VMware vCloud™ Director 1.0 Performance and Best Practices

February 2011

PERFORMANCE STUDY

vmware®

Table of Contents

Introduction 4
vCloud Director Architecture 4
Terms Used in this Paper 5
Experimental Setup 6
Server Sizing for the Oracle Database 6
Database Connection Tuning 6
Sizing for Number of Cell Instances 7
LDAP Sync 8
LDAP Sync Latency for Different Numbers of Users 8
LDAP Sync Latency on a High Latency Network 9
Performance Tips 9
OVF File Upload 10
OVF File Upload Latency for Different File Sizes 11
Concurrent OVF File Upload Latency 12
File Upload in WAN Environment 13
Sizing and Performance Tuning Tips 15
Clone vApps across vCenter Server Instances 16
Experimental Setup 16
Two Separated Datastores 17
Datastore Shared Among ESX Hosts 17
Shared and Separated Datastores 18
Results 19
Performance Tuning Tips 19
Deploy vApp 20
Deploy Single vApp with Varying Number of VMs 20
Concurrently Deploy vApps with and without Fence Mode 22
Performance Tuning Notes 23
Inventory Sync 23
Sync Latency for Different Inventory Sizes 23
In-Memory Inventory Cache 24
Inventory Cache and JVM Heap Size Tuning 25
Inventory Sync Resource Consumption 26
Load Balancing VC Listener 27
Adjusting Thread Pool and Cache Limits 28

Conclusion 30
References 30
About the Authors 31
Acknowledgements 31

Introduction

VMware vCloud™ Director gives enterprise organizations the ability to build secure private clouds that dramatically increase datacenter efficiency and business agility. When coupled with VMware vSphere™, vCloud Director delivers cloud computing for existing datacenters by pooling virtual infrastructure resources and delivering them to users as catalogs.

This white paper addresses four areas regarding VMware® vCloud Director performance:

- vCloud Director sizing guidelines and software requirements
- Best practices in performance and tuning
- Performance characterization for key vCloud Director operations
- Behavior with low bandwidth and high latency networks

vCloud Director Architecture

Figure 1 shows the deployment architecture for vCloud Director. You can access vCloud Director from a Web browser or through the VMware vCloud API. Multiple vCloud Director server instances (cells) can be deployed with a shared database. In the current 1.0 release, only the Oracle database is supported. A vCloud Director server instance can connect to one or multiple VMware vCenter™ Server instances.

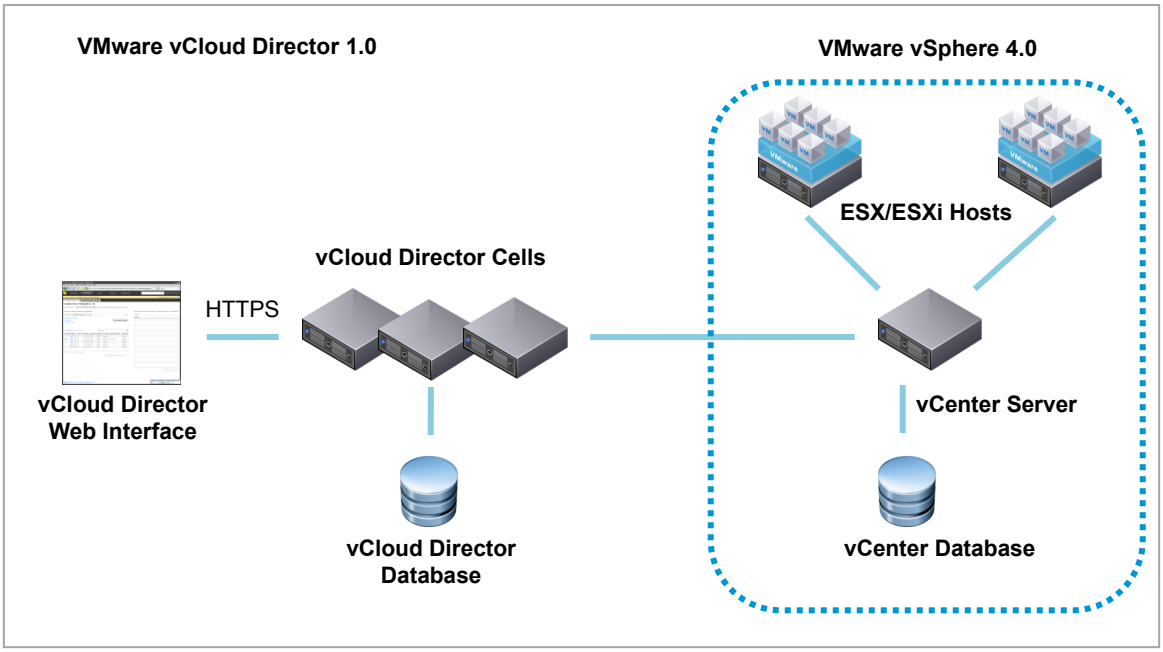

Figure 1. VMware vCloud Director High Level Architecture

Figure 2 shows the internal architecture of vCloud Director. Performance results for LDAP, Image Transfer (OVF File Upload), and VC Listener are included in this paper.

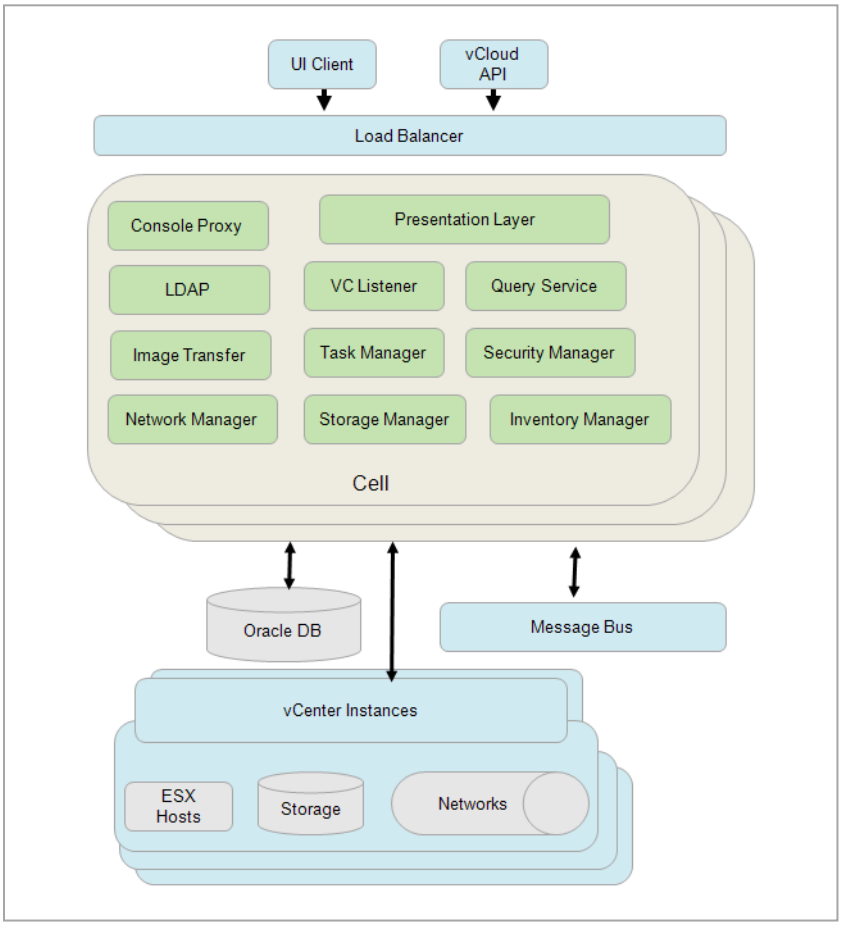

Figure 2. vCloud Director Internal Architecture

Terms Used in this Paper

Definitions for key concepts in vCloud Director 1.0 follow. These terms have been used extensively in this white paper. For more information, refer to the *vCloud API Programming Guide* [7].

