement, all peaks matched the troughs of other virtual machines on the same host. As a result, none of the hosts are overcommitted and the load is balanced across all hosts in the cluster. The throughput achieved in this scenario is used as the baseline for optimal throughput in the tests that follow.

Figure 11. Workload on a Host with Bad Placement



In the bad placement scenario, shown in Figure 11, we placed similar workloads together on each host wherever possible. As a result, peaks and valleys occur together on a host, resulting in a highly imbalanced cluster. Consequently, the hosts are either overcommitted or highly undercommitted. This results in not only major load imbalance across hosts but also poor performance in the virtual machines running on overcommitted hosts. As shown in Table 2, we used increasing numbers of virtual machines with similar workloads as we increased the size of the cluster. As a result, we saw greater imbalance for a bad placement as the cluster scaled up.

Figure 12. Workload on a Host with Symmetric Placement



In the symmetric placement scenario, the cluster is moderately imbalanced, with a mix of hosts that are somewhat overcommitted and hosts that are somewhat undercommitted. We call this symmetric placement because we use the same load distribution for each pair of hosts as we scaled up. As a result, the load imbalance remained the same for all stages of our tests.

With the smaller configurations, there are fewer workloads of the same type and hence the load distribution cannot be as flexible as it is with the larger setups.

Test Results

Our results show that DRS improves host load balance across a cluster. The load balance improvement improves as the number of hosts and virtual machines in the cluster increases. Furthermore, throughput also improves as the number of hosts and virtual machines increases. DRS performs slightly better with more hosts because there are more opportunities for balancing loads. As the size of the cluster increased, the number of migrations increased proportionally, but DRS ensured that the migrations would benefit throughput, hence the average virtual machine throughput improved. DRS did not affect throughput when the virtual machines were placed in the optimal configuration, because the cluster was already balanced. The key results presented in this section highlight these findings.

Figure 13 shows the results when virtual machines were distributed in a bad placement—that is, a highly imbalanced cluster at the beginning of the experiment with each of the virtual machine workloads varying every five minutes. Because DRS computed loads and options for balancing the loads every five minutes, this was a worst case workload. Soon after DRS finished balancing the load, the virtual machine loads changed.

As shown in Figure 13, DRS improves virtual machine throughput even when it must balance workloads that vary greatly. At all environment sizes, the average virtual machine throughput stayed within 96 percent of the optimal throughput. DRS achieved these results despite the fact that it was using VMotion, with its attendant overhead, to balance the load. Furthermore, the throughput improvement compared to the initial condition, without DRS, increased as we scaled to larger environments. This result mainly reflects the fact that as we scaled up, the number of similar workloads increased. The higher number of similar workloads, in turn, caused greater imbalance with bad placement in larger environments.

Figure 13 shows the results when we set DRS to the default aggressiveness (moderate). When we set DRS to a more aggressive threshold, we observed many more migrations, but throughput improvements were similar to those achieved with the moderate threshold. In a separate test, in which the virtual machines were running constant loads, we observed even better throughput improvements.

