VMware vCenter Server Performance and Best Practices

VMware vSphere 4.1

VMware vCenter™ Server allows you to manage all levels of a VMware vSphere™ deployment—from datacenters to clusters, hosts, and individual VMs. As a central part of vSphere, vCenter Server meets the rigorous performance demands of an Enterprise-wide IT component. The product contains the performance features expected of software that is part of a world-class virtualization platform, and it offers features that allow administrators to maintain the software's performance. This white paper addresses four areas of common concerns regarding vCenter Server performance:

- Performance improvements in vSphere 4.1 compared to vSphere 4.0
- vCenter Server sizing guidelines and software requirements
- Best practices in performance monitoring, tuning and troubleshooting
- Case studies demonstrating performance improvements in vSphere 4.1.

Contents

1 Overview

This document is structured as follows:

1 Overview—describes the major sections in this document.

2 Introduction—provides an overview of VMware vCenter and its key components.

3 What's New in vCenter Server in vSphere 4.1?—describes at a high level the various performance improvements delivered in vSphere 4.1. Introduces the supported Inventory limits in vCenter Server as well as the limits on concurrent operations in vSphere 4.1 in comparison to those in vSphere 4.0.

4 Hardware Sizing Guidelines and Software Requirements—provides hardware sizing guidelines regarding the minimum hardware resource requirements for vCenter Server, vCenter Server database (also referred to as *database* or *the database*), and vSphere Client, and discusses how these requirements vary for different deployment sizes. We also describe briefly the software requirements for vCenter Server, database, and vSphere Client.

5 Performance Best Practices, Monitoring, Tuning, and Troubleshooting—addresses common questions regarding performance monitoring, tuning and troubleshooting for vCenter. We first give an overview of the hierarchical structure inside a vCenter Server, from datacenters, to clusters, to hosts, and down to individual virtual machines (VMs). We then discuss best practices for creating datacenters and managing hosts in vCenter Server. For clusters, we specifically discuss four important features – HA (high availability), FT (Fault Tolerance), DRS (Distributed Resource Scheduler), and DPM (Distributed Power Management). For vCenter Server database, we describe the types of data stored, sizing guidelines, and performance tuning tips. For vCenter clients, we introduce the UI-based vSphere Client and other SDK clients as well as best practices for improving client performance. For each component of the vCenter architecture, we discuss the common performance problems an administrator may encounter, the UI features the administrator can utilize to diagnose problems, and the configuration options available for tuning and achieving the best performance possible for the user's specific environments.

6 Performance Monitoring and Troubleshooting Tools—describes several utilities useful for determining the root cause of performance problems in vCenter Server and vSphere Client.

7 Case Studies and Performance Improvements—demonstrates the performance improvements in vSphere 4.1 in comparison to vSphere 4.0 with data collected in a number of case studies. In particular, the experimental results show increased performance at higher vCenter Server and cluster inventory limits, faster vCenter Server startup and better client scaling, faster host and VM operations in standalone hosts, quicker VM recovery after host failures, as well as better load balancing in a DRS cluster.

8 Conclusion—summarizes some main points of this paper.

9 References—provides links to the documents referenced in this paper.

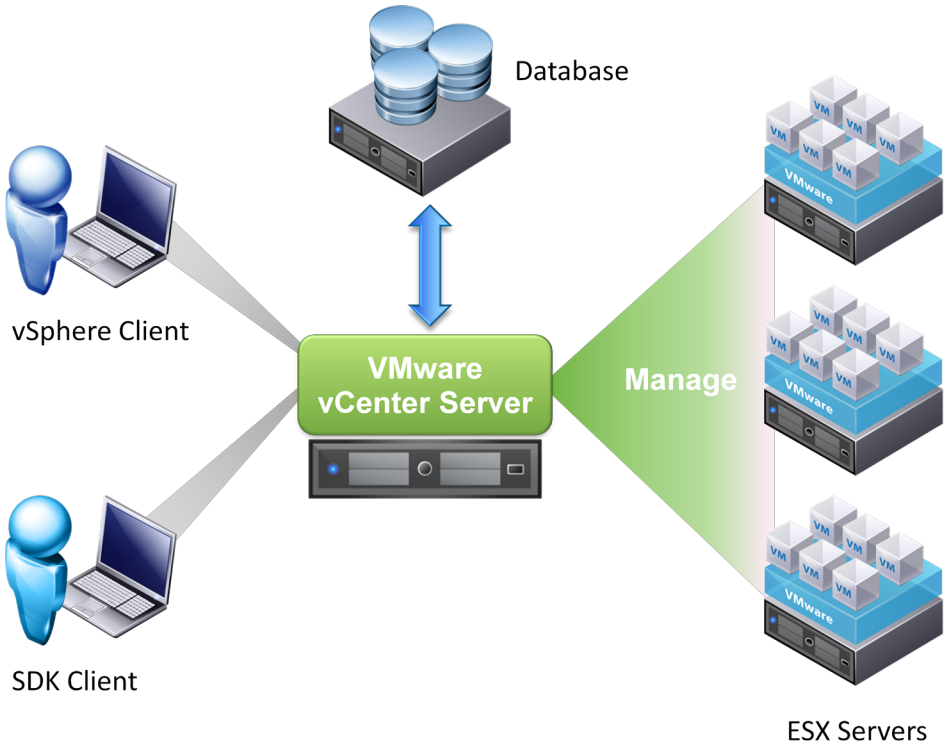
10 About the Authors and 11 Acknowledgements—presents information about the writers of this document and thanks those who helped.

2 Introduction

VMware vCenter Server provides centralized and optimized management for ESX- or ESXi-based virtual infrastructure, and offers a scalable and extensible platform for integration with a variety of other management solutions from VMware or its partners.

Figure 1 illustrates the various components of VMware vCenter and their interactions. The vCenter Server manages ESX/ESXi servers on which virtual machines (VMs) are running. The vCenter Server uses a database to store and organize inventory data. Clients are connected to vCenter Server to manage and view the inventory. There are two types of clients – vSphere Client which is a UI client and SDK Client which includes any application client, both using the vSphere Web Services SDK. vCenter Server, the database, and the clients can either run on physical machines or inside virtual machines.

Figure 1: High level architecture of VMware vCenter



This white paper focuses on vCenter Server, database, and client performance and best practices. For general vCenter Server administration information, please refer to:

- *vSphere Basic System Administration* [1]
- *ESX and vCenter Server Installation Guide* [2]

3 What's New in vCenter Server in vSphere 4.1?

3.1 Performance Improvements

Significant performance improvements have been made in vSphere 4.1 compared to vSphere 4.0. The following list highlights some of the most important performance improvements:

- Improved performance at higher vCenter Server inventory limits – up to 7 times higher operational throughput and up to 75% reduced operational latency
- Improved performance at higher cluster inventory limits – up to 3 times higher operational throughput and up to 60% reduced operational latency
- Faster vCenter Server startup – around 5 minutes for the maximum vCenter Server inventory size
- Better vSphere Client responsiveness, quicker user interaction, and faster user login
- Faster host operations and VM operations on standalone hosts – up to 60% reduction in latency
- Lower resource usage by vCenter agents by up to 40%
- Reduced VM group power on latency by up to 25%
- Faster VM recovery with HA - up to 60% reduction in total recovery time for 1.6 times more VMs
- Better load balancing with improved DRS/DPM algorithm effectiveness

To review our test results, see section 7, "Case Studies and Performance Improvements for vSphere 4.1."

3.2 Larger Inventory

The vCenter Server in vSphere 4.1 supports a larger inventory in a vSphere environment when compared with vSphere 4.0, as shown in Table 1.

Table 1. Comparison of supported inventory limits in vSphere 4.1 vs. vSphere 4.0

Limits	vSphere 4.0	vSphere 4.1
VMs per host	320	320
Hosts per cluster	32	32
VMs per cluster	1,280	3,000
Hosts per vCenter Server	300	1,000
Registered VMs per vCenter Server	4,500	15,000
Powered-On VMs per vCenter Server	3,000	10,000
Concurrent vSphere Clients	20	100
Hosts per datacenter	100	400

3.3 Concurrent Task Limits Raised

In vSphere 4.1, vCenter Server can handle a larger number of concurrent tasks at a time compared to vSphere 4.0. For vMotion tasks there are additional limits in order to prevent the network, host, or datastore from becoming a bottleneck because it can affect migration performance. Most of these limits are also increased in vSphere 4.1, as shown in Table 2.

Table 2. Limits on concurrent operations in vSphere 4.1 vs. vSphere 4.0

Limits	vSphere 4.0	vSphere 4.1
Concurrent vMotion operations per host over 1G network	2	4
Concurrent vMotion operations per host over 10G network	2	8
Concurrent vMotion operations per datastore*	4	128
Concurrent Storage vMotion operations per host	2	2
Concurrent Storage vMotion operations per datastore*	4	8
Concurrent Relocate operations per host	8	8

* This includes FC, iSCSI, and NFS, and assumes all FC and iSCSI disks are in VMFS3 format.

4 Hardware Sizing Guidelines and Software Requirements

The vCenter Server system is a physical machine or virtual machine with access to a supported database. The vCenter Server, the database, and the vSphere Client machines must meet specific hardware and software requirements.

4.1 Minimum Requirements for vCenter Server

Make sure you meet the following hardware and software requirements to run vCenter Server 4.1:

- CPU – Two 64-bit CPUs or one 64-bit dual-core processor. 2.0GHz or faster Intel or AMD processor. Processor requirements may be higher if the database runs on the same machine.
- Memory – 3GB RAM. Memory requirements might be higher if the database runs on the same machine. vCenter Server includes a service called VMware vCenter Management Web Services. This service requires at least 512MB of additional memory. The amount of memory is specified during installation and depends on the number of hosts and VMs.
- Disk storage – 3GB. Disk requirements may be higher if the database runs on the same machine.
- Networking – Gigabit connection recommended.
- Supported operating system – vCenter Server requires a 64-bit operating system. See the list in the *VMware vSphere Compatibility Matrixes* [3].
- System DSN –vCenter Server requires a 64-bit system DSN to connect to its database if the default SQL Server Express database is not used.

4.2 Minimum Requirements for vCenter Server Database

VMware supports Microsoft SQL Server, Oracle, and DB2 for the vCenter Server database. For specific versions of these supported databases, please refer to the "Database Compatibility for vCenter Server" section in the *VMware vSphere Compatibility Matrixes* [3].

Refer to your database documentation for the hardware sizing and software requirements of your database. The database hardware requirements are in addition to the vCenter Server requirements if the database and vCenter Server run on the same machine.

Additional database installation guidelines are provided in section 5.8, "vCenter Server Database."

4.3 Minimum Requirements for vSphere Client

In order to run the vSphere Client 4.1, please ensure that you satisfy the following hardware and software requirements:

- CPU – 1 CPU. 500MHz or faster Intel or AMD processor (1GHz recommended).
- Memory – 200MB RAM.
- Disk Storage – 1.5GB free disk space for a complete installation, which includes the following components:
  - Microsoft .NET 2.0
  - Microsoft .NET 3.0 SP1
  - Microsoft Visual J#
  - vSphere Client 4.1
- Networking – Gigabit connection recommended.
- Supported operating systems—see the list in the *vSphere Compatibility Matrixes* [3].

For further information, please refer to *ESX and vCenter Server Installation Guide* [2].

4.4 System Recommendations for Performance Based on Deployment Size

The number of hosts and powered-on virtual machines in your vCenter environment affects performance. The following system requirements should be used as minimum guidelines for reasonable performance. For increased performance, you can configure systems in your environment with values greater than those listed here.

Processing requirements are listed in terms of hardware CPU cores. Only physical cores are counted. In systems with hyperthreading, logical CPUs do not count as separate cores.

We define three deployment sizes according to the size of the vCenter Server inventory, as shown in Table 3. Table 4 lists the minimum hardware recommendations for the three different deployment sizes.

Table 3. Three examples of deployment sizes

Deployment Size	Hosts	Powered On VMs
Medium	50	500
Large	300	3000
Extra-large	1000	10000

Table 4. Minimum hardware recommendations for the three deployment sizes

Deployment Size	Component	Cores	Memory	Disk
Medium	vCenter Server	2	4GB	5GB
	vSphere Client	1	200MB	1.5GB
Large	vCenter Server	4	8GB	10GB
	vSphere Client	1	500MB	1.5GB
Extra-large	vCenter Server	8	16GB	10GB
	vSphere Client	1	500MB	1.5GB

NOTE The recommended disk sizes assume default log levels. If you configure more granular log levels, more disk space is required.

For more information, please refer to the *ESX and vCenter Server Installation Guide* [2].

5 Performance Best Practices, Monitoring, Tuning, and Troubleshooting

This section introduces some best practices for managing the vCenter Server, database, and clients in order to achieve the best possible performance in a vSphere environment. The topics covered in this section include an overview of vCenter Server and its inventory and performance monitoring, tuning, and troubleshooting for vCenter components.

5.1 Overview of Services and Inventory

The vCenter Server is the central management system in vSphere. It consists of two Windows services:

- **VMware vCenter Server:** Provides centralized management of hosts and virtual machines in a vSphere environment. In the Windows Task Manager, the process image name is `vpxd.exe`.
- **VMware vCenter Management Web Services:** Provides various VMware virtual management services, including query service, health service, license service, storage management service, and so on. In the Windows Task Manager, the process image name is `tomcat6.exe`.See section 5.7, "vCenter Management Web Services" for more information.

A vSphere environment contains an inventory consisting of various objects. Figure 2 shows a screenshot of a sample inventory viewed through vSphere Client. We briefly introduce each type of object here.

Datacenter An aggregation of all the different types of objects needed to do work on a virtual infrastructure: hosts, virtual machines, networks, and datastores. In this paper, we do not focus on networks and datastores. Information about networks and datastores can be found in *vSphere Basic System Administration* [1].

Host A physical machine that runs virtualization software such as ESX/ESXi for hosting virtual machines.

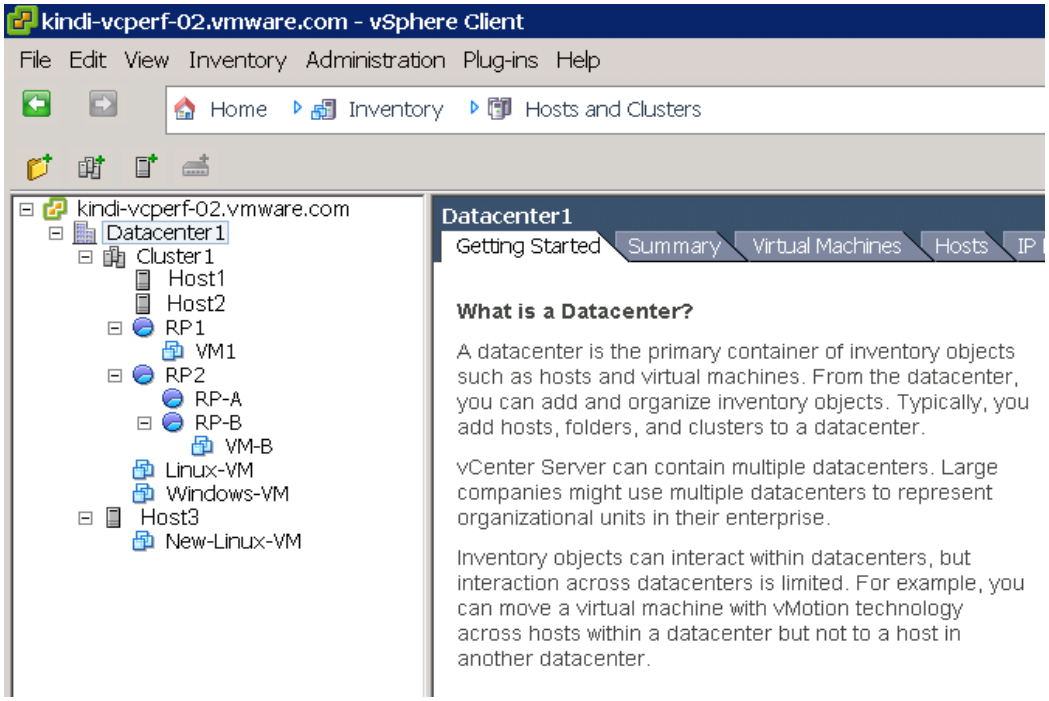
Virtual Machine A virtualized computer environment in which a guest operating system and associated application software can run.