- **vCloud Organization** – A vCloud organization is a unit of administration for a collection of users, groups, and computing resources. Users authenticate at the organization level, supplying credentials established by an organization administrator when the user was created or imported.
- **vCloud Virtual Datacenters** – A vCloud virtual datacenter (vDC) is an allocation mechanism for resources such as networks, storage, CPU, and memory. In a vDC, computing resources are fully virtualized, and can be allocated based on demand, service level requirements, or a combination of the two.

  There are two kinds of vDCs:

  – **Provider vDCs** – These vDCs contain all the resources available from the vCloud service provider. Provider vDCs are created and managed by vCloud system administrators.

  – **Organization vDCs** – These vDCs provide an environment where virtual systems can be stored, deployed, and operated. They also provide storage for virtual media, such as floppy disks and CD-ROMs.

An organization administrator specifies how resources from a provider vDC are distributed to the vDCs in an organization.

- **vCloud Catalogs** – Catalogs contain references to virtual systems and media images. A catalog can be shared to make it visible to other members of an organization, and can be published to make it visible to other organizations. A vCloud system administrator specifies which organizations can publish catalogs, and an organization administrator controls access to catalogs by organization members.
- **vCloud Cells –** vCloud cells are instances of the vCloud Director server. A vCloud cell is made up of several components, including VC Listener, Console Proxy, Presentation Layer, and others as shown in Figure 2.
- **vApp –** A vApp is an encapsulation of one or more virtual machines, including their inter-dependencies and resource allocations. This allows for single-step power operations, cloning, deployment, and monitoring of tiered applications, which span multiple VMs.

Experimental Setup

Table 1 shows the virtual machines that we configured in our test bed. All VMs ran on Dell PowerEdge R610 machines with 8 Intel Xeon CPUs @2.40GHz and 16GB RAM.

Table 1. Virtual Machine Configuration

	OPERATING SYSTEM	NUMBER OF VCPUS	RAM
Cell	64-bit Red Hat Enterprise Linux 5	4 vCPUs	8GB
Cell Database	64-bit Windows Server 2003		
vCenter Server			
vCenter Database			

This section describes performance tuning and sizing recommendations for the Oracle database.

Server Sizing for the Oracle Database

An Oracle database server configured with 16GB of memory, 100GB storage, and 4 CPUs should be adequate for most vCloud Director clusters.

Database Connection Tuning

Because there is only one database instance for all cells, the number of database connections can become the performance bottleneck. By default, each cell is configured to have 75 database connections, plus about 50 for Oracle's own use. Experiments in this paper used the default setting.

When vCloud Director operations become slower, increasing the database connection number per cell might improve the performance. If you change the CONNECTIONS value, you also need to update some other Oracle configuration parameters. Table 2 shows how to obtain values for other configuration parameters based on the number of connections, where *C* represents the number of cells in your vCloud Director cluster.

Table 2. Oracle Database Configuration Parameters

ORACLE CONFIGURATION PARAMETER	VALUE FOR C CELLS
CONNECTIONS	75 × *C* + 50
PROCESSES	= CONNECTIONS
SESSIONS	= PROCESSES × 1.1 + 5
TRANSACTIONS	= SESSIONS × 1.1
OPEN_CURSORS	= SESSIONS

A database administrator can set these values in Oracle.

For more information on best practices for the database, refer to the *vCloud Director Installation and Configuration Guide* [10] .

Sizing for Number of Cell Instances

vCloud Director can be easily scaled up by adding more cells to the system. We tested with up to 12 cell instances with 10 fully loaded vCenter Server instances.

For this experiment, the Oracle database instance ran in a host with 12 cores and 16GB RAM. Each cell ran in a virtual machine with 2 vCPUs and 4GB RAM.

In general, we recommend:

number of cell instances = n + 1 where n is the number of vCenter Server instances

This formula takes into account the considerations for VC Listener (which helps keep the vCenter Server inventory up-to-date in the vCloud Director database), cell failover, and cell maintenance. In "Inventory Sync" on page 23, we recommend a one-to-one mapping between VC Listener and the vCloud cell. This ensures the resource consumption for VC Listener is load balanced between cells. We also recommend having a spare cell for cell failover. This keeps the load of VC Listener balanced when one vCloud Director cell fails or is powered down for routine maintenance.

We assume here that the vCenter Server instances are managing a total of more than 2000 VMs. If the vCenter Server instances are lightly loaded, multiple instances can be managed by a single vCloud Director cell. For cases where vCenter Server is lightly loaded, use the following sizing formula:

number of cell instances = n ÷ 3000 + 1 where n is the number of expected powered on VMs

For more information on the configuration limits in vCenter Server 4.0 and 4.1, refer to *VMware vCenter Server 4.0 Configuration Limits* [4], *VMware vCenter Server 4.1 Configuration Limits* [5], and *VMware vCenter Server 4.1 Performance and Best Practice* [6].

LDAP Sync

vCloud Director supports importing users and groups from external LDAP servers. An external LDAP server is one that is not part of the cloud that vCloud Director manages. When a LDAP user logs in, the vCloud Director server checks if the necessary information for this user has been loaded. If the user is not loaded, vCloud Director issues a LDAP search request to fetch the user information. In addition to checking and fetching user information, the sync process also includes the removal of records for users who have been deleted in an external LDAP server.

LDAP Sync Latency for Different Numbers of Users

LDAP sync in vCloud Director is implemented with a single thread. All LDAP users are updated in a sequential manner. This approach ensures the resource consumption is small when a LDAP sync task runs a background job within the vCloud Director server.

In our tests, we observed that, as the number of users increases, latency increases. Figure 3 gives you an idea of what amount of LDAP sync latency to expect for the number of LDAP users on your system.

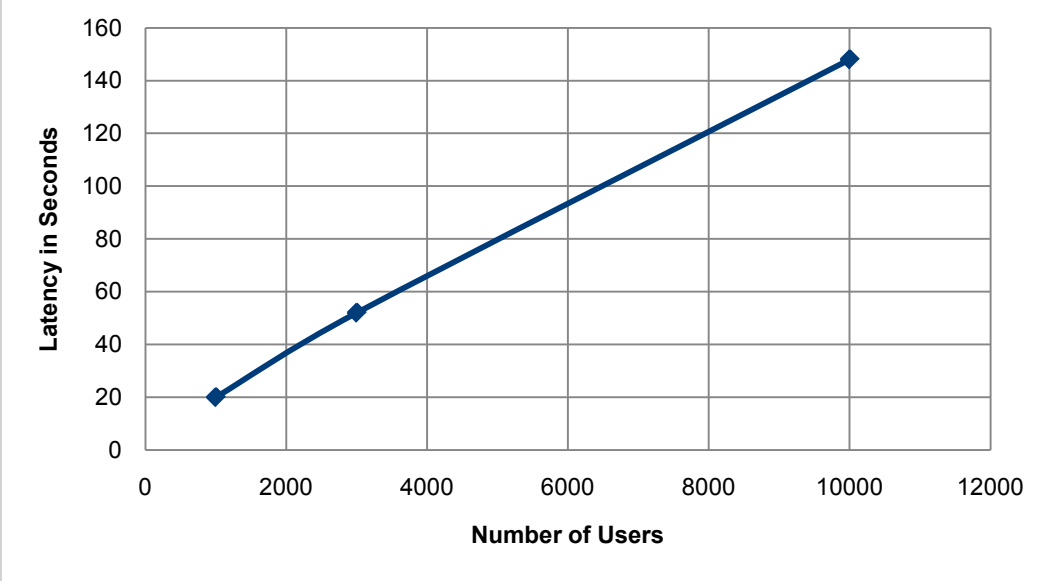

Figure 3. LDAP Sync Latency in Seconds for 1000, 3000, and 10,000 LDAP Users

LDAP Sync Latency on a High Latency Network

This test determines how much of a delay to expect for an LDAP sync operation when the external LDAP server's Active Directory contains a large number of users (10,000).

When the vCloud Director server is connected to the external LDAP server through a high latency network (for example, DSL, Satellite, or T1), the sync latency increases linearly as shown in Figure 4.

This experiment did not limit the network bandwidth. If the available network bandwidth were also limited, the LDAP sync latency might further increase.