Figure 13. Throughput with Varying Load and Bad Placement

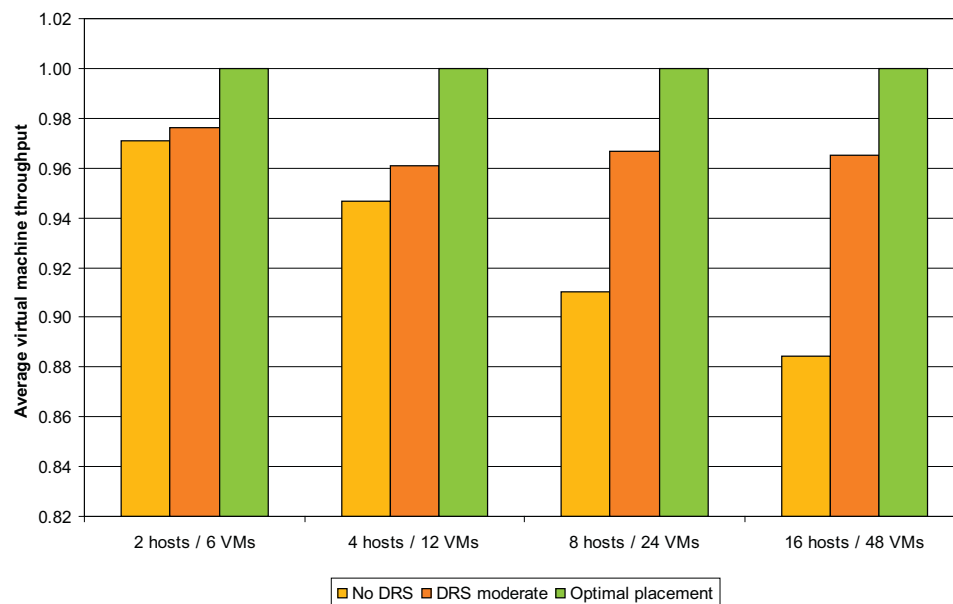
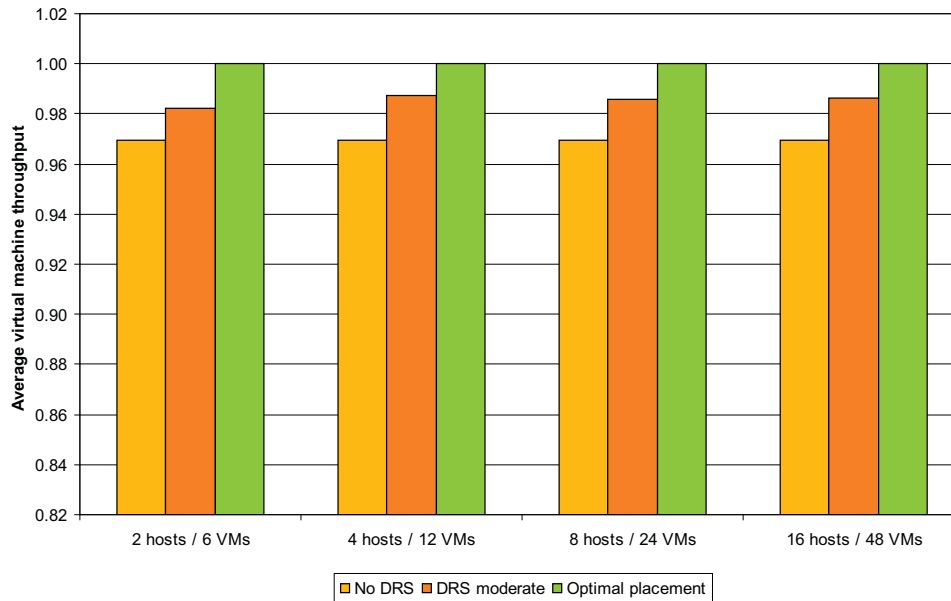


Figure 14 shows the results when we distributed virtual machines in a symmetric placement—that is, a somewhat imbalanced cluster at the beginning of the test with each of the virtual machine workloads varying every five minutes. As Figure 14 shows, the initial condition, without DRS, achieves the same virtual machine throughput at all environment sizes because the virtual machine distribution is symmetric. In each size environment, when we enabled DRS at a moderate threshold, it brought the average throughput to within 98 percent of optimal. Increasing the environment size brought minimal improvement in throughput, but the main reason optimal throughput was not achieved was the overhead of VMotion as DRS balanced the load. Overall, DRS achieved some performance gains by taking advantage of smaller opportunities for avoiding overcommitment.

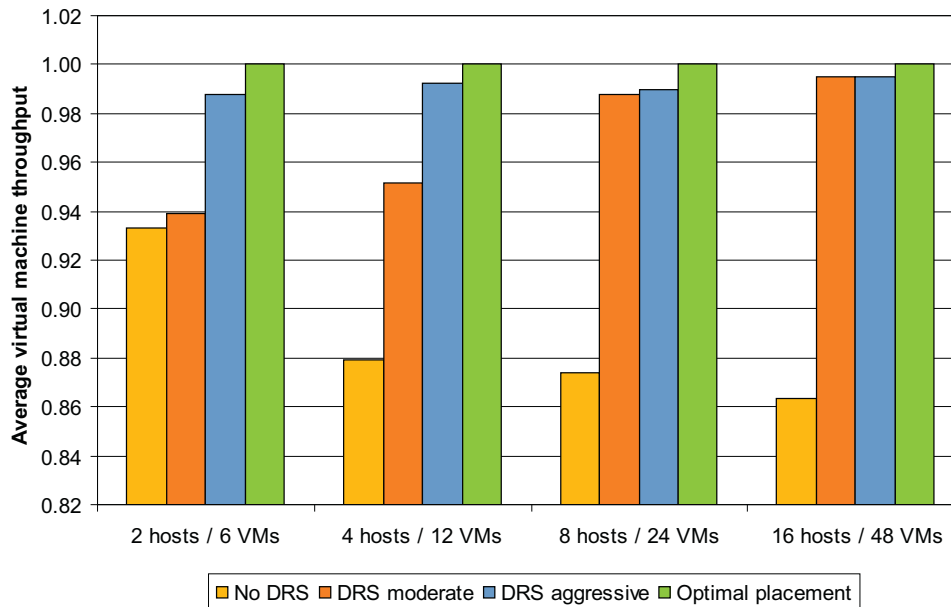
Figure 14. Throughput with Varying Load and Symmetric Placement