Cluster A collection of hosts and associated virtual machines that work together as a unit. Features such as HA/FT and DRS/DPM can be enabled in a cluster.

Resource Pool Resource pools are used to compartmentalize resources. Multiple resource pools can be created and configured under a host or a cluster and then be delegated to individual services or organizations.

In Figure 2, we see that there is a datacenter *Datacenter1* which has a cluster *Cluster1* and a standalone host *Host3*. Cluster1 contains 2 hosts, *Host1* and *Host2*, and 2 levels of resource pools (actually 3 levels because the cluster itself is a root resource pool). The first level includes *RP1* and *RP2*, and the second level includes *RP-A* and *RP-B* which are child resource pools under *RP2*. There are also virtual machines in the inventory, such as *VM1* under *RP1*, *Linux-VM* and *Windows-VM* under *Cluster1*, and *New-Linux-VM* under the standalone host *Host3*. The details of these objects are discussed in the subsequent sections.

Figure 2. A sample inventory viewed through vSphere Client



5.1.1 Datacenters

We refer to *datacenter* as the logical aggregation of different types of physical resources. It does not have to be constrained to the physical location where a company can keep all its computing resourses.

You can create many datacenters in a vCenter Server inventory. This helps to organize the inventory logically, especially for large inventories. For example, if there are resources across many locations, creating one datacenter for all the resources in each location would be reasonable.

NOTE Certain operations such as vMotion and Storage vMotion cannot be performed across different datacenters. Similarly, a vNetwork Distributed Switch is limited to within a datacenter. So, you should create the datacenter hierarchy accordingly.

5.1.2 Hosts

A host is a physical machine that runs vSphere ESX/ESXi and virtual machines that share the underlying hardware resources. How vSphere ESX/ESXi multiplexes the underlying hardware resources among several VMs is beyond the scope of this paper. In this paper, we use the terms *host*, *ESX host*, and *ESXi host* interchangeably.

A host can be managed as a part of a vCenter Server cluster or it can be managed as an isolated entity in a vCenter Server datacenter. If the host is managed as an isolated entity, it is referred to as a *standalone host*. When a host doesn't need advanced management features such as HA and DRS, the host can be run as a standalone host in a datacenter. However, when a host needs these advanced management features, it should be managed as part of a vCenter Server cluster.

A host interacts with the vCenter Server through two host management agents: *hostd* and *vpxa*. Hostd is started on the host during ESX boot up. It is primarily responsible for bookkeeping of the host-level entities like VMs, datastores, networks, and so on. It is also responsible for implementing the host-level functions of the vSphere Infrastructure API. The vCenter Server dispatches host-related operations to a host over the Web using a SOAP interface. On the host, another agent called vpxa listens to these SOAP requests and dispatches them to hostd using the vCenter Server API. When a host is added to a vCenter Server inventory, vpxa is installed and started on the host. The resource consumption of hostd and vpxa can be monitored using *esxtop* [4].

Because vCenter Server communicates with an ESX host through the vSphere Infrastructure API using a SOAP interface [13], one of the key contributors to the operational latencies is the number of network hops between vCenter Server and the ESX host. If the ESX host is located multiple network hops away from the vCenter Server, the operational latencies may increase significantly. It is therefore recommended that the ESX host resides as few network hops away from the vCenter Server as possible.

5.1.2.1 Considerations for Host Sizing

In order to size a host properly when provisioning a certain number of VMs on the host, some factors need to be taken into account in addition to the VMs' own resource demands, such as the following:

- The amount of memory reserved by the virtualization software for each VM, depending on the VM's configured memory size. This is referred to as VM memory overhead reservation. This value affects the maximum number of VMs that can be powered-on on a host. Please refer to the *vSphere Resource Management Guide* [5] for the table entitled "Overhead Memory on Virtual Machines," which lists the VM overhead memory as a function of the VM's configured memory and the number of virtual CPUs.
- The amount of resources reserved by vSphere ESX/ESXi to run the host agents on the host. The maximum memory that can be reserved for the ESX host agents in vSphere 4.1 is 1GB. Administrators might wish to change the CPU and memory reservations/limits for the host agents. They can do so by choosing the following options in the host settings of the vSphere Client: **Host > Configuration > System Resource Allocation > Advanced > *Name of host agent***. The name of the host agent is *hostd* or *vpxa* depending upon which host agent the user wants to choose. However, we recommend using the defaults.

5.1.3 Clusters

A cluster is a collection of ESX or ESXi hosts and associated virtual machines. Physical resources from all the hosts in a cluster are jointly owned by the cluster and centrally managed by the vCenter Server. The vSphere administrators must create clusters in order to take advantage of cluster-level resource management features such as:

- High Availability (HA)
- Fault Tolerance (FT)
- Distributed Resource Scheduler (DRS)

- Distributed Power Management (DPM)

Both HA and DRS can be enabled when a cluster is created or by editing cluster settings from the vSphere Client. FT and DPM are sub-features of HA and DRS, respectively, and can be enabled by editing cluster settings.

In the following sections, we briefly introduce each of these resource management features and discuss best practices for achieving good performance in the managed cluster.

5.2 High Availability (HA)

VMware HA minimizes VM downtime upon failures by restarting the VMs on alternative hosts. Compared to traditional high availability solutions, it is a low-overhead and easy-to-use solution that is independent of any guest operating systems or applications.

In vSphere 4.1, HA scalability has been improved. An HA-enabled cluster can manage up to 32 hosts and 3000 VMs.

HA monitors the health of the system and detects failures at different levels, such as hosts and VMs. Starting from vSphere 4.1, HA also has the ability to monitor individual applications running inside VMs (*vSphere Availability Guide* [6]).

5.2.1 Expected Failover Behavior

When one or more hosts in an HA cluster fail, HA will restart the powered-on VMs on the alternative hosts. HA places the VMs based on the available resources and attempts to distribute them as evenly as possible. If restart priorities are set for the VMs, then HA will initiate failover for the VMs with higher priority first, before initiating failover for any of the lower priority VMs.

There are 5 hosts, called *primary nodes*, that are responsible for maintaining the cluster state, detecting failures, and initiating failover. The other hosts are called *secondary nodes*. Therefore, at any given time, a failure involving up to 4 arbitrary hosts can be tolerated. Failures of more than 4 hosts can also be tolerated if not all the primary nodes have failed. If a primary host is disconnected, put into maintenance mode, or removed from the cluster, HA will promote a secondary node to become a new primary.

5.2.2 Resource Planning and Monitoring

When HA is enabled, spare resources such as CPU and memory are needed in order to fulfill the failover requirements. The details of resource calculation are discussed in the *vSphere Availability Guide* [6].

To monitor the HA status of the cluster, you can view the summary tab of the cluster through the vSphere Client. There is a box on the upper right called **VMware HA** where you can see the basic information such as whether admission control is enabled, what the current and configured failover capacities are, and whether host, VM, and application monitoring is enabled respectively. The failover capacity is shown in a different format based on the selected admission control policy. There is also a pop-up tab—**Cluster Operational Status**—that shows you HA configuration issues, if any. If there is a configuration issue with a host, it will also show the role of the host, whether it is a primary or a secondary.

5.2.3 HA Performance Tuning

- **Host failure detection**

HA uses heartbeats to determine liveliness of the hosts. By default, heartbeats are sent every 1 second by the hosts, and if no heartbeat is received in 15 seconds from a host, the host will be pinged and declared as failed if there is no response from the host. The heartbeat interval and failure detection time can be changed through the vSphere Client as shown in Table 5. We recommend using the default values, but offer the following guidance for those who want to experiment with these settings.

If you have a lot of hosts and have observed heavy network traffic, you might consider increasing the heartbeat interval to reduce the amount of heartbeat traffic. In doing so, you should also increase the failure

detection time to avoid false failover. The heartbeat interval cannot be larger than the failure detection time, and we do not recommend setting it to a very large value, in which case missing a few heartbeats would possibly lead to undesirable failover.

For a high latency network or network with intermittent problems, you might want to increase the failure detection time to avoid false failover.

Table 5. Advanced options for tuning failure detection time

Advanced option name	Description	Default Value
das. failuredetectiontimeinterval	Heartbeat interval	1 second
das.failuredetectiontime	Waiting time before declaring a host as failed	15 seconds

NOTE You will need to disable and re-enable HA in the cluster for the changes to take effect.

- **Failover concurrency**

When multiple VMs are restarted on one host, up to 32 VMs will be powered on concurrently by default. This is to avoid resource contention on the host. This limit can be changed through the HA advanced option: `das.perHostConcurrentFailoversLimit`. Setting a larger value will allow more VMs to be restarted concurrently and might reduce the overall VM recovery time, but the average latency to recover individual VMs might increase. We recommend using the default value.

5.2.4 HA Performance Troubleshooting

- **Why do I see a transient "Communication Timeout" error in my cluster?**

This means too many state updates are happening in the cluster and HA is having a hard time catching up. You can tune the advanced option `das.sensorPollingFreq` to a larger value to eliminate this error. The default is 10 seconds in vSphere 4.1, and it can be set to a value between 1 and 30 seconds.

The `das.sensorPollingFreq` option controls the HA polling interval. HA polls the system periodically to update the cluster state with such information as how many VMs are powered on, and so on. The polling interval was 1 second in vSphere 4.0. A smaller value leads to faster VM power on, and a larger value leads to better scalability if a lot of concurrent power operations need to be performed in a large cluster.

- **What if specific primary hosts are not performing well?**

Some hosts might not be as robust as other hosts; therefore, you might not want them to be the primary nodes. You can put these hosts into maintenance mode before enabling HA, and then they will not be chosen as the primaries. Five of the hosts that are not in maintenance mode will become the primaries. After you enable HA, take the hosts out of maintenance mode—these hosts will become secondaries. (Note that these secondaries could be promoted to primaries if and when the primaries are disconnected, removed from the cluster, or put into maintenance mode later.)

- **Why can I power on fewer VMs in my cluster when HA is enabled?**

As mentioned before, when HA is enabled, spare resources are reserved in order to fulfill the failover requirements. If you cannot power on a VM in an HA cluster, then it is likely because powering on this VM will violate the HA admission control policy and result in insufficient failover capacity in the cluster. HA admission control is done based on the policy you select, the cluster capacity, and the VMs' resource settings. There are advanced options that you can set to adjust the number of VMs that can be powered on in an HA cluster. Please see *vSphere Availability Guide* [6] for more details.

5.3 Fault Tolerance (FT)

VMware FT provides continuous high availability for virtual machines by creating and maintaining a secondary VM that is identical to the primary VM on a different host. FT uses VMware vLockstep to capture the inputs and events that occur in the primary VM and sends them to the secondary VM. In the case of a primary failure, the secondary will replace the primary in a seamless manner.

HA is a prerequisite for enabling FT. FT works together with HA to achieve a higher level of availability as well as data protection, so it is especially useful for mission-critical VMs that cannot afford downtime. The maximum number of FT VMs per host is 4 in vSphere 4.1. Currently, FT supports only uniprocessor virtual machines.

FT has minimal impact on application performance under most circumstances. Please see *VMware vSphere 4 Fault Tolerance: Architecture and Performance* [7] for details.

5.3.1 Expected Failover Behavior for FT VMs

If an FT primary VM or its host fails, the secondary VM will become the primary, and a new secondary will be started on an alternative host by HA. If an FT secondary VM or its host fails, then a new secondary will be started on an alternative host by HA.

5.3.2 Resource Planning and Monitoring

Additional resources need to be taken into account when FT is enabled, such as spare CPU cycles and additional network bandwidth. Once FT is enabled, you can view the FT-related information using the vSphere Client. The information is shown on the primary VM's summary tab, including *Fault Tolerance Status*, *Secondary Location*, *Secondary CPU and Memory usage*, *vLockstep Interval*, and *Log Bandwidth*.

Please see *vSphere Availability Guide* [6] for detailed resource planning and monitoring information.

5.4 Distributed Resource Scheduler (DRS)

VMware DRS is a cluster-level resource management feature running inside vCenter Server. When enabled, DRS continuously monitors the CPU and memory usage of individual hosts and VMs and performs the following two main tasks:

- **Initial VM placement** – When a new VM is powered on, DRS determines the optimal placement for the VM based on the VM's resource requirements, the available capacity, as well as placement constraints due to hardware compatibility or user-specified affinity (or anti-affinity) rules.
- **Load balancing** – DRS runs every 5 minutes to ensure that per-host resource utilization levels are below a threshold and are distributed evenly across the cluster; if not, DRS recommends VM migrations within the cluster to alleviate resource contention on the hosts and improve load balance in the cluster.

In addition, DRS also makes VM migration recommendations to evacuate VMs from a host entering maintenance or standby mode, or to satisfy constraints due to host resource reservation or user-specified rules. vSphere 4.1 includes DRS rules to specify the VM/Host affinity (or anti-affinity), in addition to the previously-supported VM/VM affinity (or anti-affinity) rules. The new rules are used to specify the relationship between a group of VMs and a group of hosts—whether they should be located together (affinity) or separately (anti-affinity). These new rules are especially useful for clusters with software licensing restrictions or specific availability zone requirements. More details can be found in the *vSphere Resource Management Guide* [5].

A DRS-enabled cluster is often referred to as a DRS cluster. In vSphere 4.1, a DRS cluster can manage up to 32 hosts and 3000 VMs.

5.4.1 Resource Pools

Resource pools are useful for allocating portions of the cluster-level resources to individual organizations or services. A resource pool hierarchy can be defined to reflect the natural hierarchy in an organization. By

default, the DRS cluster itself is the root resource pool for all the user-created resource pools in the cluster. vCenter Server 4.1 supports up to 512 resource pools in a DRS cluster and the resource pools can be up to 8 levels deep under the root resource pool.

5.4.2 Resource Allocation Settings

An administrator can specify resource allocation settings (reservations, limits, shares) for a VM as well as a resource pool. The DRS algorithm computes CPU and memory entitlements for a VM based on the resource allocation settings of the VM itself and of its parent resource pool, as well as cluster capacity. It also estimates the resource demand of a VM based on resource allocation data from the host.

For a VM or a resource pool containing mission-critical applications, administrators can specify a CPU or memory reservation to guarantee the level of resource allocation. If the overall cluster capacity might not meet the need of all VMs during peak hours, you can assign relatively higher shares to more important VMs or resource pools to avoid or minimize performance degradation in their hosted applications.

Please refer to *Resource Management with VMware DRS* [8] for more information on resource pools and resource settings.