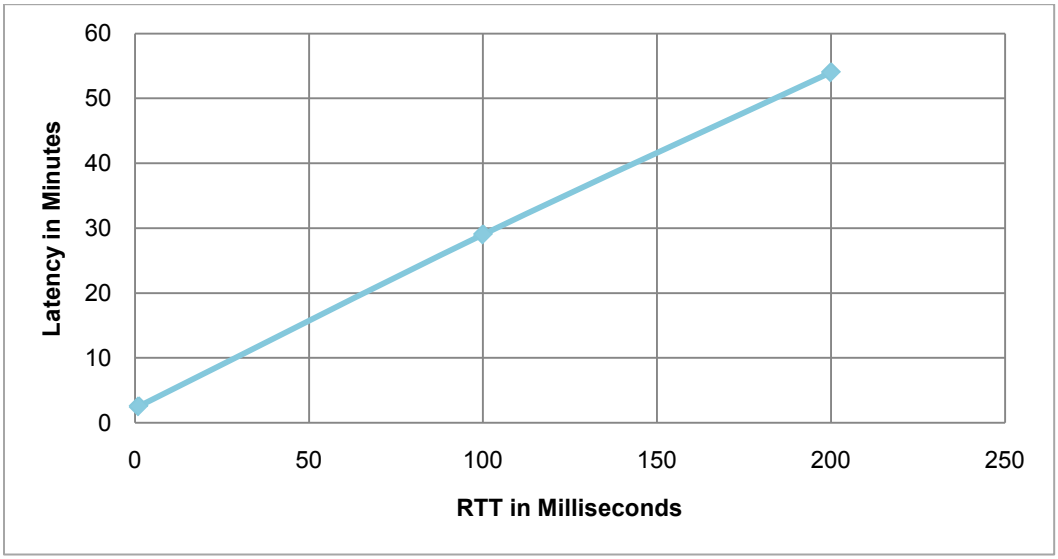

Figure 4. LDAP Sync Latency for Round-Trip Time (RTT) with 10,000 LDAP Users = 1ms, 100ms, and 200ms

Performance Tips

To speed up the time it takes for vCloud Director to initially import LDAP users from an external LDAP server, we recommend importing the groups first instead of importing individual users. When a group is imported, all users in that group will be able to log into the vCloud Director server. The vCloud Director server automatically imports the user information during the login process.

We also recommend that you infrequently run LDAP sync on the external LDAP server. When a LDAP user is initially imported, for any subsequent login, vCloud Director syncs up this user with the external LDAP server. The LDAP user information in the vCloud Director server database is always up to date, except if a user is deleted in the LDAP server. A LDAP sync operation will disable the user as "out of sync" from the vCloud Director database when a user is removed remotely.

OVF File Upload

The Open Virtualization Format (OVF) is an open, portable, efficient, and extensible format for packaging and distributing virtual systems. OVF was developed by the Distributed Management Task Force (DMTF), a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. The vCloud API supports Version 1 of the OVF standard. For more information, refer to *Open Virtualization Format Specification* [1] and *Open Virtualization Format White Paper* [2].

Because it is a widely-accepted standard format, OVF provides considerable flexibility in accommodating the needs of a diverse collection of virtualization technologies. While this flexibility entails more complexity than a vendor-specific format might require, it also provides many advantages.

- Virtual machines and appliances are distributed as OVF packages by many vendors.
- Many vendors, including VMware, offer tools that simplify creating and customizing OVF virtual machines, support converting virtual machines on existing virtualization platforms to OVF, or both.
- OVF has the power to express the complex relationships between virtual appliances in enterprise applications. Most of the complexity can be handled by the author of the appliance rather than the user deploying it.
- OVF is extensible, allowing new policies and requirements to be inserted by independent software vendors and implemented by the virtualization platforms that support them without requiring changes to other clients, other platforms, or the vCloud API itself.

Applications can be deployed in the vCloud infrastructure using vApps, which are made available for download and distribution as OVF packages. A vApp contains one or more VM elements, which represent individual virtual machines. vApps also include information that defines operational details for the vApp and the virtual machines it contains. After an OVF package is uploaded to a vDC, a vApp template is created. A vApp template specifies a set of files (such as virtual disks) that the vApp requires and a set of abstract resources (such as CPU, memory, and network connections) that must be allocated to the vApp by the vDC in which the template is deployed. Instantiation creates a vApp from the files specified in the template, and allocates vDC-specific bindings for networks and other resources. Figure 5 shows the state transitions from an OVF package to a vApp template and a vApp to be deployed. For more information, refer to *vCloud API Programming Guide* [7].

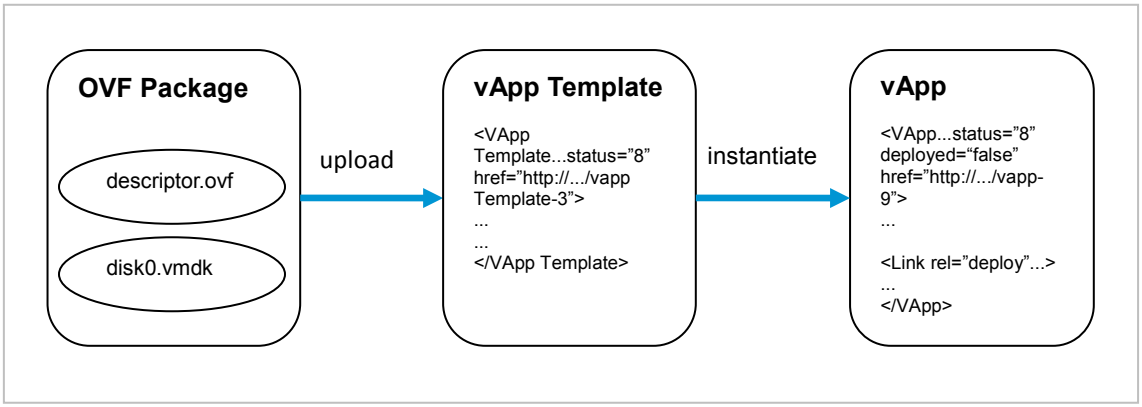

Figure 5. vApp State Transitions

As shown in Figure 6, an OVF file resides in a client machine. Before it can be uploaded into the cloud, the OVF file needs to be transferred to the vCloud Director cell. This transfer could very likely happen in a WAN environment where the bandwidth and speed of the network are limited. The performance also changes when there are concurrent file transfers within one cell. The following sections describe all of these aspects in more detail.

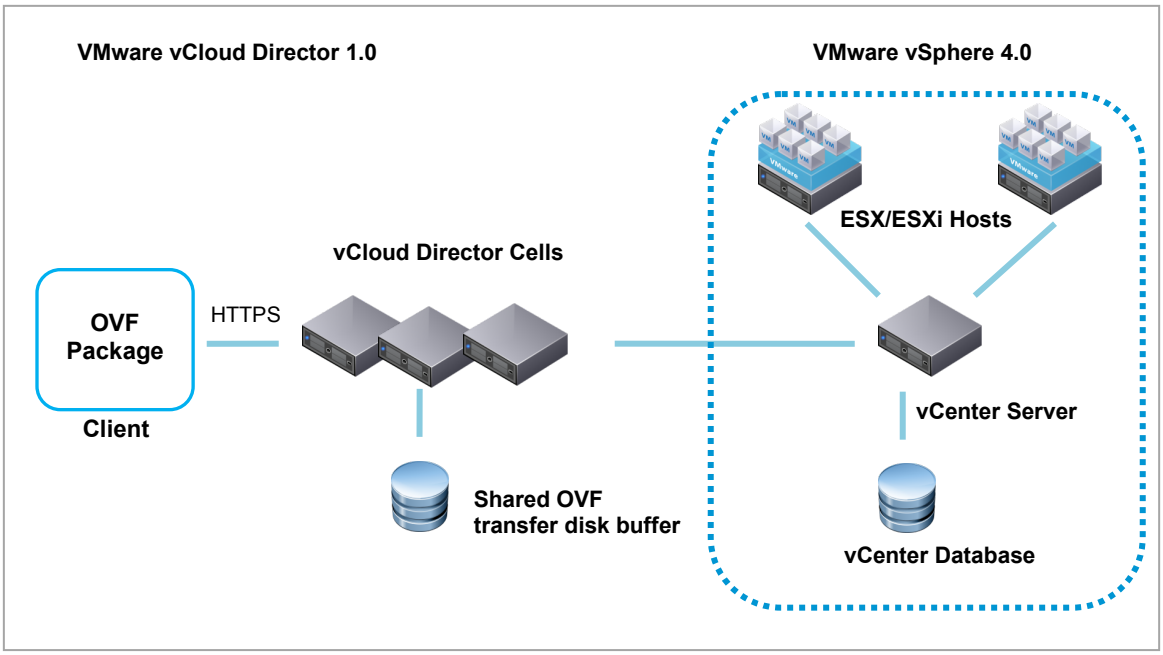

Figure 6. VMware vCloud Director High Level Architecture

Note: All the results for OVF file upload in this white paper were collected through the vCloud Director user interface.