DRS Aggressiveness

Degree of aggressiveness for DRS controls how aggressively DRS acts to balance resources across hosts in a cluster. The degree of aggressiveness is set for the entire cluster. The five options range from conservative (level one) to moderate (level three) to aggressive (level five) and control the degree to which DRS tries to balance resources across hosts within the cluster.

DRS makes VMotion recommendations based on what is referred to as their goodness value—that is, how much improvement in load balancing the action can achieve. The goodness value is translated to a rating between one star and five stars, and DRS executes migrations based on the aggressiveness threshold set for a cluster. In VMware Infrastructure 3, a migration receives a goodness value of five stars if it is recommended to resolve affinity, antiaffinity, or reservation rules. For each lower star rating the balancing impact of the migration is less. So a migration with a rating of four stars would improve the load balance more than a migration with a rating of three stars, two stars, or one star.

Figure 15 shows how throughput compares for the moderate and aggressive levels for the mix of constant load virtual machines and bad placement described in [“Scaling the Number of Hosts and Virtual Machines”](#) on page 9. The results show that the aggressive threshold achieved slightly better throughput compared to the moderate setting. However, as we scaled to larger configurations, the throughput was almost the same for both settings. When we used the aggressive setting, we saw more migrations than we did when we used the moderate setting. When we used varying workloads, not only were there more migrations in the aggressive case but the throughput differences for the moderate and aggressive settings were even smaller.

Figure 15. Throughput for Constant Load and Bad Placement

Interval Between DRS Invocations

VirtualCenter activates the DRS algorithm after a fixed interval (the default is five minutes) so DRS can make recommendations based on the past performance metrics of the virtual machines in the DRS cluster. This section discusses the impact on the DRS algorithm of changing this interval. The DRS calculations are invoked between the regularly scheduled invocations if there are changes that require DRS decisions—for example, if you change resource allocation settings. Our discussion ignores these intermediate invocations because they depend on the frequency with which you perform administrative operations in the cluster.

Our results show that choosing the appropriate DRS periodic frequency depends on the periodicity of the workloads within the virtual machines in the cluster. Fairly constant loads may benefit from frequent DRS invocations so that immediate action is taken when the load does change. However, we do not recommend an interval of less than five minutes because of the way the algorithm uses statistics. We observed no notable benefit when we increased frequency to more than once every five minutes, and running the DRS algorithm very often causes unnecessary overhead. In addition, an interval of less than five minutes might be too aggressive given that VMotion migrations can take on the order of tens of seconds to complete, depending on the load running in the virtual machine. (The guest operating system runs normally for the most of the VMotion operation and generally incurs a downtime of less than one second.)

If you notice very frequent migrations, you might feel the need to increase the invocation interval. However, before doing so, you should check the DRS aggressiveness setting. Decreasing the aggressiveness might be the better choice. When a high number of migrations is recommended in one DRS invocation, this is usually the result of an aggressive threshold setting.

If a cluster includes virtual machines with memory-intensive workloads, an infrequent invocation setting might not allow DRS to respond to memory pressure soon enough and the virtual machine might start ballooning or swapping unnecessarily. This can significantly degrade performance. However, because an increase in memory use is typically a more gradual process, keeping the default of five minutes does not affect performance for a majority of workloads. DRS also prioritizes memory balancing migrations over CPU balancing migrations when memory commitment is high in order to prevent unnecessary ballooning or swapping.