5.4.3 Better Integration Between HA and DRS/DPM

When both HA and DRS are enabled, upon host failures, the first priority is to restart the VMs as soon as possible; therefore, HA will place the VMs on the alternative hosts. After all the VMs have been restarted, different hosts in the cluster might have different levels of utilization resulting in a cluster load imbalance. At that time, DRS will perform load balancing within the cluster.

vSphere 4.1 includes better integration between HA and DRS/DPM when HA admission control is disabled. Admission control is a policy used by VMware HA to ensure failover capacity within a cluster. When VMware HA is enabled, admission control is enabled by default. You can choose to disable admission control in HA, in which case VMs will be allowed to power on even when they violate the availability constraints in the cluster.

In vSphere 4.1, if a VM cannot be restarted by the HA agent and if the host is connected to the vCenter Server, it asks DRS/DPM to try to make room for the VM by recommending vMotions and/or host power-on.

5.4.4 Better Integration Between FT and DRS

Starting from vSphere 4.1, DRS is able to place and load balance FT VMs. When Enhanced vMotion Compatibility (EVC) is enabled, DRS can place the primary VM and its secondary on separate hosts at power on and perform load balancing afterwards. When EVC is disabled, DRS will not move FT VMs (neither primary nor secondary) for load balancing purposes, but will still be able to place the secondary VM at power on. The primary in this case would be powered on using its registered host. More information about EVC can be found in *vSphere Basic System Administration* [1].

5.4.5 DRS Performance Monitoring

The vSphere Client provides a **Resource Allocation** tab for each cluster. For either **CPU** or **Memory**, the **Total Capacity** represents the total capacity in the cluster that can be allocated to the VMs. Note that these numbers are lower than the **Total CPU Resources** and **Total Memory** shown under the **Summary** tab of a cluster, because the latter numbers are the total raw CPU and memory capacities in the cluster. The differences between these two sets of numbers are due to CPU and memory resources that are reserved for the virtualization software.

When DRS is enabled, the **VMware DRS** panel under the cluster summary tab shows the **Target host load standard deviation** and the **Current host load standard deviation**. We refer to these two metrics as the current and target *cluster imbalance*. If the current imbalance is below the target imbalance, the cluster is shown as **Load balanced**; otherwise, it's shown as **Load imbalanced.** The same panel also has a link to a

pop-up window containing the **Resource Distribution Chart**. This chart shows the per-host CPU and memory utilization across the cluster. For each host, each colored-block represents the resource demand of one or multiple VMs. For CPU resource, the color represents how satisfied the VM is in terms of the percent of the entitled resource it is receiving. For example, green indicates satisfaction and red indicates resource contention on the host.

NOTE For each host, the CPU and memory utilization shown in **Resource Distribution Chart** might be different from the **CPU usage** and **Memory usage** shown in the **Resources** panel under the host **Summary** tab. This is because the former represents only the sum of the VMs' resource demands, whereas the latter also includes the resources used by the virtualization software.

5.4.6 DRS Performance Tuning

The vSphere Client offers a tunable parameter, the **Migration threshold**, from the cluster settings menu that allows the administrator to control the aggressiveness of the DRS algorithm. This parameter adjusts the value of the target imbalance used by the DRS algorithm. The default setting of the migration threshold is 3, representing a medium level of aggressiveness.

In addition to the migration threshold, DRS offers a number of advanced options that allow you further control over the algorithm performance. Table 6 highlights some of these options and explains what they are used for. Improvements were made in the DRS algorithm in vSphere 4.1 to make MinGoodness adaptive to the cluster inventory size and to make MaxMovesPerHost adaptive to the maximum number of concurrent vMotions per host (see Table 2 in Section 3) and the average migration time observed from previous migrations. These improvements make DRS less conservative compared to that in vSphere 4.0 and allow DRS to reach a steady state more quickly when there is significant load imbalance in the cluster.

Table 6. DRS advanced options for performance tuning

Advanced option name	Description	Default Value	Most Aggressive Value
CostBenefit	Whether to take migration cost into account	1	0 (No cost-benefit analysis)
MinImbalance	Used to compute target imbalance	50	0
MinGoodness	Minimum improvement in cluster imbalance required for each move	Adaptive	0 (All moves are considered)
MaxMovesPerHost	Maximum number of moves per host recommended per invocation	Adaptive	0 (No limit)

NOTE It is recommended that the above DRS advanced options **be left at their default values**, unless there is a strong reason to change them; for example, there is severe resource contention on some hosts and there is idle capacity on other hosts in the cluster.

5.4.7 DRS Performance Troubleshooting

- **Why is a cluster shown as "Load imbalanced" and DRS is not recommending any VM migrations?**

This can be due to many reasons. For instance, VM migrations may be filtered out due to constraints and rules in the cluster that make these moves unacceptable. In another instance, maybe the costs of previous migrations exceed the expected benefits from the potential moves. You can adjust the migration threshold to a more aggressive level, if you see better cluster load balance as a priority over other considerations. On the other hand, as long as all the VMs are receiving 100% of their entitled resources, it may be acceptable to have a slightly imbalanced cluster.

In addition, the VMware community page, *Scripts for Proactive DRS* [9], provides a set of scripts for more advanced DRS users to conduct more proactive load balancing in a cluster.

- **What if DRS recommends too many VM migrations even though all the hosts are lightly utilized?**

You can lower the migration threshold to make the DRS algorithm more conservative so that fewer VMs are migrated in the cluster. If the most conservative threshold is chosen, DRS will only recommend moves 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.

5.4.8 DRS Best Practices

- **Leave room for vMotion-related CPU reservation**

In addition to the VM-level CPU reservation made by the users, ESX also reserves a certain amount of CPU capacity for carrying out a vMotion task for a VM on both the source and the destination hosts of the vMotion. In vSphere 4.1, this CPU reservation is 30% of a processor core for a 1Gb network interface and 100% of a core for a 10Gb network interface. Such amount of CPU resource is needed to take full advantage of the network bandwidth in order to minimize migration time. It is therefore recommended that an administrator always leave some CPU capacity in a cluster as unreserved. This helps ensure that vMotion tasks get the resources needed and are completed as quickly as possible.

- **Keep a sufficient number of drmdump files**

The drmdump files produced by DRS can be very useful in diagnosing potential DRS performance problems during a support call. The number of drmdump files stored is limited by the maximum size of the drmdump directory, for which the default is set at 10MB. For large clusters, it is possible that not enough drmdump files are kept before being rolled over. In that case, you can use the DRS advanced option, **DumpSpace** (in MB), to increase the maximum size of the drmdump directory.

5.5 Distributed Power Management (DPM)

VMware Distributed Power Management or DPM is a power management feature within DRS that can be enabled to help reduce power consumption of a DRS cluster. DPM can dynamically consolidate VMs onto fewer ESX hosts in a cluster during periods of low cluster utilization and then power off the un-needed hosts. When workload demands increase, these ESX hosts are powered back on and virtual machines are redistributed to them to ensure good application performance. This feature is especially effective for datacenters with high peak-to-mean ratio in the workloads or with cyclic demand patterns such as time-of-day or day-of-week effects.

DPM uses three types of host-waking protocols, Wake-on-LAN (WoL), IPMI, or iLO, to power on or off hosts when needed. Users are expected to test each participating host for entering and exiting standby mode with whichever protocol was chosen for that host before enabling DPM for a cluster. Please refer to *VMware Distributed Power Management Concepts and Use* [10] for more details.

DPM is complementary to the host power management (HPM) policies that use both the processor P-states and C-states to reduce host power consumption when the workloads running on a host do not require full CPU capacity or when some CPUs are idle. Using DPM and HPM together can offer greater power savings than when either solution is used alone.

5.5.1 DPM-Related Performance Improvements

DPM-related performance improvements in vSphere 4.1 include the following:

- In vSphere 4.1, new ability is added within vCenter Server to schedule tasks to enable or disable DPM. When DPM is disabled, all hosts in the cluster are powered on. This can be used by customers who are concerned about DPM running during a critical time of a day or by those concerned about some hosts being left in standby for too long.
- The computation of the cluster imbalance metric is improved by excluding standby hosts so that DRS does not act more aggressively than intended when there are hosts in the standby mode.

5.5.2 DPM Performance Tuning

The aggressiveness of the DPM algorithm can be tuned by adjusting the **DPM Threshold** in the cluster settings menu. This parameter controls how much per-host resource utilization can be outside the target utilization range before DPM makes host power-on/off recommendations. The default setting for the threshold is 3 (medium aggressiveness). In addition, the DPM advanced options **MinPoweredOnCpuCapacity** and **MinPoweredOnMemCapacity** allow the administrator to specify the minimum amount of CPU (in MHz) or memory (in MB) capacity to be kept on in the cluster. These parameters are listed in Table 7.

Table 7. DPM advanced options for performance tuning

Advanced option name	Default Value	Maximum Value
MinPoweredOnCpuCapacity	1	CPU capacity of the cluster
MinPoweredOnMemCapacity	1	Memory capacity of the cluster

5.5.3 DPM Performance Troubleshooting

The following are some common DPM-related performance problems that might be encountered.

- **What if DPM does not recommend host power offs in spite of low utilization in the cluster?**

In a cluster with VMware HA enabled, DPM maintains excess powered-on capacity to ensure that sufficient idle resources are available during an HA failover. As a result, DPM might not power off some hosts even though the cluster may appear to be lightly utilized. If there seems to be more idle resource than the HA failover capacity, and lower power consumption is a top priority for a certain datacenter, then the administrator can either set the DPM threshold to a more aggressive setting, or adjust DRS advanced options (as mentioned in the previous DRS section) to make DRS less conservative so that it is more likely to recommend moves.

- **What if not enough hosts are kept powered on to accommodate load spikes?**

For some datacenters that often have unexpected spikes in VM resource demands, the administrator can use either **MinPoweredOnCpuCapacity** or **MinPoweredOnMemCapacity** to ensure that a minimum number of hosts are always on even during a period of low cluster utilization.

DPM can be disabled on individual hosts that are running mission-critical VMs, and the new VM/Host affinity rules in vSphere 4.1 can be used to ensure that these VMs are not migrated away from these hosts.

In addition, the VMware community page, *Scripts for Proactive DPM* [11], provides a set of scripts for more advanced DPM users to conduct more proactive power management.

Refer to the *VMware Distributed Power Management Concepts and Use* [10] for more details on DPM.

5.6 vCenter Server Tasks

This subsection discusses some general guidelines regarding vCenter Server tasks. In vSphere 4.1, vCenter Server can handle up to 500 concurrent tasks at a time *depending* on what tasks are being performed. Additional tasks are queued until ongoing tasks are completed.

5.6.1 Task Timeout Settings

Tasks in vCenter Server will time out if there is no activity on the link between the vCenter Server and the hosts for a given amount of time. By default, most operations time out after 60 seconds, while some long operations like Enter Maintenance Mode and Add Host time out after 120 seconds. These settings can be changed using the vSphere Client through **Administration > vCenter Server Settings > Timeout Settings**.

5.6.2 Task Lifetime

Once a task is complete in vCenter Server, the task shows up in the recent tasks pane for a default lifetime of 10 minutes after which tasks can be found in the **Tasks & Events** tab. The task list is capped at 200 tasks in order to allow for better scalability. These default values work well. However, you can reduce the task lifetime by adding the following tag to the `vpxd.cfg` file.

```
<task>
<completedLifetime>600<completedLifetime>
</task>
```

Indicate the value of `completedLifetime` in seconds.

You can also change the length of the task history list in the `vpxd.cfg` file:

```
<task>
<completedMaxEntries>200</completedMaxEntries>
</task>
```

The `vpxd.cfg` file is located at:

```
<Program Data Folder>\VMware\VMware VirtualCenter
```

5.7 vCenter Management Web Services

With vSphere 4.0, a major architectural change has been introduced to improve scalability and to provide various functionalities. An open source Java Servlet implementation of Apache Tomcat is being deployed to run various services. In vSphere 4.1, we have extended these Web Services in Apache Tomcat, including:

- **Query Service:** Supports the Search functionality.
- **Health Service:** Supports the vCenter Service Status.
- **Common Information Model Service:** Supports the Hardware Status for ESX hosts.
- **License Service:** Supports Licensing Reporting Manager.
- **Storage Management Service:** Supports Storage Views.
- **Stats Retrieval Service:** Supports Performance Charts > Overview.

5.7.1 Tomcat Services

You can configure or monitor Tomcat through **Start > All Programs > VMware > VMware Tomcat > Configure/Monitor Tomcat**.

- **Logging:** Log level and log path can be set in the Logging tab.
- **Java:** VMware vCenter Management Web Services consists of various services running within Apache Tomcat, which is a Java Servlet. Thus, all the known tuning options for Java can be used to configure Tomcat in the Java tab. However, we recommend using the default options. One exception is the memory pool for Java Virtual Machine, described below.

5.7.2 Max Heap Size for Java Virtual Machine

Based on the inventory size that will be deployed in vCenter Server, you should choose the appropriate options for the Java Virtual Machine (JVM) Max Memory, during installation:

- Small (less than 100 hosts) 1024 MB
- Medium (100-400 hosts) 2048 MB
- Large (more than 400 hosts) 4096 MB

These values are set in the **Maximum memory pool** field in the **Java** tab of the **Configure Tomcat** window. Be aware that the JVM has its own memory overhead. For example, if a user chooses to assign 4096MB to

the maximum JVM heap size, then the `tomcat6.exe` process can grow up to 4.5GB, since the memory overhead to run the JVM is around 300-400MB.

In Windows Task Manager, the `tomcat6.exe` process can reach its maximum memory right after startup. However, this does not mean that there is a memory contention. JVM manages its own memory. The actively used memory is much lower than this assigned memory in Task Manager.

5.8 vCenter Server Database

VMware vCenter Server uses a database to store information about the state of a VMware vSphere environment, and historical performance statistics. Performance statistics and their associated stored procedure operations are among the largest and the most resource-intensive components of the vCenter database.

The data in the vCenter database can be divided into three broad categories:

1 **Inventory**—includes host and virtual machine configurations, resources and virtual machine inventory, user permissions, and roles.

2 **Alarms, events, and tasks**—Alarms are notifications that are triggered in response to selected events, conditions, and states that occur with objects in the inventory. Tasks are system activities that do not complete immediately.

3 **Performance statistics**—includes the performance and resource utilization statistics for datacenter entities, such as hosts, clusters, resource pools, and virtual machines. vCenter provides configurable settings to control how performance statistics are collected across the vSphere environment.

Performance statistics can take up to 90 percent of the vCenter database size, and hence are a primary factor in the performance and scalability of the vCenter Server database.

5.8.1 Database Sizing Guidelines

Sizing your vCenter Server database correctly is crucial to the performance of your vCenter installation. Depending on the number of entities you are managing with your vCenter installation, and the amount of performance statistics you have set vCenter to collect, the size of your database could be an important factor in your vCenter Server deployment.

vCenter Server comes bundled with a Microsoft SQL Server Express edition database.

With the SQL Server Express edition database, vCenter can support up to 5 VMware ESX hosts and 50 virtual machines in the inventory.

Sizing calculators provided by VMware can be used to estimate the size of the vCenter Server database. Links to sizing calculators for Microsoft SQL Server and Oracle are listed in section 9, "References."

The database sizing calculators [20] [21] provide an idea about the disk size that the vCenter Server database is expected to consume. To determine if you will need to run your vCenter Server database on a separate machine, you will also need to understand its resource usage.