OVF File Upload Latency for Different File Sizes

Before it can upload an OVF package, the client needs to verify the checksum of the VMDK files. After this occurs, the files in the OVF package are transferred to the vCloud Director cell. Upon the completion of file transfers from the client to the cell, the cell verifies the checksum of VMDK files again to ensure that no files were corrupted during this transfer. The last step is to upload the OVF package into one of the ESX hosts by using deployOVFPackage API calls against vCenter Server.

The majority of time is spent in client-to-cell transfer, as shown in Figure 7. For bigger files, the total latency to upload is longer.

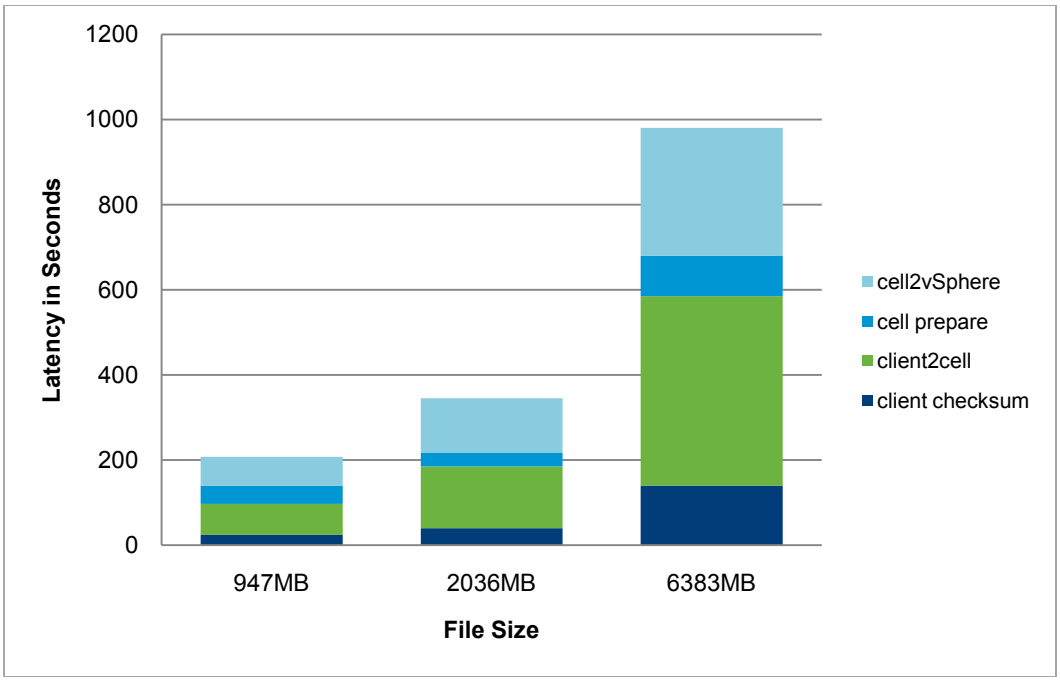

Figure 7. OVF File Upload Latency Breakdown for Different File Sizes

Concurrent OVF File Upload Latency

In a production environment, a single cell handles multiple, concurrent OVF file transfers at the same time. The performance characterization of the OVF file upload changes when the disk I/O pattern goes from sequential I/O to random I/O. This is shown in Figure 8. Comparing the first bar with the second bar, to upload two OVF files concurrently, each file transfer latency is almost doubled. The effectiveness of random disk I/O between two concurrent file transfers and three concurrent file transfers is much less significant as shown in the second bar and the third bar of Figure 8.

For this experiment, the vCloud Director server ran in a 64-bit Red Hat Linux 5 virtual machine with 4 vCPUs and 8GB RAM. The OVF file transfer location was on the same disk as the cell log files. A local disk of an ESX host served as the datastore to back up the cell VM.

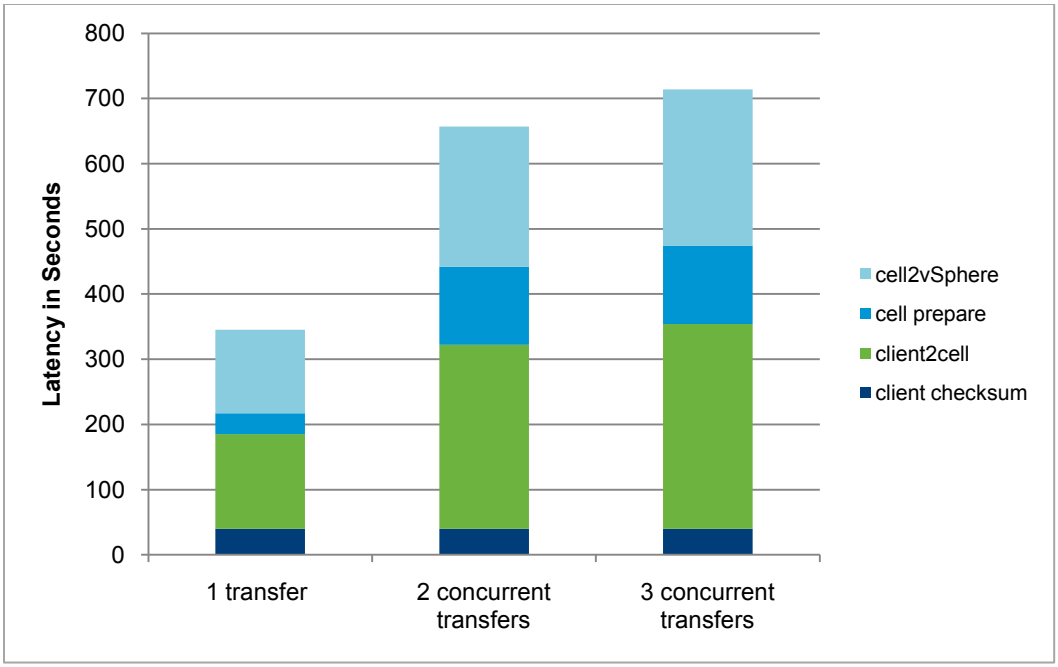

Figure 8. OVF Uploads with 1, 2, and 3 OVF Packages Transferring at the Same Time

File Upload in WAN Environment

Because OVF package upload often occurs in a WAN (Wide Area Network), it is important to show latency trends with various network bandwidths and speeds. Table 3 shows the bandwidth, latency, and error rate for three typical networks in a WAN.

Table 3. Typical Networks in a WAN

NETWORK	BANDWIDTH	LATENCY	ERROR RATE
DSL	512Kbps	100ms	0.05%
Satellite	1.5Mbps	500ms	0.10%
T1	1.5Mbps	100ms	0.05%

A WAN simulator, WANem, was used for this experiment. The layout is shown in Figure 9. WANem (Wan Area Network EMulation) is a piece of software running on Knoppix, a well-known Linux distribution based on Debian. For more information, refer to the *WANem Simulator Web site* [3].