Although we recommend keeping the default value of five minutes, you can change the frequency by adding the following options in the `vpxd.cfg` file, changing 300 to the desired number of seconds.

```
<config>
...
  <drm>
    <pollPeriodSec>
300 <!--# of seconds desired between 60 and 3600 -- >
    </pollPeriodSec>
  </drm>
...
</config>
```

Cost-Benefit Analysis

Cost-benefit calculations act as a pivot in determining whether a single migration across hosts is beneficial or not. Using the history of virtual machine workloads, DRS determines the expected workload of the virtual machines on each host during interval between its calculation and the next time DRS will be invoked. Before migrating any virtual machine, it calculates the gain achieved by migrating the virtual machine minus the expected migration cost (VMotion requires resources) and accounts for the potential worst-case workload of the virtual machine after it migrates to the destination host. The migration is allowed only if the total gain is positive.

Virtual machines with highly varying workloads can benefit from using cost-benefit analysis by not migrating often, only when DRS calculates that the target virtual machine and other affected virtual machines will benefit from the move before DRS would be invoked again. We tested several scenarios in which cost-benefit analysis is either turned on or turned off while the cluster runs varying workloads. The results demonstrate that with cost-benefit analysis enabled, DRS makes better decisions. We observed very few unnecessary migrations (unnecessary migrations include ping-pong migrations back and forth between two hosts as load changes) when cost-benefit analysis was enabled. Redundant migrations might occur in early rounds, as we observed in our tests, because DRS needs to adapt to understand the migration costs, which vary for particular virtual machines and network environments. We recommend that you leave the cost-benefit algorithm enabled — the default setting.

Heterogeneous Host Environments

Heterogeneous host environments introduce additional complexities that can affect the performance of a DRS cluster. This section covers several issues to consider, such as VMotion compatibility and architectural differences.

If you are configuring a DRS cluster with heterogeneous hosts, be sure to consider the following factors:

- VMotion hardware compatibility — Virtual machines cannot use VMotion to migrate across different CPU types (from Intel to AMD or vice versa). Also, there may be CPU incompatibilities across different generations from the same manufacturer that can also prevent VMotion — for example, CPUs that are SSE-enabled to those that are not SSE-enabled. However, in VMware Infrastructure 3 beginning with version 3.5 Update 2, Enhanced VMotion Compatibility gives you the ability to ensure VMotion compatibility for the hosts in the cluster by presenting the same CPU feature set across hosts, even when the actual CPUs on the hosts differ. See “VMware VMotion and CPU Compatibility” for details. See [“Resources”](#) on page 19 for a link.
- Hardware differences across compatible hosts — Even if hosts are VMotion compatible, they may be running at different clock speeds, use different hyperthreading settings, or have different memory subsystem architectures — for example, different cache sizes or NUMA enabled on one and disabled on the other. As a result, the effective resource entitlements of virtual machines might be different on different hosts, and these differences might lead to differences in performance across hosts.

- Highly skewed host sizes
 - Virtual machines might not fit on all hosts. For example, virtual machines with four virtual CPUs cannot migrate to hosts with only two physical CPUs.
 - DRS tends to favor migrating virtual machines to a host with more memory or CPU resources to balance memory or CPU loads. Essentially, the algorithm tries to balance the relative utilizations for each host. So if a cluster has virtual machines with identical loads, DRS typically places more virtual machines on a host that has more CPU or memory resources. If the virtual machine loads are primarily CPU-intensive, the host memory sizes do not really matter. Conversely, if the loads are memory-intensive, CPU capacities do not matter.
- Host configuration—The configuration of the host network or datastore may disallow migration using VMotion because not all hosts share the same virtual machine or VMotion network or datastore.

DRS does a reasonable job in balancing load across heterogeneous hosts so long as the cluster is made up of a subset of hosts that are VMotion compatible. The factors discussed in this section may limit benefits. We recommend a homogenous cluster wherever possible.

Impact of Idle Virtual Machines

Based on the tests we performed in our labs with DRS, in environments ranging from small to large, idle virtual machines were generally not migrated because they did not use significant resources. However, in an imbalanced DRS cluster with aggressive mode, a large number of idle virtual machines per host, or both, DRS would migrate the idle virtual machines because they have some memory overhead. Furthermore, some operating systems use CPU cycles for idling—for example, for delivering guest timer interrupts—and this can also affect DRS migration recommendations.