The amount of resources your vCenter Server database consumes depends on the number of entities, including virtual machines and ESX hosts that you plan to manage and the amount of performance statistics you plan to collect in your vCenter Server installation. Please refer to the *VMware vCenter 4.0 Database Performance for MS SQL Server 2008* [12] to get an idea of the resource usage for a SQL Server 2008 system.

If the vCenter Server database is running on a different machine than the vCenter Server, the network latency between the machines is something that can adversely impact the performance of your vCenter Server installation.

5.8.2 Database Performance Tuning and Best Practices

- **Database file placement on disk**

For any database server, there are two types of files that generate I/O traffic in your system—data files and log files. Data files generally cause a lot of random reads, whereas log files generate a lot of sequential writes from/to the physical disk. Since these two loads are very different, it is considered a best practice for performance to separate out these two types of files onto drives backed by different physical disks.

- **Performance statistics collection in vCenter Server**

vCenter collects real-time statistics from the host agent running on each ESX host it manages. Every 30 minutes, vCenter inserts the real-time statistics into its database as historical statistics.

You can set the amount of performance statistics collected in your vCenter Server databse. One way to do that is by setting the statistics level to be between 1 and 4. Choosing the right statistics level for your vCenter database is crucial to its performance. You can also selectively turn off rollups for particular collection levels. This significantly reduces the size of the vCenter database and speeds up the processing of rollups. However, this approach might provide fewer performance statistics for historical data.

VMware recommends that you configure statistics at level 1. For more details and guidelines for setting the statistics level, please refer the *VMware vCenter 4.0 Database Performance for MS SQL Server 2008* [12].

- **Database connection pool for vCenter Server**

vCenter Server starts up with a database connection pool of 50 threads. This pool is then dynamically sized based on the interaction between the vCenter Server and the database. It grows adaptively as needed, based on the vCenter Server workload, and does not require any modification. However, if a heavy workload is expected on the vCenter Server, this pool can be increased up to 128. Note that this might result in increased memory consumption by vCenter Server and an increased vCenter Server startup time.

- Performance tuning for Oracle

Apart from the general database performance tuning guidelines mentioned above, there are a few Oracle specific tips that can greatly enhance the performance of vCenter Server.

  - Allocate sufficient memory for the Oracle process. When using Automatic Memory Management in Oracle, check the **MEMORY_TARGET** and **MEMORY_MAX_TARGET** values. **MEMORY_TARGET** specifies the amount of memory that Oracle can utilize, and **MEMORY_MAX_TARGET** is the maximum value that can be set for **MEMORY_TARGET**.
  - In vCenter Server 4.1, add a Unique Index in the **VPXV_DEVICE_COUNTER** table for better performance.

```
create unique index VPXI_DEVICE_COUNT_IDX on
VPXV_DEVICE_COUNTER(entity_id, device_name, stat_id);
```

  - Set appropriate **PROCESSES** and **SESSIONS** parameters. Oracle creates a new server process for every new connection that is made to it. The number of connections an application can make to the Oracle instance then depends on how many processes Oracle can create. **PROCESSES** and **SESSIONS** together determine how many processes Oracle can create at a time. VMware recommends a value of 800 for both these parameters in large vSphere environments.

For more information on vCenter Server database performance, please refer the *VMware vCenter 4.0 Database Performance for MS SQL Server 2008* [12].

5.9 Clients

There are two primary ways to manage VMware vSphere environments.

- vSphere Client is the principle application providing a user interface to manage VMware vSphere environments. It is distributed with the vCenter Server.
- VMware vSphere Web Services SDK is a software development kit with which users can write scripted client applications to access, manage and monitor their vSphere environments. We refer to all such clients as SDK clients in this paper.

vSphere Clients and SDK Clients can be used to:

- Monitor the vSphere infrastructure, track the current state of the system, and troubleshoot.
- Manage the vSphere infrastructure and initiate various tasks including configuration, provisioning, and power operations.

5.9.1 vSphere Clients

The vSphere Client is the interactive client application to manage VMware vSphere environments. The vSphere Client is improved in vSphere 4.1 to have better responsiveness than in vSphere 4.0. The vSphere Client in version 4.1 also provides easier navigation and quicker user interaction through an improved user interface.

The vSphere Client provides various functional components to monitor and manage vSphere infrastructure:

- **Inventory**: To view the entities such as datacenters, resource pools, clusters, networks, datastores, templates, hosts, and virtual machines. The inventory options include Search, Hosts and Clusters, VMs and Templates, Datastores, and Networking.
- **Administration**: To configure and monitor the state of the hosts and vCenter Server systems. The administration options include Roles, Sessions, Licensing, System Logs, vCenter Server Settings, vCenter Service Status, and Licensing Reporting Manager.
- **Management**: To monitor and manage the vSphere inventory objects. The management options include Scheduled Tasks, Events, Maps, Host Profiles, and Customization Specification Manager.

5.9.1.1 Supported Limits for vSphere Clients

The vCenter Server informs all the connected vSphere Clients about all the inventory and system changes as fast as it can. When the hardware sizing guidelines in section 4 are adhered to, vCenter Server 4.1 can support up to 100 concurrent vSphere Clients. A higher number of vSphere Clients has two consequences:

- vCenter Server informs all the connected vSphere Clients about the updates in the inventory state. Thus, for each additional vSphere Client, vCenter Server utilizes more resources in terms of CPU and memory. Refer to section 7.5, "Better vSphere Client Scaling" for details.
- vCenter Server informs all the connected vSphere Clients about the updates in the inventory state and it does this in parallel. However, when the number of connected vSphere Clients is large, serialization occurs in the CPU scheduler or in the network stack. As a result, some of those vSphere Clients will receive updates sooner, while other clients receive them later.

5.9.1.2 Best Practices for vSphere Clients

- Disconnect vSphere Clients that are not necessary, instead of letting them remain connected and sit idle. This considerably reduces resource utilization of vCenter Server.
- If many vSphere Client instances are running on a machine, be aware of the hardware resource restrictions and monitor the resource usage on the system. Refer to Section 4 for the minimum hardware requirements for a vSphere Client.

- Search instead of navigate. The most common operation in the vSphere Client is selecting an entity in order to check some information on the summary page or to initiate a task. Users can navigate in the inventory panel to select an entity. However, the best practice is to use the search feature. Inventory search provides an elegant way to reach the entire inventory including virtual machines, hosts, datastores, and networks from anywhere within the vCenter, especially for large inventories. You can perform advanced searches by specifying multiple criteria in **Inventory Search View**. You can also perform quick searches for some keywords using the search field in the top right corner of the vSphere Client. Search functionality also reduces the resource utilization on vCenter Server compared to navigating in the inventory panel.

5.9.1.3 Performance Tuning for vSphere Clients

- **Hiding the virtual machines in inventory view:** In the inventory view, it may be difficult to navigate and manage entities except virtual machines if the inventory is large. The user might choose to temporarily hide the display of virtual machines in vSphere Client by going to **Edit > Client Settings > Lists** and simplify the hierarchical view.
- **Viewing large maps:** You should be cautious viewing maps with a large number of entities. For example, if there are thousands of virtual machines in a datacenter, then the map might be difficult to read and it may take a considerable time to appear. As a precaution, a parameter **Maximum requested topology entities** has been defined in **Client Settings > Maps**, and it is set to 300 by default. This helps vCenter Server compute large maps. You can still choose to load the topology map with a large number of entities, and optionally increase the maximum number of entities from the **Client Settings** page.
- **Maximum number of pop-out console windows:** You can open a maximum of 25 pop-out console windows within a vSphere Client session, by default. If a powerful machine is being used, you may choose to increase this value in **Client Settings > General**.
- **Client-Server Time-Out:** If there is no response from the vCenter Server after 30 seconds, by default, then vSphere Client actions time out. A custom value can be set in **Client Settings > General**. This option can be utilized for a scenario of frequent, intermittent network problems.
- **Maximum number of items in Tasks & Events:** The maximum number of items listed on the **Tasks & Events** tab can be changed in **Client Settings > Lists**. This can be especially useful for scenarios with higher rates of changes in the vSphere infrastructure.
- **Performance charts default view:** The vSphere Client presents the **Overview** performance charts by default. The default view can be changed to **Advanced** in **Client Settings > General**.

Advanced users can refer to *Customizing the vSphere Client* [16] for more information.

5.9.2 SDK Clients

Using the VMware vSphere Web Services SDK, users can develop applications that target the VMware vSphere Application Programming Interface (API).

To learn more about the VMware vSphere API and supported SDK libraries, refer to the *VMware vSphere Web Services SDK Documentation* [14].

VMware vSphere Web Services SDK client applications (SDK clients) are commonly written to perform operations such as monitoring the vSphere inventory, performing management tasks on inventory items, and tracking the progress of tasks. Such SDK client operations tend to have a performance impact on the vCenter Server to which the clients are connected. As a result, the number of SDK clients connected to a vCenter Server and the specific operations that each client performs can significantly affect the performance of the vCenter Server.

5.9.2.1 Best Practices for SDK Clients

Here are some best practices that can be followed while writing SDK clients to manage the vSphere environment more efficiently.

- **Retrieving and Monitoring Object Properties**

There are two ways to retrieve object properties from vCenter Server to monitor for changes—the *PropertyCollector* and *ViewManager* interfaces.

Using a PropertyCollector object, a user can create a PropertyFilter in the client, specifying the objects and their properties that need to be monitored. vCenter Server then creates and maintains a filter object based on these specifications and updates the client whenever a property specified in the filter changes state.

Using ViewManager, the user can create customized objects that represent subsets, or Views, of specific instances of selected object types on the server. These Views can then be used to monitor object properties.

There are three types of Views:

- **Inventory View** covers the entire inventory
- **List View** covers a collection of specified managed objects
- **Container View** covers the managed objects in a container

The following factors can affect the performance of PropertyCollector for any given session:

- Number of objects
- Number of properties
- Density of property data (composite, nested data objects)
- Frequency of changes to the objects and properties on the server
- Depth of traversal (number of properties traversed)

In addition, a vCenter Server is affected by the number of PropertyCollector instances and the number of filters each is supporting across all sessions on the server.

Views are easier to maintain in the vCenter Server, and hence are more efficient, even with multiple client sessions connecting to the server. Consider using the ViewManager and View objects to minimize both the overhead on PropertyCollector and the amount of network traffic for your client application.

- **Monitoring Tasks**

Another important and very common operation that SDK clients need to do is to monitor the progress of previously issued management tasks to the vCenter Server.

Any API method that ends with "_Task" returns immediately on execution, with a Task object. This Task object can then be used to monitor the progress of the task in question, for example, by creating a View on the Task object.

Task objects are short-lived, and hence need to be used as soon as possible. If the returned Task object cannot be used immediately, try using the *TaskHistoryCollector*, which can also provide information about completed tasks.

5.9.2.2 Performance Troubleshooting for SDK Clients

Following best programming practices while writing your client application is crucial to good performance for not just the client application, but the vSphere installation too. You can refer to *Programming Code Samples from VMware's Community Page* [18] for examples of good programming practices in every supported client platform.

If you happen to find your client application being slow or causing slowness in your vCenter Server, one way to debug is to monitor SOAP messages that are exchanged between your client and your vCenter Server. An easy way of doing this is to enable HTTP, and use *tcpmon* to capture packets exchanged between the client and the server.

You can also examine the vCenter Server logs, either in *verbose* or in *trivia* mode, to look for clues.

For more information, refer to the *VMware APIs and SDKs Documentation* [13].

6 Performance Monitoring and Troubleshooting Tools

As we have discussed earlier, vCenter Server interacts with several components: Tomcat Web Services, vCenter Server database, ESX hosts, vSphere Clients, and several other SDK clients when attached. The responsiveness (latency and throughput) of client-driven operations is a complex function of how efficiently the vCenter Server communicates with these individual components and the performance of these individual components. Multiple factors can influence the vCenter Server performance. These are:

- Resource constraints on the vCenter Server machine
- Sub-optimal database performance
- Sub-optimal SDK clients performance
- High network latency between vCenter Server and hosts
- High network latency between vCenter Server and database

In this section, we familiarize the reader with a variety of tools available to monitor the performance of vCenter Server and the individual components that the vCenter Server interacts with. When used effectively, these tools can help root-cause performance bottlenecks in the vCenter Server ecosystem.

6.1 Monitoring vCenter Server and vSphere Client

The vCenter Server runs as a Microsoft Windows application on modern Microsoft Windows operating systems. Please refer to section 4 for software requirements to run vCenter Server. There are multiple tools available in Microsoft Windows to monitor vCenter Server performance. Two of the most commonly used tools are:

- Microsoft Windows Task Manager
- Microsoft Windows perfmon toolkit

The first step to monitor the resource usage on the vCenter Server is to launch the Windows Task Manager and visualize the CPU and memory usage in the Performance tab. If the CPU or memory utilization is more than the expected values for your inventory sizes (see section 4), then you can open the **Process** tab and sort the processes according to their CPU and memory usage. This will give you some idea about the processes that are utilizing more than expected resources.

Microsoft Windows *perfmon* is a very useful tool for monitoring and plotting many performance metrics for multiple processes over a given sampling period. There are multiple online tutorials available on how to use perfmon, which make this tool easy to learn. This tool lets a user monitor the CPU, Memory, Storage and Network resources by individual processes over a sampling interval. The user can pre-configure the set of metrics they want to monitor for a given process and monitor them while the application is running. The description for these metrics can be seen in detail in the perfmon tool by enabling the **Show Description** check box. The internal details of how perfmon works are beyond the scope of this paper.

6.1.1 Using Perfmon Counters

Table 8 below lists the main perfmon performance counters that can be used to monitor vCenter Server and vSphere Client performance. Each column represents a distinct perfmon performance object and all the values under the column represent the performance counters that can be monitored for that performance object. For example, we recommend monitoring all counters associated with the **Process** performance object in perfmon for processes named "_Total", "vpxd", "tomcat" and "VpxClient". The counts associated with the process "_Total" refer to the sum of the counts of all the processes that are running on the system. It helps users understand the overall resource utilization of the system.

Table 8. Perfmon counters for monitoring vCenter Server and vSphere Client. Every column is a distinct perfmon performance object and all values under the column are performance counters for that object.

Process (all counters)	System	Network Interface	Physical Disk
_Total	Threads	Bytes Received/second	Disk Bytes/second
vpxd	Context Switches/second	Bytes Sent/second	Disk Read Bytes/second
tomcat	Processor Queue Length	Bytes Total/second	Disk Reads/second
VpxClient		Packets Received/second	Disk Write Bytes/second
		Packets Sent/second	Disk Writes/second
		Packets Total/second	Current Disk Queue Length
			Avg. Disk Seconds/Read
			Avg. Disk Seconds/Write

6.1.2 Using Log Files for Troubleshooting

In case of any problems, you might want to check the log files for the various vCenter components. Here's where you can find the different log files.