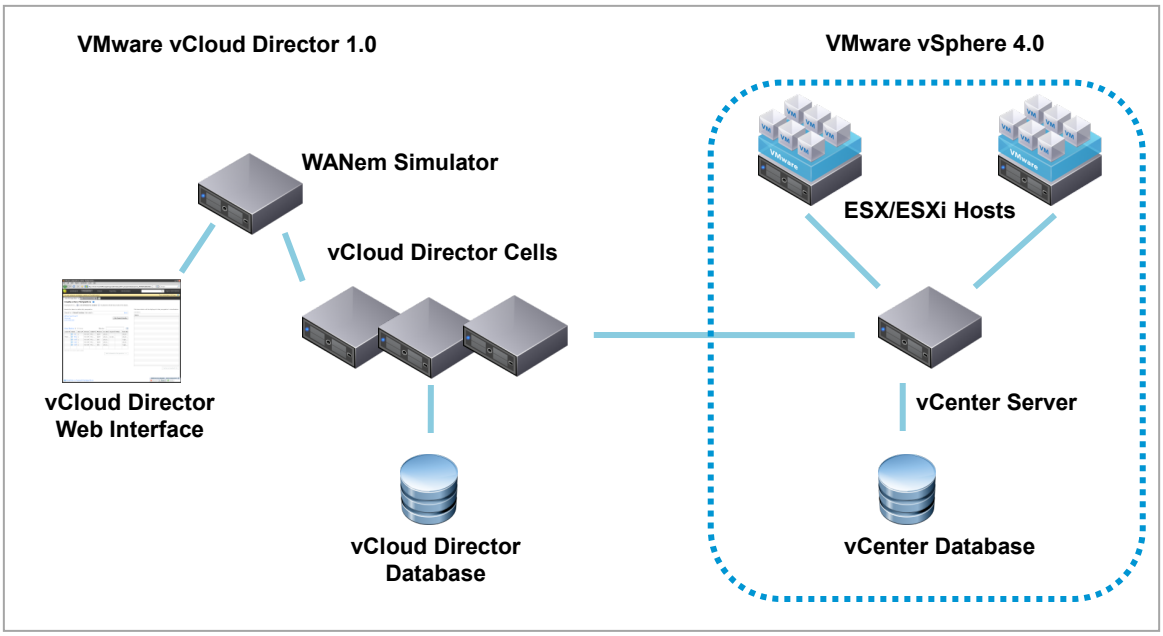

Figure 9. OVF File Upload with WANem Simulator

Compared with DSL, T1 bandwidth is much larger. The total time to upload the same 295MB size VMDK file for T1 is twice as fast as the one on DSL, as shown in Figure 10.

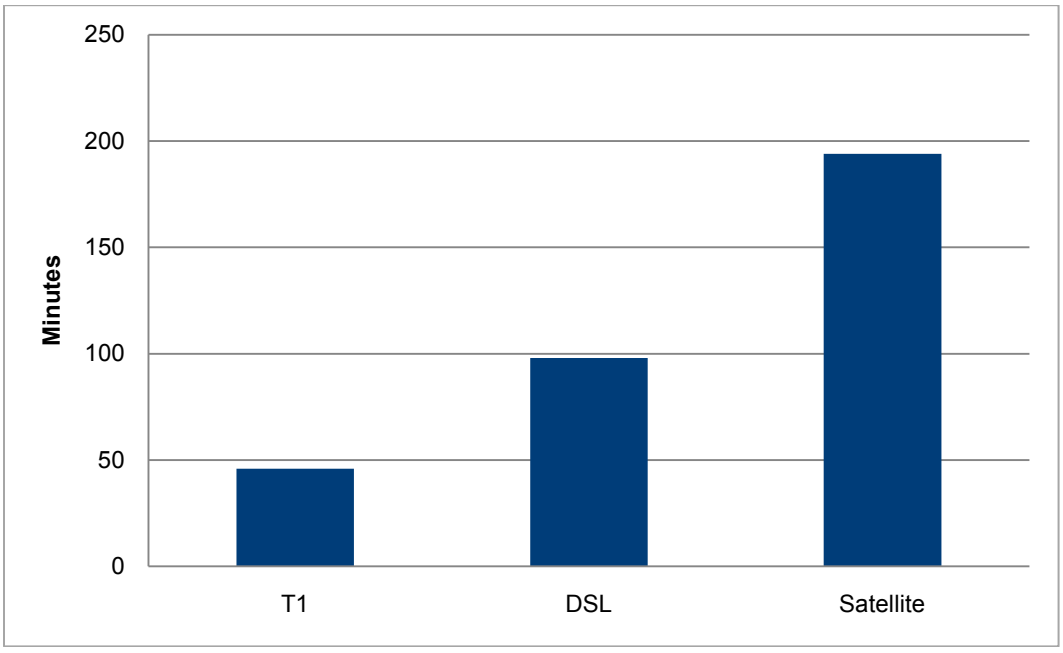

Figure 10. OVF File Upload Trend for T1, DSL, and Satellite Network

Although the bandwidth of a satellite network is 1.5Mbps, which is twice as big as DSL network bandwidth, the underlying link cannot fully utilize the bandwidth due to the TCP maximum receive window size of 64KB. The TCP receive window size determines the amount of received data that can be buffered on the receive side before the sender requires an acknowledgement to continue sending. For a sender to use the entire bandwidth available, the receiver must have a receive window size equal to the product of the bandwidth and the roundtrip delay.

Here are some examples that show how the TCP receive window size affects bandwidth utilization:

- For a DSL network:

  the bandwidth delay product = bandwidth × roundtrip delay = 512Kbps × (100ms × 2 ÷ 1000)s = 12.8KB

  Here, we take the DSL bandwidth from the previous table, 512Kbps. To get the roundtrip delay, we multiply the DSL latency from the previous table (100ms) by 2 because we need to measure the latency to and from the destination. Then we divide the product by 1000 to convert milliseconds to seconds.

  This is well within the TCP maximum receive window size of 64KB.

- For a satellite network:

  the bandwidth delay product = 1.5Mbps × (500ms × 2 ÷ 1000)s = 1.5Mb = 192KB

  This is bigger than the TCP maximum receive window size of 64KB, so 1.5Mbps is not fully utilized.

Sizing and Performance Tuning Tips

For OVF file upload, the vCloud Director server temporarily stores the OVF packages before they are transferred to ESX hosts. The required disk space for the temporary buffer is determined by two elements. One is the maximum number of concurrent OVF package uploads; the other is the file size for each OVF package. For example, if at the peak, 20 concurrent transfers are expected, and each OVF package contains about 10GB VMDK files, the required space is 20 × 10GB = 200GB. In a WAN environment, because an OVF transfer might take longer to finish, the disk requirement for OVF upload can be larger because there may be more OVF packages that are in the transfer state.

The same disk space could also be used for cloning vApps across vCenter Server instances. In that case, the disk requirement should also include how many concurrent vApp clone operations could occur and how big each vApp is. The next section describes how to achieve the best performance for cloning vApps across vCenter Server instances.

If a file upload fails in the middle of a transfer between the client and the cell, the temporary files are kept in the cell for some time. As a consideration for that, we recommend adding 20% disk space. The following is a sizing formula:

disk size = max number of concurrent transfers × average file size × 1.2

Here, *max number of concurrent transfers* includes both OVF file transfers and other file transfers caused by operations such as those initiated by cloning vApps across vCenter Server instances.

Because this disk buffer is very I/O intensive, we recommend separating it from the logging disk of the vCloud Director server. This can be done by NFS mounting. This way, the disk operations can be separated from the cell logging and OVF file uploads.

Clone vApps across vCenter Server Instances

In vCloud Director 1.0, a VM clone operation is triggered when a vApp is instantiated from a vApp template. When the underlying organization vDC does not have access to the source datastore on which the vApp template resides, the same mechanism used to upload an OVF package is used to create the vApp. A unique feature is implemented in the vCloud Director 1.0 release to improve the latency. This feature works when the underlying organization vDC has access to the source datastore of the vApp template. The following section provides more details on this topic.

Experimental Setup

The test bed consisted of two vCenter Server 4.1 instances. Each vCenter Server instance had two ESX 4.1 hosts in a cluster.