DRS Performance Best Practices

Clustering configurations can have significant impact on DRS performance. VMware recommends the following DRS configurations and practices for optimal performance:

- When deciding which hosts to group into a DRS cluster, try to choose hosts that are as homogeneous as possible in CPU and memory. This ensures higher performance predictability and stability. VMotion is not supported across hosts with incompatible CPUs. Hence with heterogeneous systems that have incompatible CPUs, DRS is limited in the number of opportunities for improving the load balance across the cluster. To ensure CPU compatibility, systems should be configured with CPUs from the same vendor, with similar CPU family and SSE3 status. However, in VMware Infrastructure 3 beginning with version 3.5 Update 2, Enhanced VMotion Compatibility gives you the ability to ensure VMotion compatibility for the hosts in the cluster by presenting the same CPU feature set across hosts, even when the actual CPUs on the hosts differ. See “VMware VMotion and CPU Compatibility” for details. See “[Resources](#)” on page 19 for a link.

DRS performs initial placement of virtual machines across all hosts even if they are not VMotion compatible.

If heterogeneous systems do have compatible CPUs but have different CPU frequencies, differing amounts of memory, or both, the systems with more memory and higher CPU frequencies are generally preferred, all other things being equal, as hosts for virtual machines, because they have more room to accommodate peak load.

Using machines with different cache or memory architectures may cause some inconsistency in performance of virtual machines across those hosts.

- When more ESX hosts in a DRS cluster are VMotion compatible, DRS has more choices to better balance workloads across the cluster. We recommend clusters of up to 32 hosts.

Besides CPU incompatibilities, some misconfigurations can make two or more hosts incompatible. For instance, if the hosts' VMotion network adapters are not connected by a Gigabit Ethernet link, the VMotion migration might not occur between the hosts. You should also be sure the VMotion gateway is configured correctly, VMotion network adapters on the source and destination hosts have compatible security policies, and the virtual machine network is available on the destination host. See the *Resource Management Guide* and the *ESX Server 3 Configuration Guide* for details. See "[Resources](#)" on page 19 for links.

- The default migration threshold (moderate) works for most configurations. You can set the migration threshold to more aggressive levels when all of the following conditions are satisfied:
 - The hosts in the cluster are relatively homogeneous.
 - The virtual machines' resource utilization remains fairly constant.
 - The cluster has relatively few constraints on where a virtual machine can be placed

You should set the migration threshold to more conservative levels when the converse is true.
- The default DRS frequency is once every five minutes, but you can set it to any period between one and 60 minutes. You should avoid changing the default value. If you are considering a change in the setting, see "[Interval Between DRS Invocations](#)" on page 14.
- In general, do not specify affinity rules unless you have a specific need to do so. In some cases, however, specifying affinity rules can improve performance.
 - Keeping virtual machines together can improve performance if the virtual machines need to communicate with each other, because network communication between virtual machines on the same host enjoys lower latencies.
 - Separating virtual machines maintains maximal availability of the virtual machines.

For example, if two virtual machines are both Web server front ends to the same application, you want to make sure that they do not both go down at the same time.