6.1.2.1 vCenter Server Logs

Process logs: This is the main vCenter Server log file. The vCenter Server runs as a windows process named `vpxd.exe`. The log file contains information about important activities that the vpxd process is involved in, and is usually a very good starting point for debugging vCenter Server issues.

```
<Program Data Folder>\VMware\VMware VirtualCenter\Logs\vpxd-X.log
```

Profiler logs: vCenter Server has a built-in profiling mechanism that captures information about Reference-Counted objects that the server uses, and dumps them in profiler logs. These logs are supplements to the vpxd logs mentioned above, and provide information about memory and CPU usage of the server, as well as latencies of various tasks. These logs, in conjunction with the vpxd logs are very helpful in support situations.

```
<Program Data Folder>\VMware\VMware VirtualCenter\Logs\vpxd-profiler-X.log
```

6.1.2.2 vSphere Client and vCenter Management Web Services Logs

vSphere Client:

```
<Users Folder>\<User>\AppData\Local\VMware\viclient-X.log
```

Apache Tomcat Web Services logs:

```
<Program Files>\VMware\Infrastructure\Tomcat\Logs\
```

6.1.2.3 Other vCenter Services Logs

Query Service:

```
<Program Data Folder>\VMware\VMware VirtualCenter\Logs\vws.log
```

Stats Retrieval Service (Performance Charts):

```
<Program Data Folder>\VMware\VMware VirtualCenter\Logs\stats.log
```

Storage Management Service:

```
<Program Data Folder>\VMware\VMware VirtualCenter\Logs\sms.log
```

Common Infrastructure Model Service:

```
<Program Data Folder>\VMware\VMware VirtualCenter\Logs\cim-diag.log
```

6.1.3 Tools in vSphere Client for Monitoring and Troubleshooting

vCenter Service Status: Provides a quick overview of the health of vCenter Server components, including vCenter database, vCenter Management Web Services, and Storage Management Service.

Alarms: You can define alarms that generate notifications in response to specified events, conditions and states that occur with the entities in the inventory. Some predefined alarms exist for monitoring clusters, hosts, datacenters, virtual machines, and so on. For example, the "virtual machine cpu usage" alarm is set automatically for each virtual machine and triggers a notification when a virtual machine's CPU usage exceeds the specified percentage.

Events: The vCenter Server records various events, which are defined as user actions and system actions that occur on objects, such as powering on/off a virtual machine, or rebooting a host.

Performance Charts: Performance charts provide a graphical interface for you to monitor the performance of the devices and objects managed by the vCenter Server. The **Overview** pane includes preconfigured charts for CPU, memory, network, and storage metrics. The **Advanced** pane supports manually configured charts for these metrics so that users can customize the graphs.

View System Logs: You can view the vCenter Server process logs and profiler logs mentioned in section 6.1.2.1 through vSphere Client from **Home > Administration > System Logs**. These logs include detailed information about the activities within the vCenter Server.

Maps: You can view the physical and virtual maps of the entities in the vSphere infrastructure.

6.2 Monitoring vSphere ESX/ESXi Hosts and VMs

There are multiple tools that can be used to monitor the ESX/ESXi host and VM performance. Here we discuss two such very useful tools. One is esxtop and the other is vSphere Client.

Esxtop allows the monitoring and collection of data for all system resources, including CPU, memory, disk and network, for a single host. When used interactively, this data can be viewed on different types of screens: one each for CPU, memory, disk, network and interrupt statistics. The VMware community article *Interpreting esxtop Statistics* [4] describes esxtop in detail.

The vSphere Client can also monitor system performance across multiple hosts and VMs, utilizing over 150 performance counters exposed by the vSphere infrastructure. The VMware Technology Exchange talk *VI Performance Monitoring* [19] describes how to monitor and use these counters in detail.

Administrators can also use the VMware vSphere API to write custom tools to monitor performance counters for ESX/ESXi hosts and VMs. For more details, please refer to *VMware APIs and DSKs Documentation* [13].

7 Case Studies and Performance Improvements for vSphere 4.1

In this section, we present experimental results from a number of case studies to demonstrate the performance improvements made in vSphere 4.1. See section 3, "What's New in vCenter Server 4.1?" for a list of these improvements. Note that, in all the experiments, the vCenter Server was run on a physical machine.

7.1 Improved Performance at Higher vCenter Server Inventory Limits

Experiment

In order to understand how vSphere 4.1 performs at the supported maximum inventory size with respect to the numbers of VMs and hosts per vCenter Server instance, we compared it with vSphere 4.0 at its supported maximum inventory size. Table 9 shows the inventory size used for this experiment.

Table 9. VM and host inventory for the experiment

Version	Hosts	Powered On VMs	Registered VMs	Clusters
vSphere 4.0	320	3000	5000	10
vSphere 4.1	1024	10000	15000	32

In order to represent a typical customer workload, we analyzed data from different customer environments and defined workloads that covered a mix of various VM operations. The frequency of each operation was tuned according to how commonly the operation occurs in customer environments. Two primary workloads were defined:

- *Heavy load:* equivalent to 8 concurrent operations/cluster
- *Light load:* equivalent to 2 concurrent operations/cluster

The concurrent operations are those commonly seen in customer environments, such as clone, power-on, power-off, and reconfigure. Based on customer data, most environments have their normal peak load closer to the light load if not much lower in terms of operation arrival rate. However, we defined the heavy load in order to characterize vCenter Server for environments with sudden bursts in load. Each workload was run for an hour in order to ensure that the throughput could be sustained for a longer duration.

VMware HA and DRS were enabled on all the clusters. As you will see from the experimental results, we have significantly improved in throughput and latency under both workloads.

Testbed

vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon X5570 (Nehalem), 8 cores @ 2.93GHz w/ SMT enabled, 48GB RAM

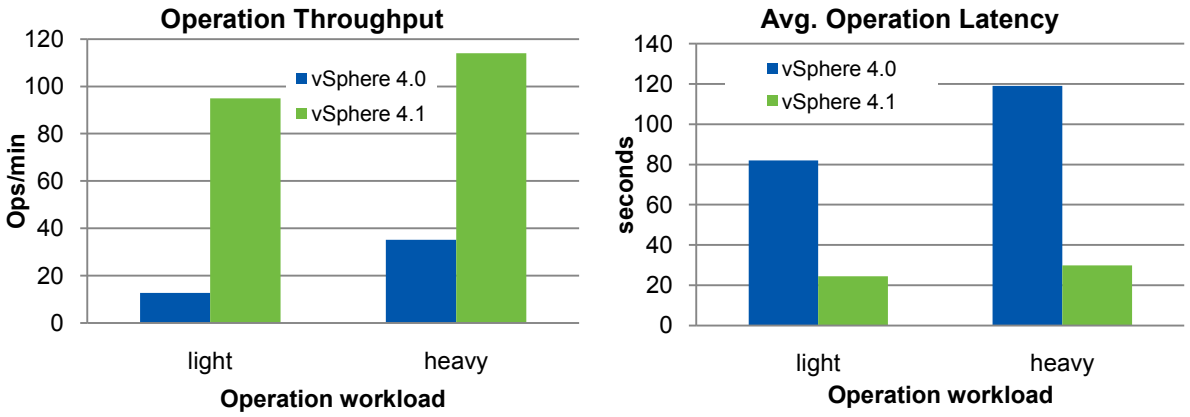
vCenter Database: 64-bit Windows 2008 Server SP1, Intel Xeon 5355, 8 cores @ 2.66GHz, 32GB RAM

Results

In addition to increased supported limits, vCenter Server also improved significantly in operational throughput across the whole vCenter Server inventory, as shown in Figure 3(a). Furthermore, as shown in Figure 3(b), as the operation workload increases, the average operational latency in vSphere 4.1 does not change as much as that in vSphere 4.0.

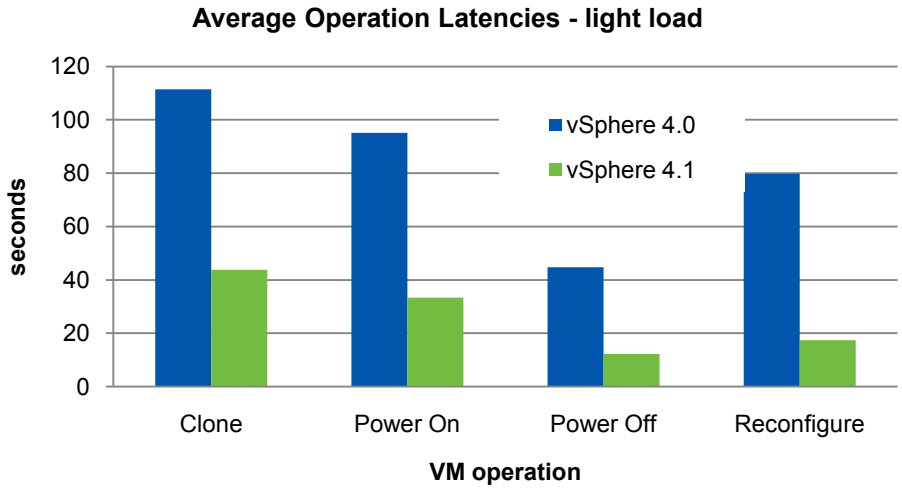
It is important to note the relative improvements instead of the absolute numbers, because the absolute numbers depend not only on hardware, but also on other factors such as inventory and workload.

Figure 3. Operation throughput (a) and average operation latency (b): vSphere 4.1 vs. vSphere 4.0



In vSphere 4.1, the average latencies of such operations as Clone, Power On, Power Off and Reconfigure, have improved significantly. Figure 4 shows these operation latencies under the light load. For example, VM Power On latency has improved 3 times with 3 times as many VMs and hosts in the inventory.

Figure 4. Average operation latencies of common VM operations with 2 concurrent operations per cluster: vSphere 4.1 vs vSphere4.0



7.2 Improved Performance at Higher Cluster Inventory Size

Experiment

In order to understand how vSphere 4.1 performs at the supported maximum cluster size with respect to the number of VMs per cluster instance, we compared it with vSphere 4.0 at its supported maximum cluster size. The inventory sizes used for the experiment are shown in Table 10.

Table 10: VM and host inventory for the cluster experiment

Version	Hosts	Powered On VMs	Registered VMs	Clusters
vSphere 4.0	32	1000	1000	1
vSphere 4.1	32	3000	3000	1

In order to represent a typical customer workload, we analyzed data from different customer environments and defined workloads that covered a mix of various VM operations. The frequency of each operation was tuned according to how commonly the operation occurs in customer environments. Two primary workloads were defined:

- *Heavy load:* equivalent to 16 concurrent operations/cluster
- *Light load*: equivalent to 4 concurrent operations/cluster

Based on customer data, most environments have their normal peak loads closer to the light load if not much lower in terms of operation arrival rate. However, we defined the heavy load in order to characterize vCenter Server for environments with sudden bursts in load. Each of the workloads was run for an hour in order to ensure that the throughput could be sustained for a longer duration.

VMware HA and DRS were both enabled on the cluster. As you can see we have significantly improved in throughput and latency under both workloads.

Testbed

vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon E5420, 8 cores @ 2.50GHz, 16GB RAM

vCenter Database: 64-bit Windows 2008 Server SP1, AMD Opteron 2212 HE, 4 cores @ 2GHz, 12GB RAM

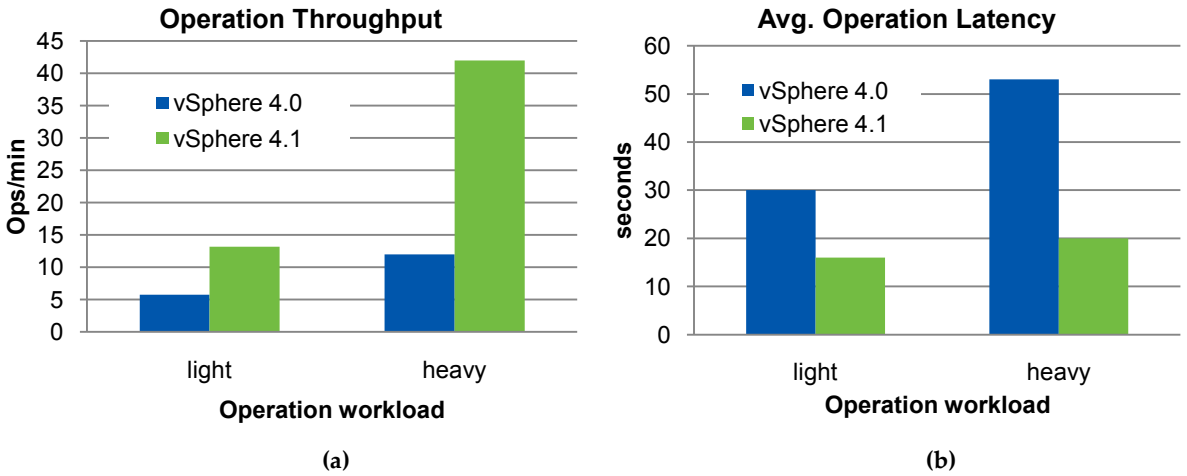
Results

In addition to increased supported limits, vCenter Server cluster performance has also improved significantly in terms of operational throughput and latencies at these higher limits. This is shown in Figure 5(a) and 5(b) below. The operational throughput is defined in terms of the total number of operations completed per minute on the cluster.

Furthermore, with vSphere 4.1 the average operation latency is not as significantly impacted by an increase in the workload, and operation throughput scales much more gracefully with increased load.

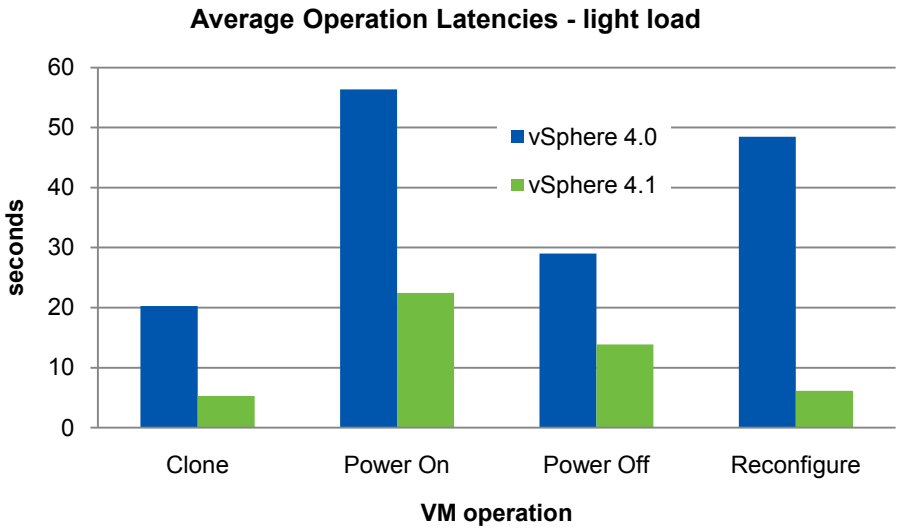
The important thing to take away is the relative improvement because the absolute numbers are not only hardware dependant, but also dependant on other factors such as inventory and workload.

Figure 5. Operation throughput (a) and average operation latency (b) for a single cluster: vSphere 4.1 vs vSphere4.0



As the overall latency improvement suggests, vCenter Server 4.1 has improved latency for such common VM operations as Clone, Power On, Power Off and Reconfigure. VM Power On has improved more than 2 times with 3 times as many VMs and hosts in the inventory. Figure 6 shows the operation latencies under the light load. We observed similar improvements in vSphere 4.1 under heavy load.

Figure 6. Average operation latencies of common VM operations with 4 concurrent operations in a single cluster: vSphere 4.1 vs vSphere4.0



7.3 Improved End-to-End Latency for Concurrent Operations in a Cluster

Experiment

We performed a set of operations on several hundred VMs concurrently to assess the scalability of vCenter Server, and calculated the average end-to-end latency to complete each type of operations. The vCenter Server inventory consisted of 32 ESX hosts within a cluster with DRS and HA enabled. The inventory sizes used for the experiment are shown in Table 11.