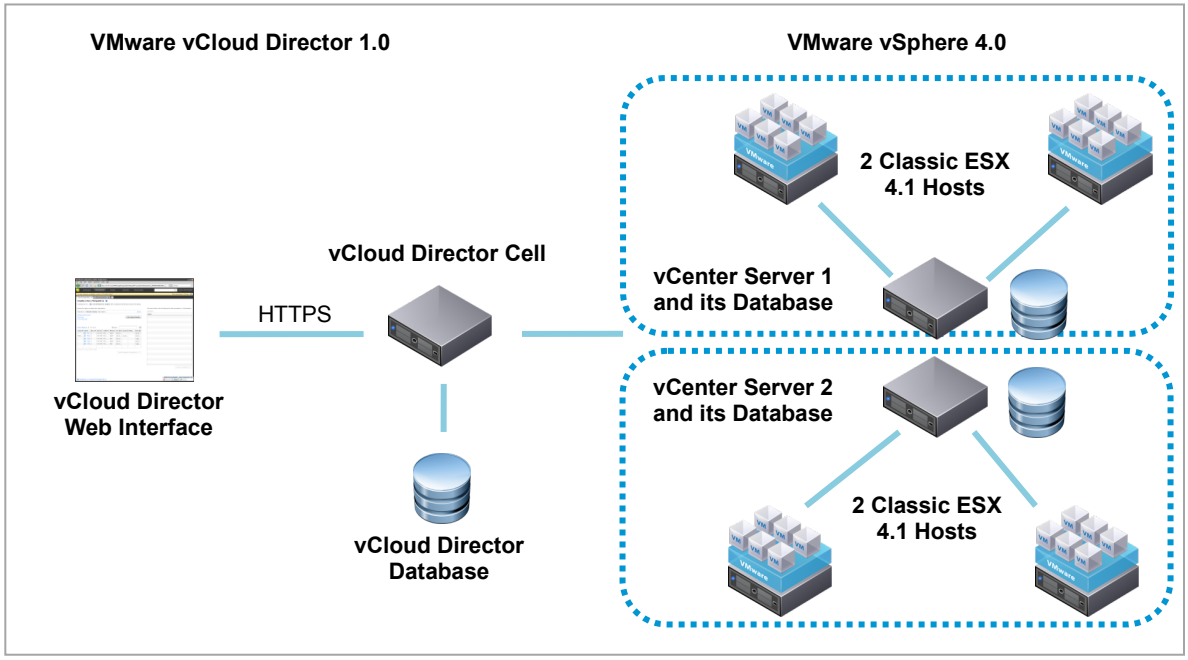

Figure 11. Testbed Setup for Inter-vCenter Clone Experiments

Two provider vDCs were created based on the two clusters from these two vCenter Server instances. Corresponding organization vDCs were derived from the two provider vDCs. These two organization vDCs were presented in the same organization. We then studied three datastore configurations to show the latency of the cloning operation. Our tests show that datastore accessibility plays a very important role in terms of latency.

Two Separated Datastores

Figure 12 shows two separated datastores.

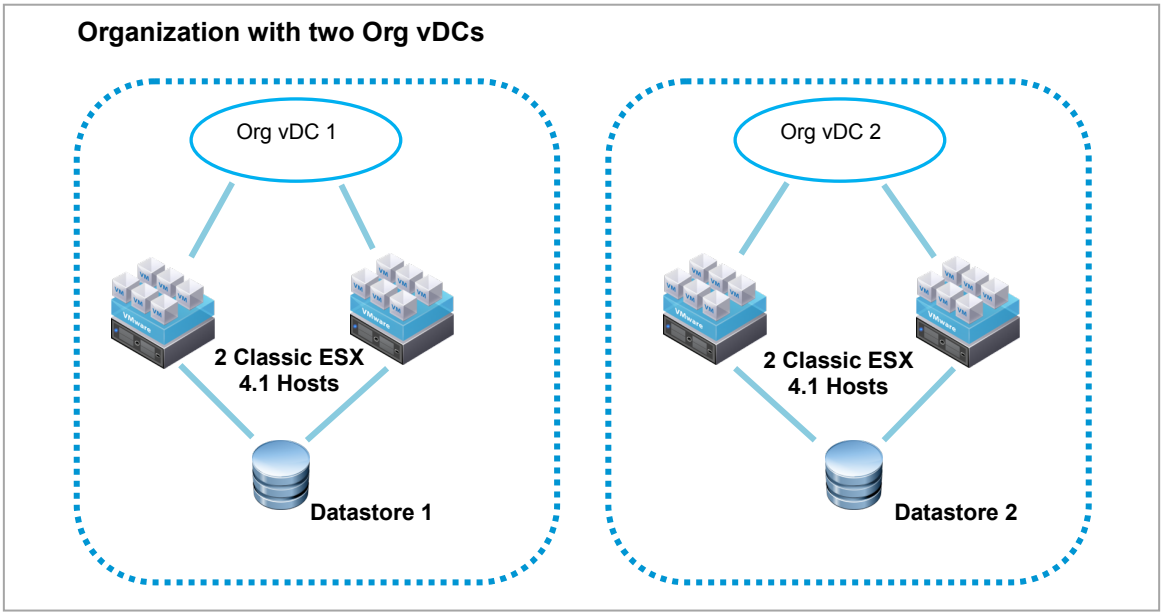

Figure 12. Two Separated Datastores

In this design, the source vApp template resides on Datastore 1 in Org vDC1. A template was cloned to Datastore 2, which was accessed by one of the two ESX hosts managed by Org vDC 2.

Because ESX hosts in Org vDC 2 did not have access to Datastore 1, the vApp template was uploaded to the cell first, and then a deployment OVF package API call was issued to the vCenter Server that was mapped to Org vDC 2.

Datastore Shared Among ESX Hosts

If ESX hosts in Org vDC 2 have access to the same datastore as shown in Figure 13, the process to upload the vApp template OVF package and then deploy it to vCenter Server can be replaced by single file copy API calls to vCenter Server. This design improves performance dramatically. We discuss this in "OVF File Upload" on page 10 with more details.

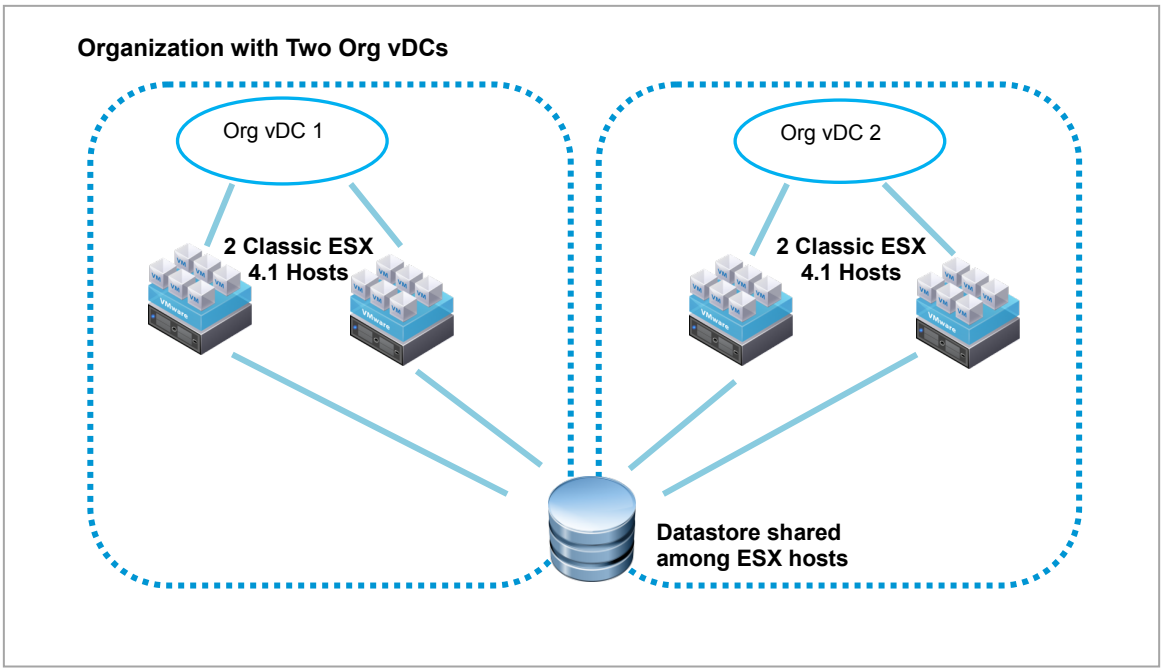

Figure 13. Datastore Shared Among ESX Hosts

Shared and Separated Datastores

Even if the destination datastore is not the same as the source datastore (as seen in Figure 14), the process to clone VMs can still be optimized. In this case, VM files are copied directly between datastores and the requirement to have the cell as the middle tier for buffering the OVF package is eliminated.