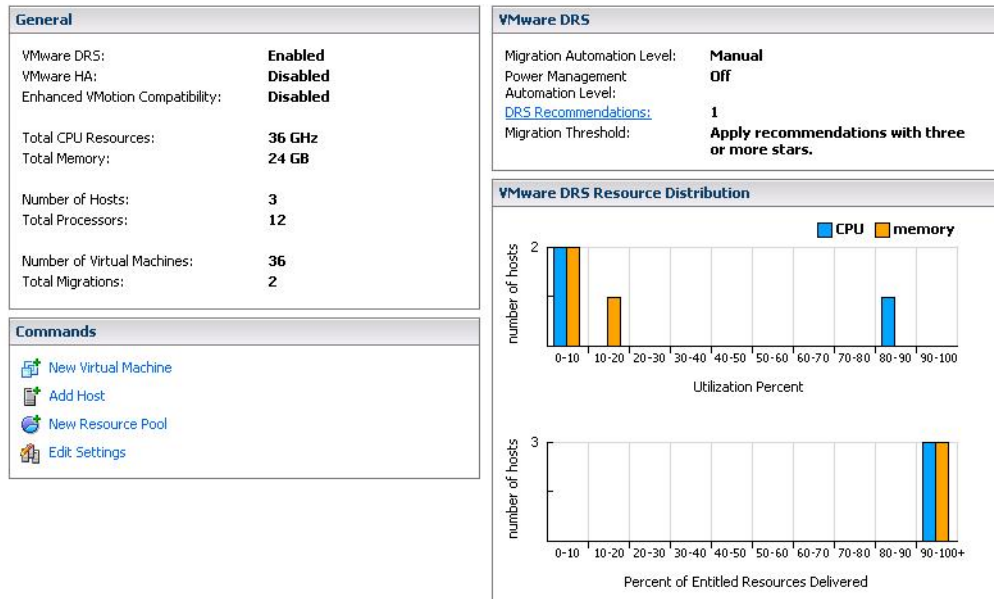
Another example of virtual machines that might need to be separated is virtual machines with I/O-intensive workloads. If they share a single host, they might saturate the host's I/O capacity, leading to performance degradation. DRS does not make virtual machine placement decisions based on their usage of I/O resources.
- Assign resource allocations to virtual machines and resource pools carefully. Be mindful of the impact of limits, reservations and virtual machine memory overhead. See "[Resource Allocation Recommendations](#)" on page 8 for details.
- Virtual machines with smaller memory sizes or fewer virtual CPUs provide more opportunities for DRS to migrate them in order to improve balance across the cluster. Virtual machines with larger memory size or more virtual CPUs add more constraints in migrating the virtual machines. Hence you should configure only as many virtual CPUs and as much memory for a virtual machine as needed.
- You can specify DRS modes of automatic, manual, or partially automated at the cluster level as well as the virtual machine level. We recommend that you keep the cluster in automatic mode. For virtual machines sensitive to VMotion, you can set manual mode at the virtual machine level. This setting allows you to decide if and when the virtual machine can be migrated. Keep virtual machines in DRS automatic mode as much as possible, because virtual machines in automatic mode are considered for cluster load balance migrations across ESX hosts before virtual machines that cannot be migrated without user actions.

Monitoring DRS Cluster Health

In VMware Infrastructure 3, you can monitor a DRS cluster's health by tracking resource utilization of the cluster, load distribution across hosts, and whether the virtual machines are receiving the resource to which they are entitled. This information is displayed in two graphs that are part of the cluster summary display.

Host Utilization Percentage Graph

The host utilization graph, the top graph in the VMware DRS Resource Distribution section, is a histogram that shows the number of hosts on the Y axis and the utilization percentage on the X axis. If the cluster is unbalanced, you see multiple bars, corresponding to different utilization levels. The blue bars represent CPU usage and the orange represent memory usage. For example, in the screen capture below, two hosts are at 0–10 CPU utilization, the third at 80–90 percent CPU utilization, each represented by a blue bar. In this cluster, DRS is in manual mode (moderate aggressiveness) and is making a recommendation. After the recommendation is applied the blue bars become closer to each other on the X axis. The closer the blue and orange bars are to each other, the more balanced the cluster is.



Percent of Entitled Resources Delivered Graph

Chart (Bottom DRS Resource Distribution Chart)

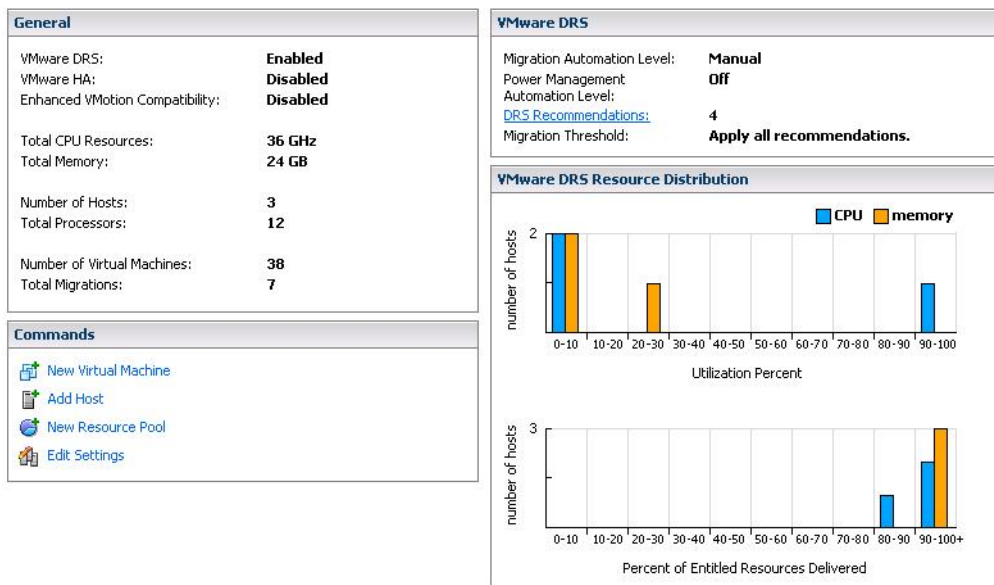
The entitled resources delivered graph, the bottom graph in the VMware DRS Resource Distribution section, is a histogram that shows the number of hosts on the Y axis and the percentage of entitled resources delivered for each host on the X axis. The top graph reports raw resource utilization values. The bottom graph incorporates additional information about resource settings for virtual machines and resource pools.