Table 11. VM and host inventory for concurrent operations in a cluster

Version	Hosts	Registered VMs	Clusters
vSphere 4.0	32	1024	1
vSphere 4.1	32	3200	1

Testbed

vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon E5420, 8 cores @ 2.50GHz, 16GB RAM

vCenter Database: 64-bit Windows 2008 Server SP1, AMD Opteron 2212 HE, 4 cores @ 2GHz, 12GB RAM

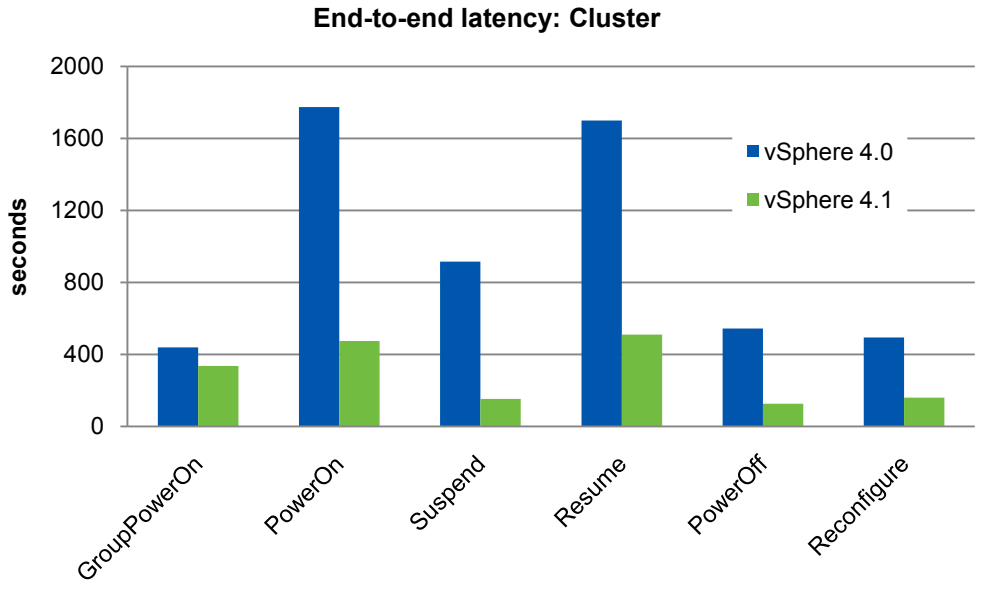
Results

We observed improved end-to-end latencies in vSphere 4.1 for all the concurrent operations performed. By end-to-end latency we refer to the time it takes to complete the operation on all the associated VMs.

Figure 7 shows the average improvements in end-to-end latency for executing such concurrent operations on 500 virtual machines in vSphere 4.1 vs. those in vSphere 4.0 (both at their inventory limits as described in the "Testbed" section). Note that inventory limits in vSphere 4.1 are much larger than those in vSphere 4.0.

There were some cluster-level locking improvements in vCenter Server 4.1 that helped improve the performance of several concurrent operations in the cluster. As is evident from Figure 7, several VM power related operations have improved significantly in vSphere 4.1.

Figure 7. End-to-end latencies of concurrent operations in a single cluster: vSphere 4.1 vs vSphere4.0



7.4 Faster vCenter Server Startup

Experiment

When the vCenter Server is started up, information is loaded from the vCenter Server database. During this phase. you will not be able to connect to the vCenter Server using the vSphere Client. In this experiment, we measured the startup time of vCenter Server for various inventory sizes.

Testbed

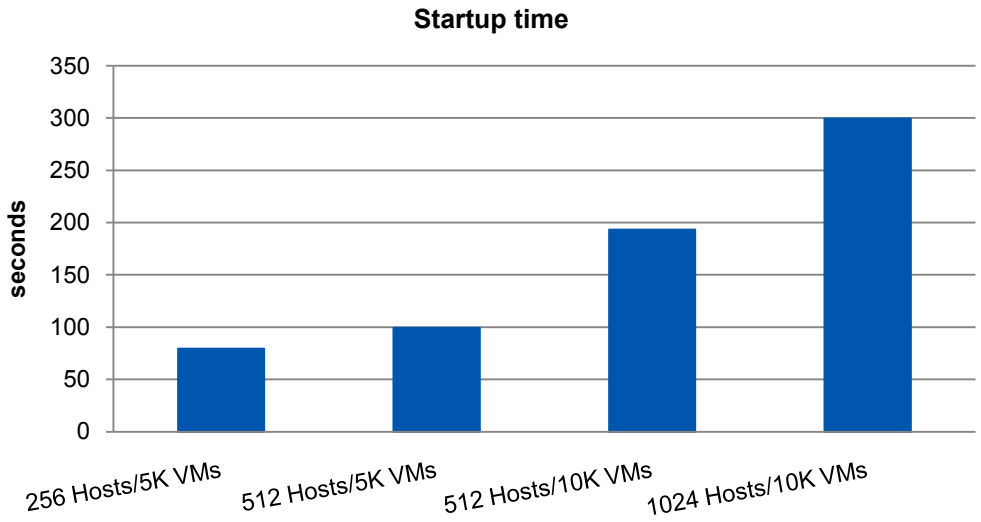
vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon X5570 (Nehalem), 8 cores @ 2.93GHz w/ SMT enabled, 48GB RAM

vCenter Database: 64-bit Windows 2008 Server SP1, Intel Xeon 5355, 8 cores @ 2.66GHz, 32GB RAM

Results

As Figure 8 below suggests, vCenter Server startup time scaled with inventory size and increased with both increasing numbers of VMs and hosts. In these experiments, all the VMs were powered on. Close to the maximum supported limit, startup time was around 5 minutes. Note that the absolute number of seconds for startup time depends on the vCenter Server and database hardware as well as the network distance between the two components. In these experiments, vCenter Server and the database were on separate machines, but in the same subnet to reduce the number of network hops.

Figure 8. vCenter Server start-up time with VM and host inventory scaling



7.5 Better vSphere Client Scaling

Experiment

In vSphere 4.1, vCenter Server can support up to 100 vSphere Clients. In this experiment, we studied three experiments varying the number of connected vSphere Clients: 0, 50, and 100. We created a medium sized inventory of 500 hosts and 10,000 VMs. We ran 20 scripted SDK clients generating a background load of various operations including create/delete folders, create/delete snapshots, power on/off, clone, delete, reset, and reconfigure VMs.

Testbed

vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon X5570 (Nehalem), 8 cores @ 2.93GHz w/ SMT enabled, 48GB RAM

vCenter Database: 64-bit Windows 2008 Server SP1, Intel Xeon 5355, 8 cores @ 2.66GHz, 32GB RAM

Results

For each experiment, vCenter Server maintained the same throughput for the operations that SDK clients generated. However, the resource utilization on the vCenter Server machine increased with the number of connected vSphere Clients, as expected.

Result 1: Maintain the same throughput and latency (as expected)

SDK clients generated the same load on the background for three experiments, each with a different number of connected vSphere Clients: 0, 50, and 100.

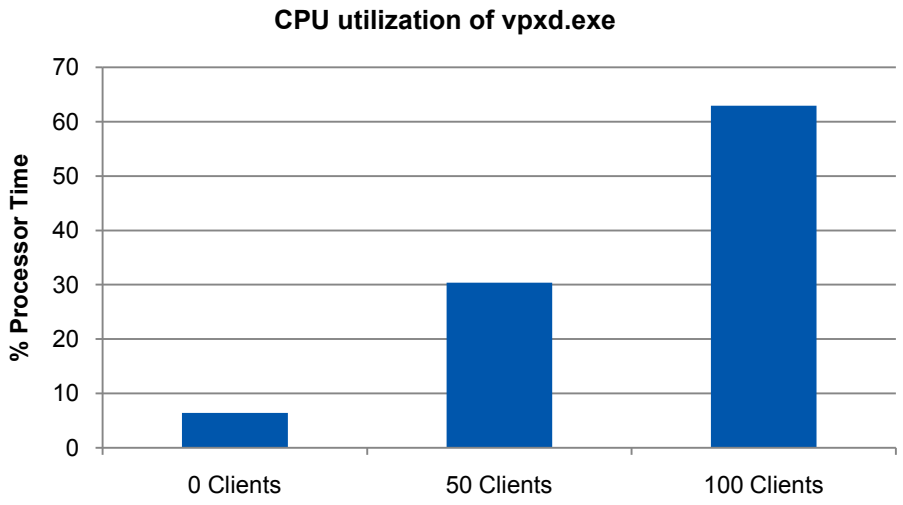
- vCenter Server was capable of maintaining the same throughput of 16-17 operations per minute with a very small variation for each experiment.
- The average latency per operation also remianed stable around 12-13 seconds per operation.

Result 2: Higher resource utilization on vCenter Server system

- vCenter Server used on average 20-25MB extra memory for each connected vSphere Client.
- vCenter Server utilized more CPU for each connected vSphere Client. In Figure 9, CPU utilization of the vCenter Server (`vpxd.exe`) is presented with varying number of vSphere Clients under load.

Be aware that the operating system and other applications—including VMware vCenter Management Web Services (`tomcat6.exe`)—also used computational resources during this experiment. The CPU utilization reported here is with a background load being generated by the SDK clients. For an idle setup, in the absence of load, the CPU utilization is small and does not vary noticeably with the number of connected vSphere Clients.

Figure 9. vCenter Server (vpxd.exe) CPU utilization (normalized over all processor cores on the physical machine) with varying number of vSphere Clients under load



Result 3: Higher variation for vSphere Clients getting updates from vCenter Server

vCenter Server has to inform all the connected vSphere Clients. The parallelism is bounded by the number of CPUs and the network paths. As a result, some of those vSphere Clients will receive the updates sooner than the others. A larger number of connected vSphere Clients can result in a higher observed latency for the updates in *some* of those clients.

7.6 Faster Standalone Host Operations

7.6.1 Operations on a Single Standalone Host

Operations that can be run on a host using the VMware vSphere API can be broadly categorized as:

- VM operations
- Host management operations

- Inventory operations

In vSphere 4.1, the latencies for VM and host operations have improved by up to 3 times. In this section, we describe a few experiments that we did to compare the performance of VM and host operations on a single standalone host in vSphere 4.1 to that in vSphere 4.0.

Experiment

Run the following concurrent VM operations while scaling the number of VMs on the host

- Create VMs
- Power On VMs

Run the following host operations while scaling the number of powered on VMs on the host

- Add host
- Reconnect host

Setup

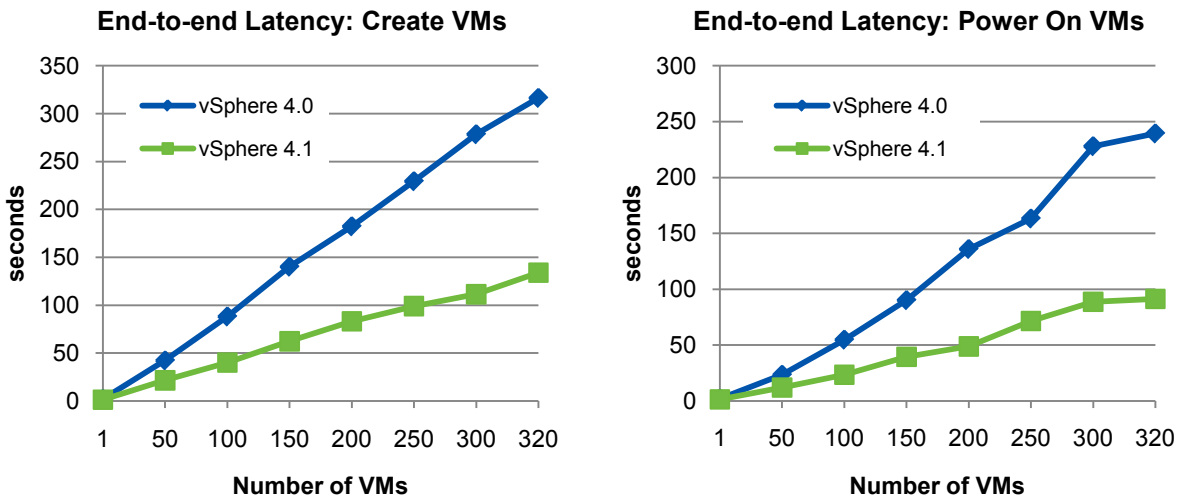
vCenter Server and Database: 64-bit Windows 2008 Server SP1, AMD Opteron 2378 (Shanghai), 4 cores @ 2.4GHz, 8GB RAM

Host: Intel Xeon X5570 w/ SMT enabled in the BIOS, DDR3 1033 MHz 96 GB RAM

Result

In vSphere 4.1, the latencies for all concurrent VM operations have significantly improved as compared to vSphere 4.0. The latencies for many operations have improved by more than 2 times. In Figure 10(a) and 10(b), we show the latencies for creating VMs and powering them on a standalone host. The X-axis is the number of VMs and the Y-axis is the latency for doing the respective VM operation. As can be seen, the latency for creating 320 VMs has improved by 2.5 times and those for powering on 320 VMs have improved by almost 3 times.

Figure 10. Latencies for concurrent Create-VMs (a) and concurrent Power-On-VMs (b) on a single standalone host

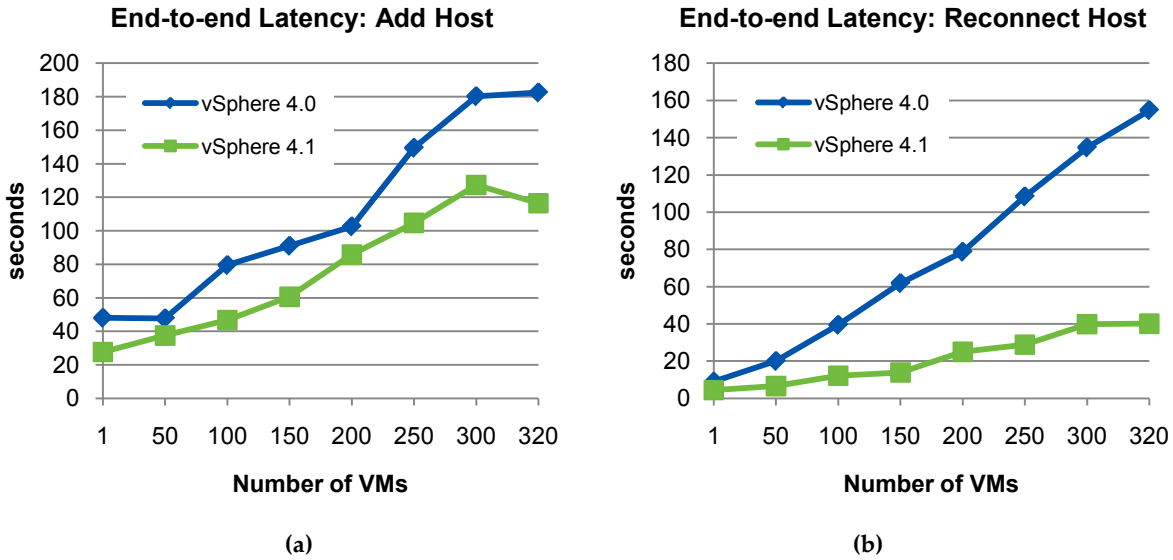


(a) (b)

In vSphere 4.1, the latencies for host operations have also significantly improved as compared to vSphere 4.0. In Figure 11(a) below, we show the latencies for adding a standalone host to a datacenter when a given number of VMs are powered on. The X-axis shows the number of VMs powered on and the Y axis shows the

latency of adding a host to vCenter Server. Similarly, in Figure 11(b), we show the latencies for reconnecting a standalone host to vCenter Server while a given number of VMs are powered on. We see approximately 60% improvement in the add host latencies while there is more than 3 times improvement in the reconnect host latencies. Adding a host with 320 VMs powered on takes less time than adding a host with 300 VMs powered on. This discrepancy can be attributed to the noise in our measurements.