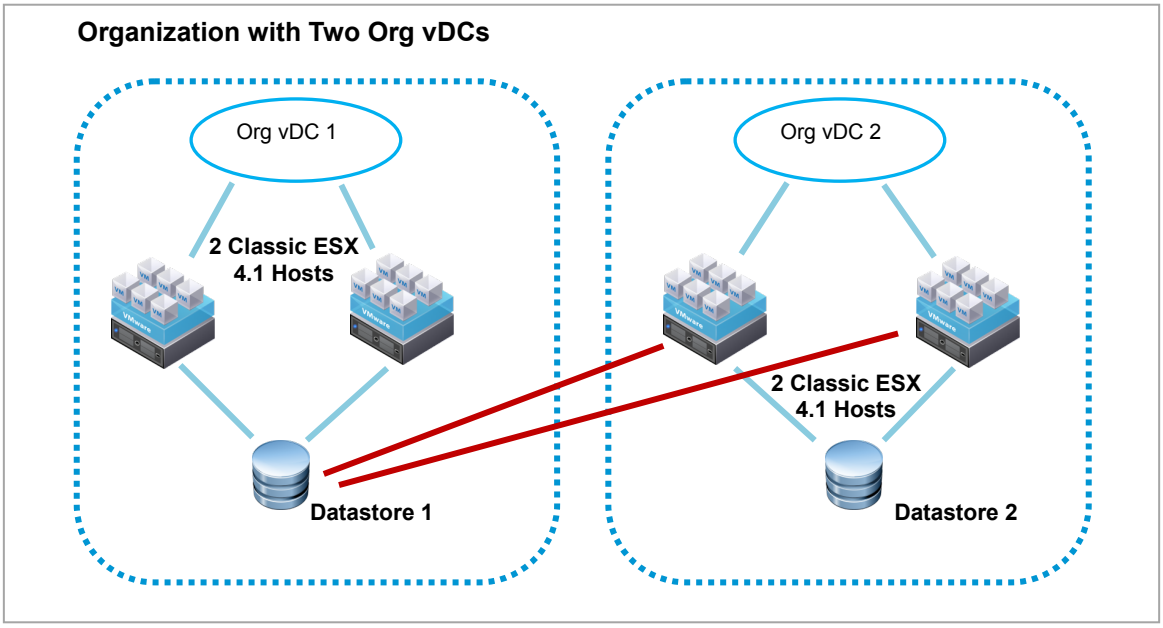

Figure 14. Shared and Separated Datastores

Results

Copying OVF packages directly between two datastores or within the same datastore is much faster than the approach of uploading OVF packages. Figure 15 shows the latency trends for the three configurations.

The last bar is for Figure 12, where the destination organization vDC does not have access to the source datastore on which the source vApp template resides. At almost eight minutes (488 seconds), this setup has the highest latency. In contrast, the first and second bars (for the configurations shown in figures 13 and 14) show a large improvement at close to 1 minute.

VM clones can be completed eight times faster when the datastore setup is correctly designed.

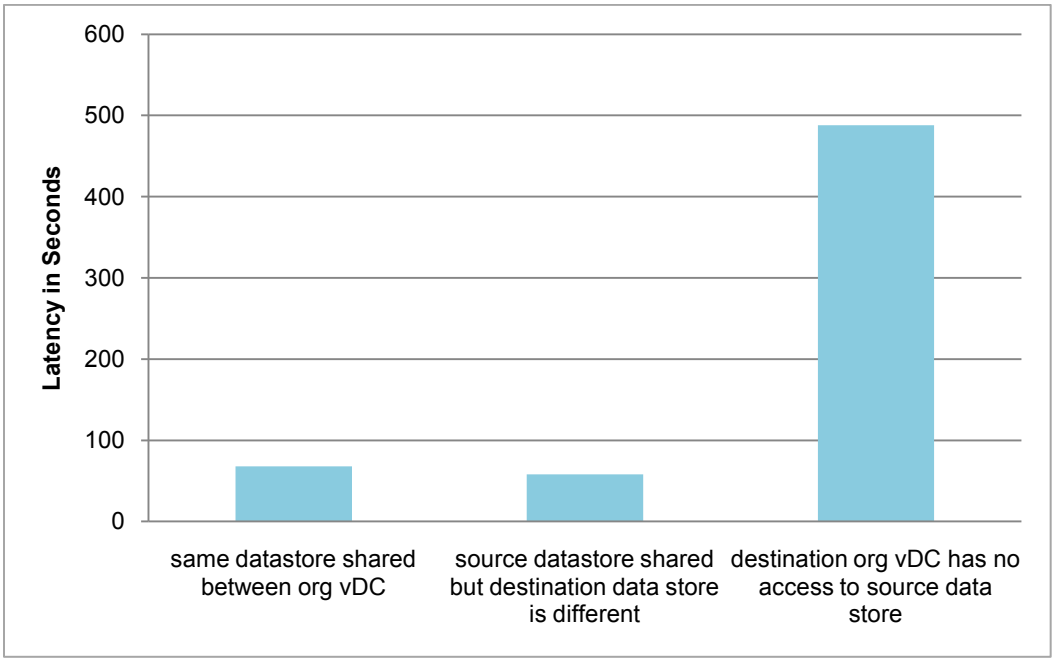

Figure 15. VM Clone Latency for Different Datastore Setups

For the first two bars in Figure 15, the latency numbers are almost the same as the ones achieved directly through vSphere. This shows that the additional time required by vCloud Director is negligible for VM clone operations when compared with the time taken to clone VMs in vSphere.

Performance Tuning Tips

To achieve better performance for VM clones in vCloud Director 1.0, we recommend having a central datastore to hold the most popular vApp templates and media files and have this datastore mounted to at least one ESX host per cluster. This way, the destination organization vDC has access to both the source and destination datastores. This removes the need to copy the OVF package files twice as discussed in "OVF File Upload" on page 10.

Deploy vApp

In vCloud Director 1.0, vApps can be deployed with fence mode or without fence mode.

When a vApp is deployed in fence mode, the connections between the vApp and the organization network traverse a vShield™ Edge virtual appliance, which provides protection for the vApp network and also extends the organization network so that identical virtual machines can be run in different vApps. By extending the organization network in this way, it is possible to run multiple, identical vApps without conflict—the vShield Edge appliance is deployed on a per-vApp network basis, isolating the overlapping Ethernet MAC and IP addresses.

Figure 16 illustrates the difference between a vApp that is deployed with and without fence mode.

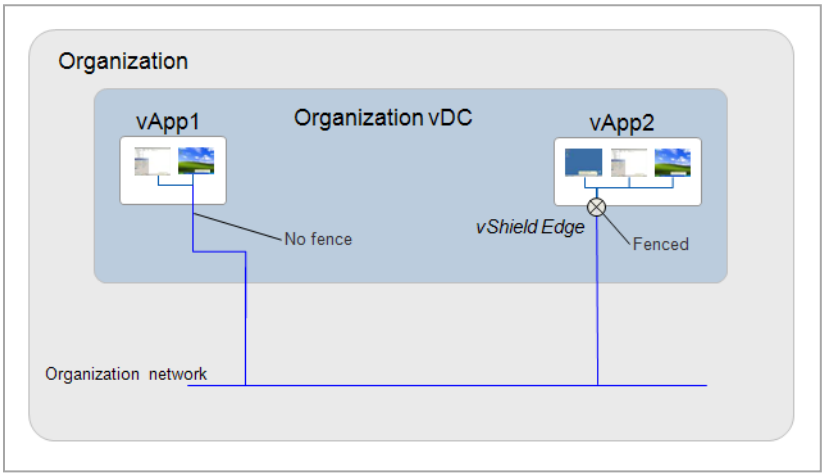

Figure 16. vApps Deployed with and without Fence Mode

In this section, we compare the performance of deploying vApps with and without fence mode.