DRS computes a resource entitlement for each virtual machine, based on virtual machine and resource pool configured shares, reservations, and limits settings, as well as the current demands of the virtual machines and resource pools. What the virtual machine demands is not necessarily what it deserves or is entitled to because of the virtual machine and resource pool configurations.

After computing the entitlements, DRS computes a resource entitlement for each host by adding the resource entitlements for all virtual machines running on that host. The percentage of entitled resources delivered is equal to the host's capacity divided by the entitlements of the virtual machines. This means that in a cluster in which all the entitled resources are delivered, the graph should have a single bar for each resource in the 90–100 percent histogram range. The screen shot in the previous section shows that even though the DRS cluster was not balanced, each of the virtual machines got the resources to which it was entitled.

However, there may be cases in which hosts do not deliver all entitled resources. For example, in the screen capture below, one of the hosts could deliver only 80–90 percent of entitled resources because CPU resources were overcommitted on the host.

In summary, if the bars in this graph appear in the rightmost category, all virtual machines are getting the resources to which they are entitled.



Conclusions

The effectiveness and scalability tests we performed with DRS demonstrate that resources are distributed across hosts in a DRS cluster according to the resource allocations and that better throughputs are achieved especially in the presence of varying loads and random virtual machine placement on hosts. The DRS framework also provides the ability to adjust the aggressiveness of the DRS algorithm and interval between invocations of the algorithm. In addition, DRS avoids wasteful migrations with cost-benefit analysis and minimizes migration of idle virtual machines.

Based on our experience and customer experience in the field, we provided a list of best practices that you can follow to avoid pitfalls.

Finally, DRS also provides a few mechanisms you can use to monitor the DRS cluster performance and potentially provide feedback on how the resources are being delivered and whether the DRS settings need to be adjusted.

Resources

- *ESX Server 3 Configuration Guide*
http://www.vmware.com/pdf/vi3_35/esx_3/r35/vi3_35_25_3_server_config.pdf
- “Resource Management with VMware DRS”
<http://www.vmware.com/resources/techresources/401>
- *Resource Management Guide*
http://www.vmware.com/pdf/vi3_35/esx_3/r35/vi3_35_25_resource_mgmt.pdf
- “VMware VMotion and CPU Compatibility”
<http://www.vmware.com/resources/techresources/1022>

If you have comments about this documentation, submit your feedback to: docfeedback@vmware.com

VMware, Inc. 3401 Hillview Ave., Palo Alto, CA 94304 www.vmware.com

Copyright © 2008 VMware, Inc. All rights reserved. Protected by one or more of U.S. Patent Nos. 6,397,242, 6,496,847, 6,704,925, 6,711,672, 6,725,289, 6,735,601, 6,785,886, 6,789,156, 6,795,966, 6,880,022, 6,944,699, 6,961,806, 6,961,941, 7,069,413, 7,082,598, 7,089,377, 7,111,086, 7,111,145, 7,117,481, 7,149,843, 7,155,558, 7,222,221, 7,260,815, 7,260,820, 7,269,683, 7,275,136, 7,277,998, 7,277,999, 7,278,030, 7,281,102, 7,290,253, and 7,356,679; patents pending. VMware, the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

Revision 20090109 Item: PS-060-PRD-01-01