Figure 11. Latencies for adding (a) and reconnecting (b) a standalone host with VMs powered on



(a) (b)

7.6.2 Concurrent Operations Across Multiple Standalone Hosts

Experiment

We performed a set of concurrent operations on several hundred VMs to assess the scalability of vCenter Server, and calculated the average end-to-end time to complete each type of operation. We also calculated the average power on operation latency when running power on operations in various batch sizes, to demonstrate the scalability of the group power on API. The group power-on method can be used as an alternative to the regular power-on method when powering on several VMs. Group power-on reduces the end-to-end latencies of VM power-on operations by synchronizing with the ESX/ESXi hosts more efficiently.

The vCenter Server inventory consisted of 32 standalone hosts, without any cluster. The inventory sizes used for the experiment are shown in Table 12.

Table 12: VM and host inventory for concurrent operations on standalone hosts

Version	Hosts	Registered VMs	Clusters
vSphere 4.0	32	1024	0
vSphere 4.1	32	2560	0

Testbed

vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon E5420, 8 cores @ 2.50GHz, 16GB RAM

vCenter Database: 64-bit Windows 2008 Server SP1, AMD Opteron 2212 HE, 4 cores @ 2GHz, 12GB RAM

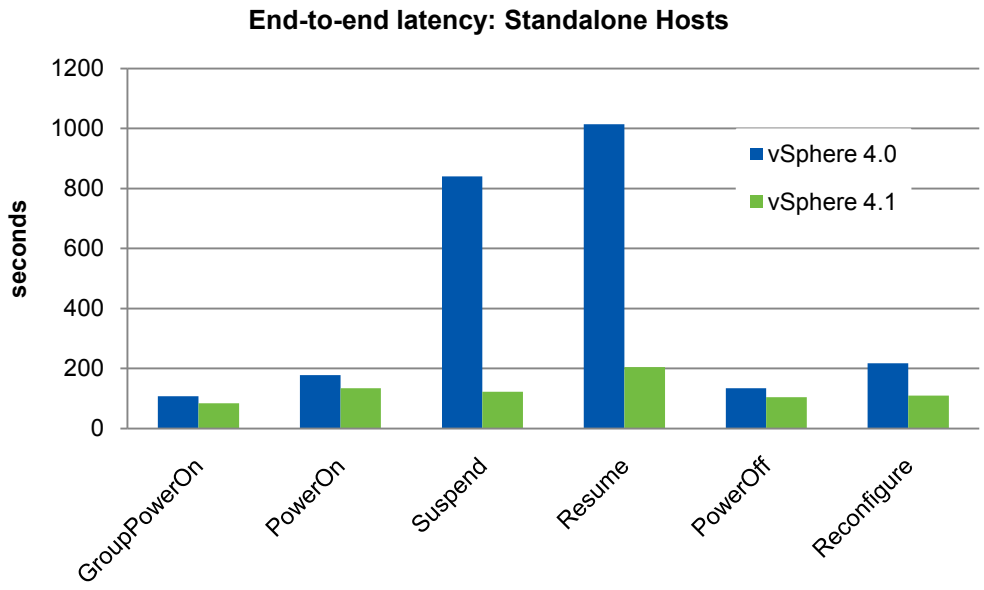
Results

We observed improved end-to-end latencies in vCenter 4.1 for all the concurrent operations performed. *End-to-end latency* refers to the time it takes to complete the operation on all the associated VMs.

Figure 12 shows the average improvements in end-to-end latencies for executing such concurrent operations on 500 virtual machines in vSphere 4.1 versus those in vSphere 4.0 (close to inventory limits).

There were some significant improvements in how vCenter Server handles concurrent power operations such as Suspend and Resume, as can be seen in Figure 12.

Figure 12. Concurrent operations end-to-end latency for standalone hosts: vSphere 4.1 vs. vSphere 4.0



7.7 Reduced VM Group Power-On Latency

Using the group poweron API is an efficient way to power on multiple virtual machines. When the user selects a group of virtual machines from the vSphere Client screen and performs a power-on operation on them, vCenter Server invokes the group power-on method.

Figure 13 shows the improvement in VM group power-on latency for 500 VMs in both standalone hosts and hosts in a DRS cluster in vSphere 4.1 vs. vSphere 4.0.

Figure 13. Group Power-On latency: vSphere 4.1 vs. vSphere 4.0

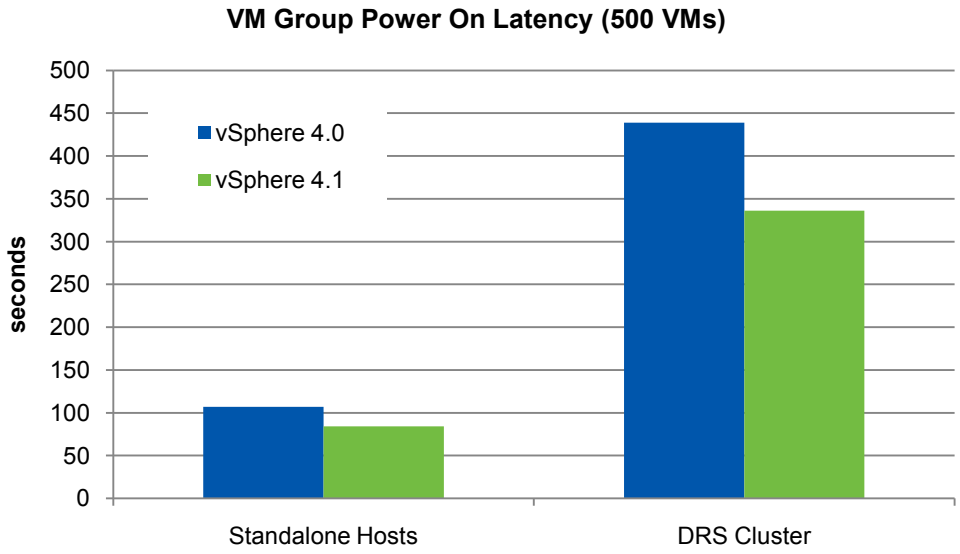


Figure 14 shows how the performance of the group power-on method improves as we power on more VMs in a group, when working with standalone hosts. Using the group power-on API makes powering on multiple VMs more efficient by decreasing the individual power-on latency.

Figure 14. Individual Power On latency when batched for standalone hosts: vSphere 4.1 vs. vSphere 4.0

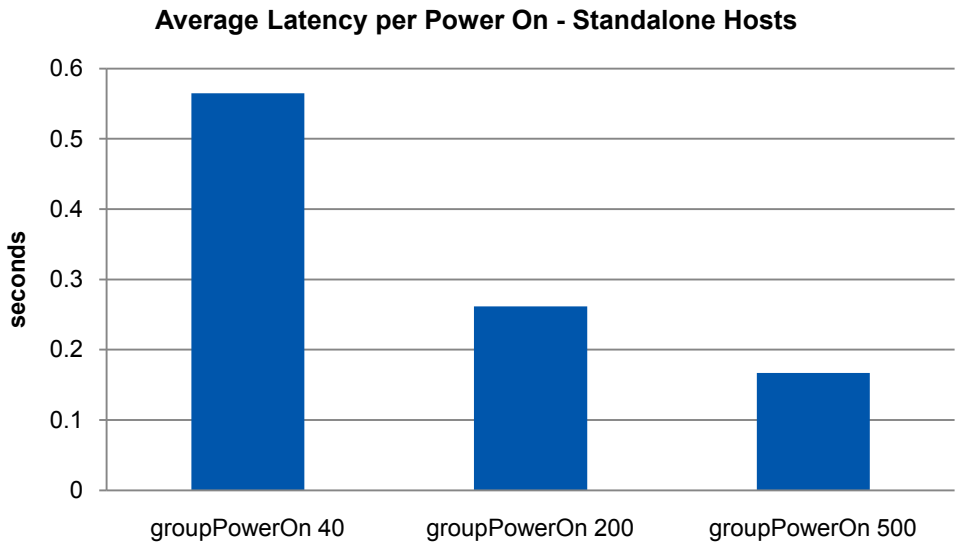
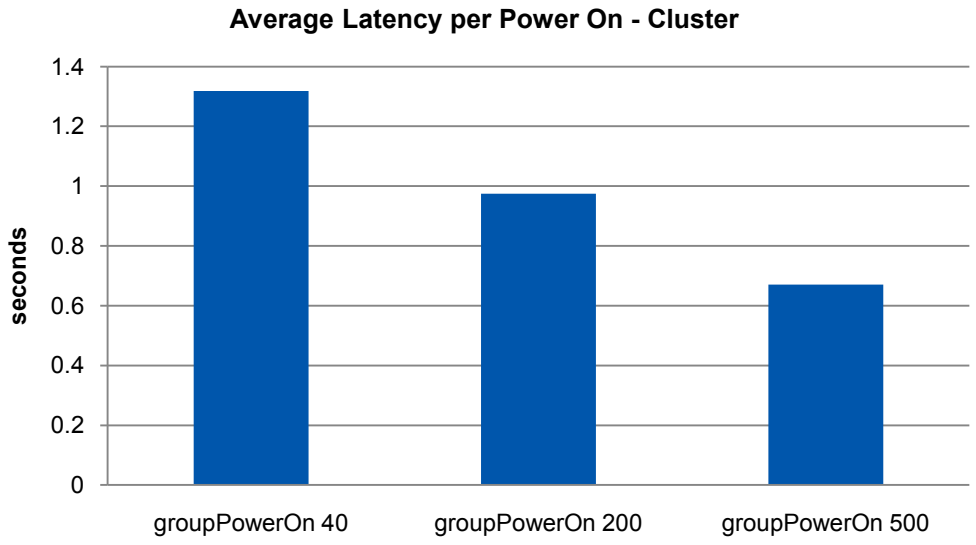


Figure 15 shows how power-on operations benefit from batching in a DRS-enabled cluster. When powering on a VM in a DRS-enabled cluster, there is an initial cost (in the form of running a DRS initial placement algorithm) for placing the VM in the appropriate resource to keep the load balanced across the cluster. When the group power-on method is used, the DRS initial placement algorithm is invoked only once for all the VMs. This makes powering on multiple VMs more efficient by amortizing the cost involved in powering them on across the cluster and therefore decreasing the average VM power-on latency.

Figure 15. Average Individual power-on latency when batched in a single cluster



7.8 Faster VM recovery with HA

Experiment

In this experiment, we measure how soon the VMs can restart after host failure. We used two hosts, initially one with VMs powered on and the other one empty. We powered off the host with VMs, and observed the VMs restart on the other host. Table 13 shows the inventory sizes used for the experiment. Note that for vSphere 4.1, there were 320 powered on VMs on the host, which is the vSphere 4.1 limit, while for vSphere 4.0 there were200 VMs.

Table 13: VM and host inventory for HA effectiveness experiment

Version	Hosts	Powered On VMs	Clusters
vSphere 4.0	2	200	1
vSphere 4.1	2	320	1

Testbed

vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon E5420, 8 cores @ 2.50GHz, 16GB RAM

vCenter Database: 64-bit Windows 2008 Server SP1, AMD Opteron 2212 HE, 4 cores @ 2GHz, 12GB RAM

ESX hosts: Intel Xeon X5570 (Nehalem), 8 cores @ 2.93GHz, 96GB RAM

Results

The failover performance is shown in Figure 16 and we compare the performance in vSphere 4.1 with that in vSphere 4.0 using three metrics:

- **Minimal Recovery Time:** Time from failure to when the first VM restarts
- **Average Recovery Time:** Average time for each VM to restart
- **Total Recovery Time:** Time from failure to when the last VM restarts

From Figure 16, we see that VMs recover much faster in vSphere 4.1, even with more VMs—recall that there were 320 VMs for vSphere 4.1 and 200 VMs for vSphere 4.0. The first VM restarted in 46 seconds after failure, the last VM restarted in less than 2.5 minutes after failure, and the average VM recovery time was 1.5 minutes.

Compared to vSphere 4.0, the average recovery time has improved by 50% and the total recovery time has improved by 60%.

In this experiment, there was only 1 alternative host. We expect the performance to be even better if there are more alternative hosts in the cluster, because the VM power-on operations will be distributed to multiple hosts, which will reduce the recovery time.

Figure 16. Recovery time and VM failover latency: vSphere 4.1 vs. vSphere 4.0

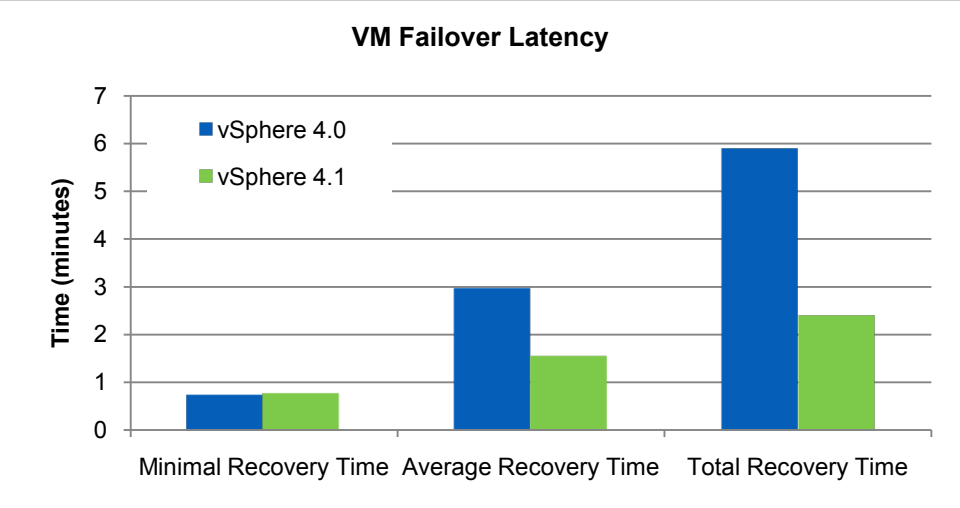
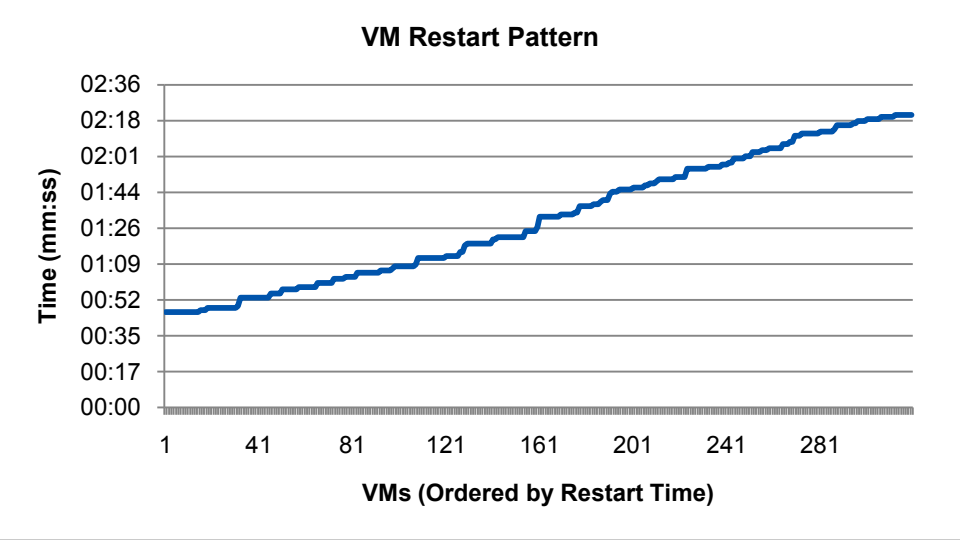


Figure 17 shows the pattern of how the VMs get restarted during failover. The x-axis represents the VMs ordered by the restart time, and the y-axis is the recovery time for each VM. From this graph, we can see that the VMs restart in a linear pattern.

Figure 17. VM restart pattern during failover in vSphere 4.1



7.9 Better Load Balancing with Improved DRS/DPM Effectiveness

Experiment

We ran the following two DRS/DPM effectiveness experiments with DRS/HA enabled. The inventory sizes used for the experiments are listed in Table 14.

Table 14. VM and host inventory for DRS/DPM effectiveness experiments

Version	Hosts	Powered On VMs	Clusters
vSphere 4.1	32	1280	1

DRS effectiveness test: DPM was disabled in this test so all 32 hosts were powered on. We started with all 1280 VMs being idle in the cluster. These VMs were evenly distributed across all 32 hosts, due to DRS load balancing. Then we injected a CPU load of roughly 20% of a core into 640 VMs on 16 of the hosts, which created an overload situation on these hosts and a significant load imbalance in the cluster. DRS was expected to re-balance the load across the cluster.

DPM effectiveness test: DPM was enabled in this test. We started with 1280 idle VMs in the cluster. Since the overall utilization in the cluster was low, DPM put 13 hosts into standby mode and only 19 hosts were kept powered on. We injected the same load increase into 640 VMs, causing most of the 19 hosts to be overloaded. DPM/DRS was expected to power on all standby hosts and re-balance the load across the cluster.

Testbed

vCenter Server: 64-bit Windows 2008 Server SP1, Intel Xeon E5420, 8 cores @ 2.50GHz, 16GB RAM

vCenter Database: 64-bit Windows 2008 Server SP1, AMD Opteron 2212 HE, 4 cores @ 2GHz, 12GB RAM

ESX hosts: Intel Xeon E5420 CPU, 8 cores @ 2.5 GHz, 32 GB RAM

Network: 1Gb vMotion network configured on a private VLAN

Host enter/exit standby protocol: IPMI

Results

Figure 18 and 19 show the results from the DRS/DPM effectiveness experiments. In all these figures, the x-axis represents time in 5-minute intervals. Based on these charts, we derive following metrics for capturing the effectiveness of the DRS/DPM algorithm:

- **Time to Respond:** How many intervals does DRS/DPM take to respond to the VM load changes?
- **Time to Stabilize:** How many intervals does it take the cluster to reach a steady state?
- **Number of vMotions:** How many vMotions are recommended by DRS to balance the cluster?
- **Cluster Imbalance:** The value of the "Current host load standard deviation" in the cluster, as defined in section 5.4.5*
- **Number of Hosts Powered-On:** The total number of hosts powered on by DPM.

* **Note**: The final target cluster imbalance was 0.05 in both experiments, determined by the DRS algorithm.

Figure 18. DRS effectiveness test result: cluster imbalance in each 5-minute interval

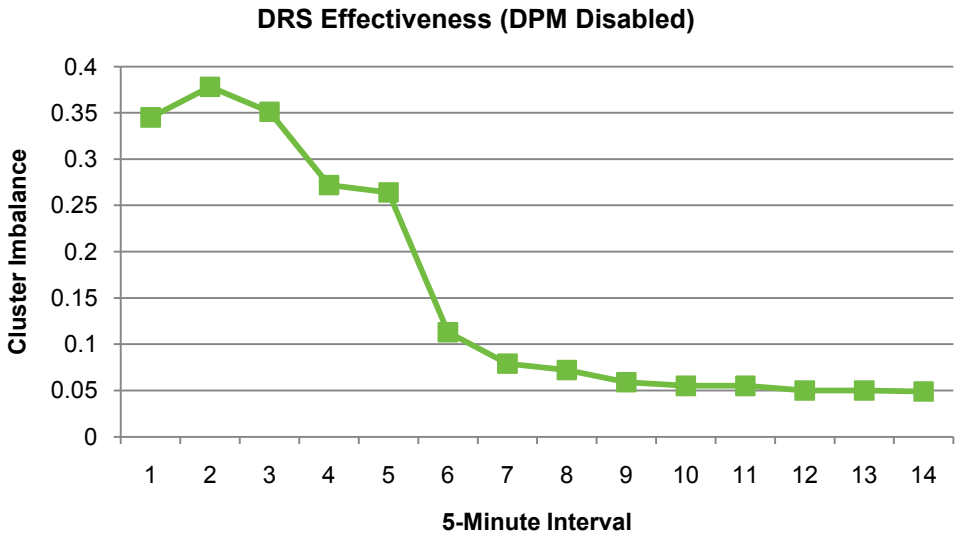
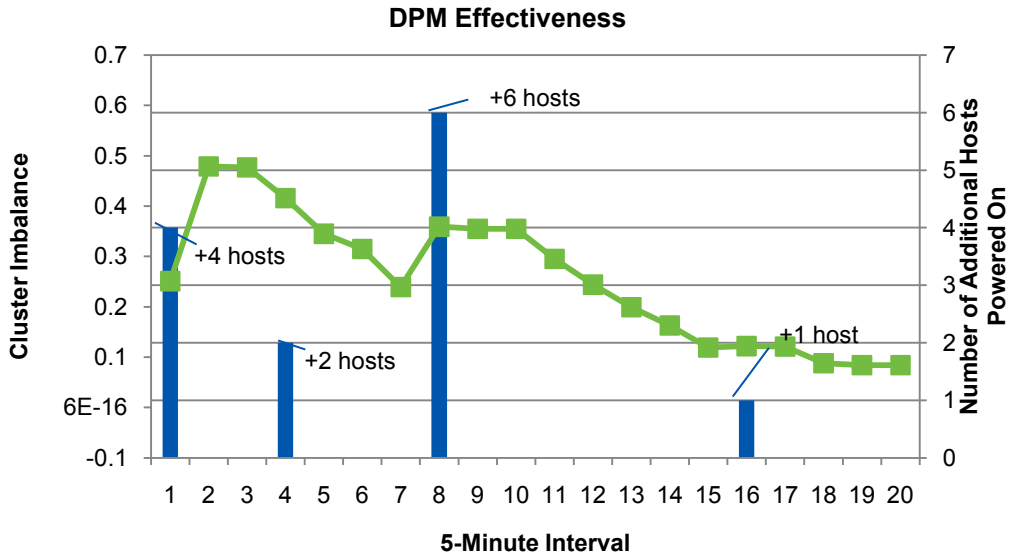


Figure 19. DPM effectiveness test result: cluster imbalance (green line) and number of hosts powered on by DPM (blue bars) in each 5-minute interval



In the DRS effectiveness experiment (Figure 18), the cluster started with a high imbalance of 0.345 initially after the VM load change. DRS responded within the first 5-minute interval and was able to gradually reduce the cluster imbalance and reach a "Load balanced" state in the 14th interval. Note that this amount of time is fairly reasonable because DRS needed to recommend a total of 274 vMotions in order to re-balance the cluster (with approximately 20 vMotions per interval) due to the significant VM load change injected in this test

In the DPM effectiveness experiment (Figure 19), the cluster started with 19 powered-on hosts. All of them were overloaded after the VM demand change, with a slight imbalance in the cluster. DPM responded to the overload situation within the first 5-minute interval, and started bringing 4 hosts out of standby mode immediately. Over the course of the experiment, a total of 13 hosts were powered on by DPM and the cluster reached full capacity in the end. When more hosts were being powered on, the cluster imbalance

increased, and DRS continuously migrated VMs to re-balance the load within the cluster. It took 20 intervals for the cluster to reach a steady state, after a total of 287 vMotions, with approximately 14 vMotions per interval. The extra overall latency in the DPM-enabled case is attributed to the additional latency in powering on 13 standby hosts to accommodate the increased VM demands.

Notice that in the DPM-enabled case, the cluster remained slightly "Load imbalanced" when DRS stopped recommending more VM migrations. As discussed in section 5.4.6, "DRS Performance Tuning," this is a result of the cost-benefit analysis and minimum goodness filtering within the DRS algorithm. Given that all the VMs were receiving their entitled resources at that time, a slight load imbalance should not be a big concern.

8 Conclusion

In this paper, we discussed some of the great new performance features for vCenter Server that come with vSphere 4.1, hardware sizing guidelines and software requirements, performance best practices with performance tuning, monitoring, and troubleshooting tips, performance monitoring tools, and case studies to demonstrate the performance improvements made in vSphere 4.1. Some highlights of the performance best practices for vCenter Server 4.1 include:

- Make sure you size your system properly according to the inventory size.
- The number of network hops between vCenter Server and the ESX host affects operational latency. The ESX host should reside as few network hops away from the vCenter Server as possible.
- Effective resource planning and monitoring is necessary when using HA, FT, DRS, and DPM.
- Monitor Web Services and make sure the max heap size for Java virtual machine is set correctly according to Inventory size.
- For the vCenter Server database, separate database files for data and for logs onto drives backed by different physical disks, and make sure statistics collection times are set conservatively so that they will not overload the system.
- Be aware that the number of clients connected to vCenter Server affects its performance.
- Use performance monitoring tools to ensure the health of your system and to troubleshoot problems that arise. Performance charts give you a graphical view of statistics for your system.

9 References

[1] vSphere Basic System Administration:
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_admin_guide.pdf

[2] ESX and vCenter Server Installation Guide:
http://www.vmware.com/pdf/vsphere4/r41/vsp_41_esx_vc_installation_guide.pdf

[3] VMware vSphere Compatibility Matrixes:
http://www.vmware.com/pdf/vsphere4/r40/vsp_compatibility_matrix.pdf

[4] Interpreting esxtop Statistics: http://communities.vmware.com/docs/DOC-9279

[5] vSphere Resource Management Guide:
http://www.vmware.com/pdf/vsphere4/r41/vsp_41_resource_mgmt.pdf

[6] vSphere Availability Guide: http://www.vmware.com/pdf/vsphere4/r41/vsp_41_availability.pdf

[7] VMware vSphere 4 Fault Tolerance: Architecture and Performance:
http://www.vmware.com/files/pdf/perf-vsphere-fault_tolerance.pdf

[8] Resource Management with VMware DRS: http://www.vmware.com/resources/techresources/401

[9] Scripts for Proactive DRS: http://communities.vmware.com/docs/DOC-10231

[10] VMware Distributed Power Management Concepts and Use: http://www.vmware.com/files/pdf/DPM.pdf

[11] Scripts for Proactive DPM: http://communities.vmware.com/docs/DOC-10230

[12] VMware vCenter 4.0 Database Performance for MS SQL Server 2008: http://www.vmware.com/pdf/vsp_4_vcdb_sql2008.pdf

[13] VMware APIs and SDKs Documentation: http://www.vmware.com/support/pubs/sdk_pubs.html

[14] VMware vSphere Web Services SDK Documentation: http://www.vmware.com/support/developer/vc-sdk/

[15] VMware vSphere Web Services SDK Programming Guide: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/sdk40programmingguide.pdf

[16] Customizing the vSphere Client: http://www.vmware.com/support/developer/vc-sdk/vcplugin/vSphere_Plugin_40_Technote.pdf

[17] VMware vSphere Web Services SDK 4.0 Developer's Setup Guide: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/sdk40setupguide.pdf

[18] Programming Code Samples from VMware's Community Page: http://communities.vmware.com/community/developer/codecentral

[19] VI Performance Monitoring: http://www.vmware.com/files/webinars/communities/VI-Performance-Monitoring.pdf

[20] VMware vCenter Database Sizing Calculator for Oracle http://www.vmware.com/support/vsphere4/doc/vsp_4x_db_calculator_oracle.xls

[21] VMware vCenter Database Sizing Calculator for Microsoft SQL Server http://www.vmware.com/support/vsphere4/doc/vsp_4x_db_calculator.xls

10 About the Authors

Nikhil Bhatia is a Performance Engineer at VMware. In this role, his primary focus is to evaluate and improve the performance of VMware vSphere. Prior to VMware, Nikhil was a researcher at Oak Ridge National Laboratory (ORNL) in the Computer Science and Mathematics Division. Nikhil received a Master of Science in Computer Science from the University of Tennessee, where he specialized in tools for performance analysis of HPC applications.

Chirag Bhatt is a Staff Engineer with the Performance Engineering group at VMware. In this role, he has been focusing on VMware vCenter Server performance and scalability, as well as VMware DRS performance. Chirag received his M.S. degree in Computer Science from Stanford University.

Lei Chai is a member of technical staff with the Performance Engineering group at VMware. In this role, she has been focusing on analyzing and improving the performance and scalability of VMware's availability solutions. Lei received her Ph.D. in Computer Science from Ohio State University, where she specialized in High Performance Computing, InfiniBand, middleware optimizations in multi-core systems, and NFS over RDMA, and published a number of research papers on these topics.

Adarsh Jagadeeshwaran is a senior member of technical staff with the Performance Engineering group at VMware. He has been working on VMware vCenter Server scalability and performance. He holds a Master's degree in Computer Science from Syracuse University.

Alper T. Mizrak joined VMware, Research & Development in 2007 and he has been working on the performance and scalability of VMware products. Dr. Mizrak received his Ph.D. in Computer Science in December 2007 from the University of California, San Diego. During his graduate studies, Dr. Mizrak conducted research in the area of network security and distributed systems. His graduate research resulted in receipt of the William C. Carter Award—an award granted by the IEEE Technical Committee on Fault-Tolerant Computing and the IFIP WG-10.4 on Dependable Computing and Fault Tolerance - to recognize an individual who has made a significant contribution to the field of dependable computing through his or her graduate dissertation research; a book, "Secure Networking: Detecting Malicious Routers," published in 2008; 8 academic papers published in peer-reviewed journals, conferences and workshops; and 3 technical reports.

Xiaoyun Zhu is a Staff Engineer with the Performance Engineering group at VMware. In this role, she has been working on evaluating and improving performance and scalability of VMware's resource management solutions including VMware DRS and DPM. Prior to VMware, Xiaoyun was a Senior Research Scientist at HP Labs for eight years, developing automated IT systems and services management solutions using control theory, optimization, algorithms, and simulations. She received her Ph.D. in Electrical Engineering from California Institute of Technology. Dr. Zhu has co-authored over 50 refereed papers in journals and conference proceedings, and has been a program committee member for a number of conferences including IM, NOMS, DSOM, ICAC, and MASCOTS.

11 Acknowledgements

This white paper is the result of a collaborative effort among various teams within VMware, including the core vCenter Server team, the distributed resource management team, the availability team, as well as the vCenter Server performance team. The authors would like to thank *Kinshuk Govil*, *Ravi Soundararajan*, *Anne Holler*, and *Chethan Kumar* for reviewing the document and providing insightful feedback to help improve the quality of the paper. We would also like to thank *Rajit Kambo* and *Jennifer Anderson* for their support of this work. Finally, we would like to thank *Julie Brodeur* for editing the paper.