Deploy Single vApp with Varying Number of VMs

First, we compare the performance when a single vApp with a varying number of VMs is deployed and powered on.

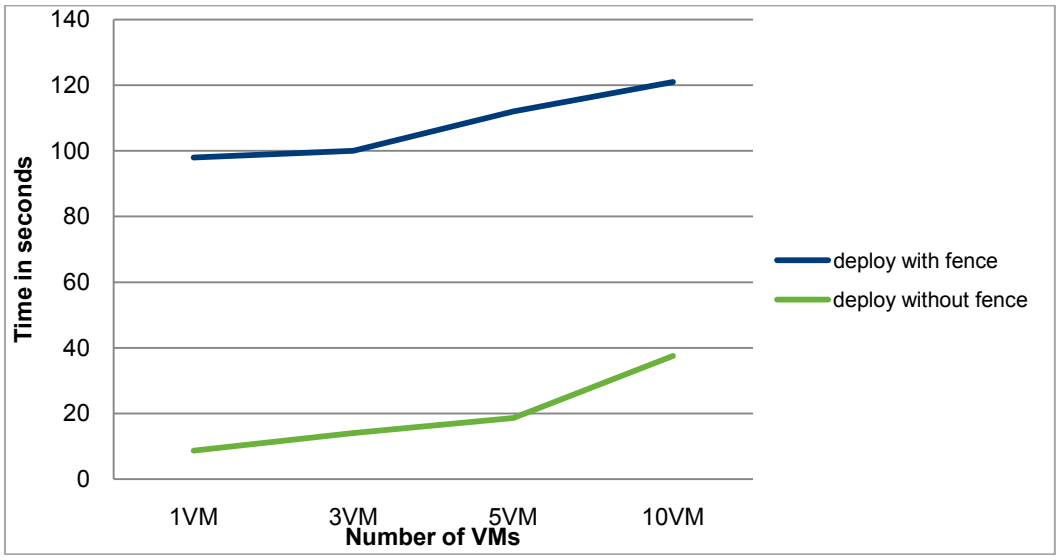

Figure 17. Deploy a Single vApp with and without Fence Mode

As shown in Figure 17, when compared with deploying a vApp without fence mode, deploying a vApp with fence mode takes a longer time. This is because vCloud Director needs to perform extra configuration in order to deploy a vApp with fence mode. More detail is shown in Figure 18. Before powering on the vApp, vCloud Director needs to deploy and configure a vShield Edge VM as the gateway, which takes more than 80 seconds.

If the vApp is deployed without fence mode, only the last two steps (reconfigure VMs and power them on) are required.

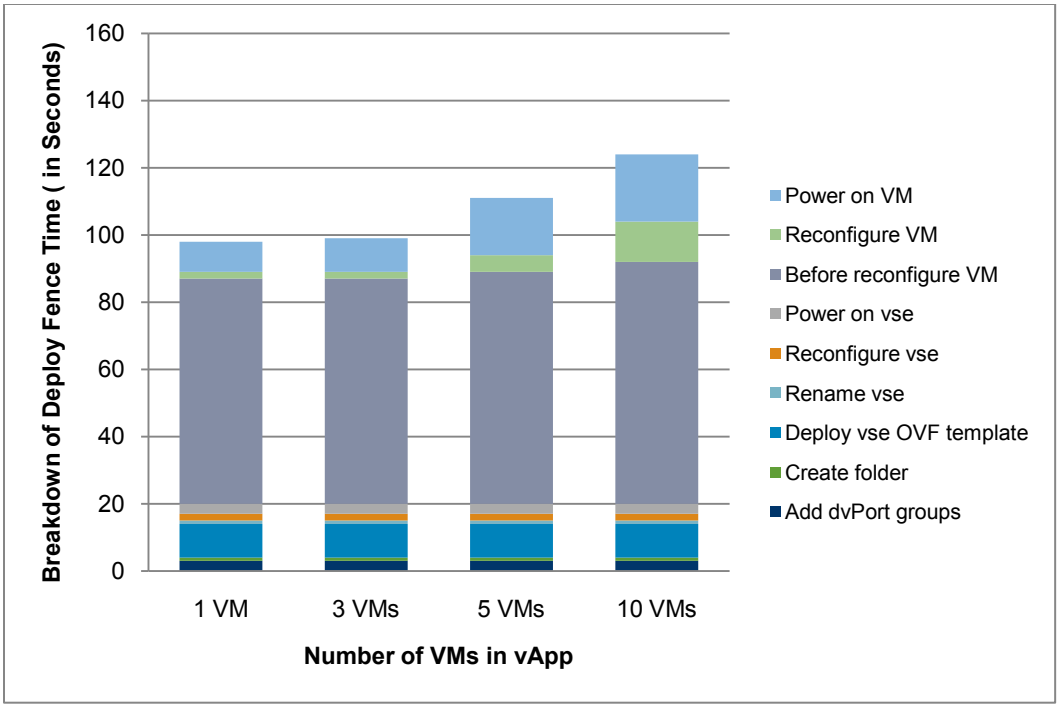

Figure 18. Deploy a Single vApp with Fence Mode: Operation Breakdown

Figure 18 also shows that deploying vApps with an increasing number of VMs increases the deployment time increases only slightly. This is because only one vShield Edge instance is required, even with multiple VMs in a vApp.

Concurrently Deploy vApps with and without Fence Mode

In this section, we compare the performance when we concurrently deploy and power on vApps with and without fence mode.

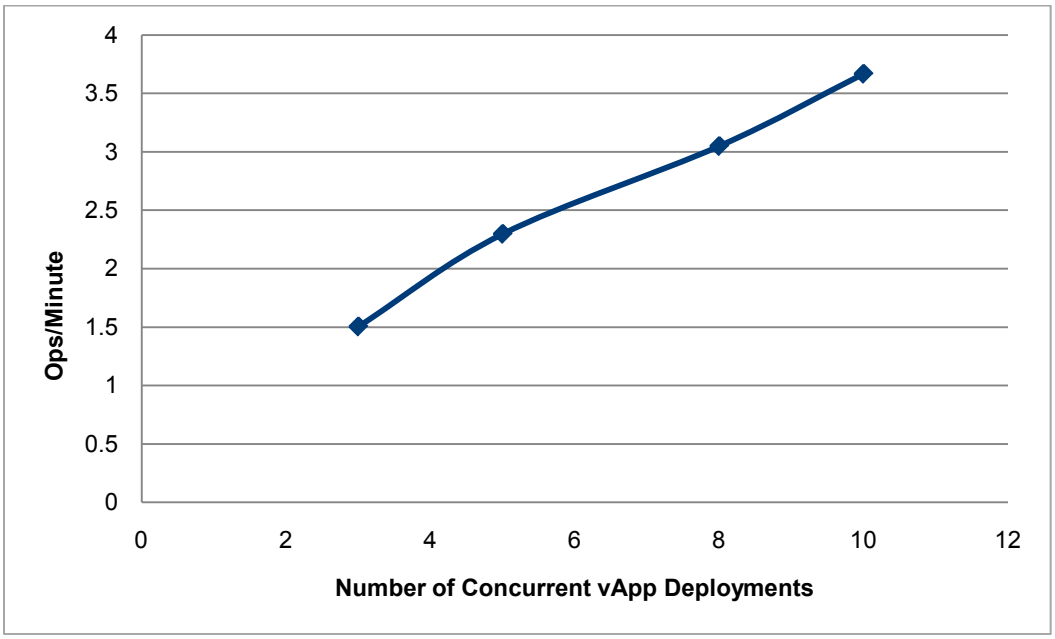

Figure 19. Concurrently Deploy vApps with Fence Mode

For fence mode, because each operation takes a longer time, the throughput is not very large when the concurrency level remains low. However, the throughput keeps increasing when the concurrency level increases. This means that when vApp deployments are performed concurrently, there is no significant performance degradation.

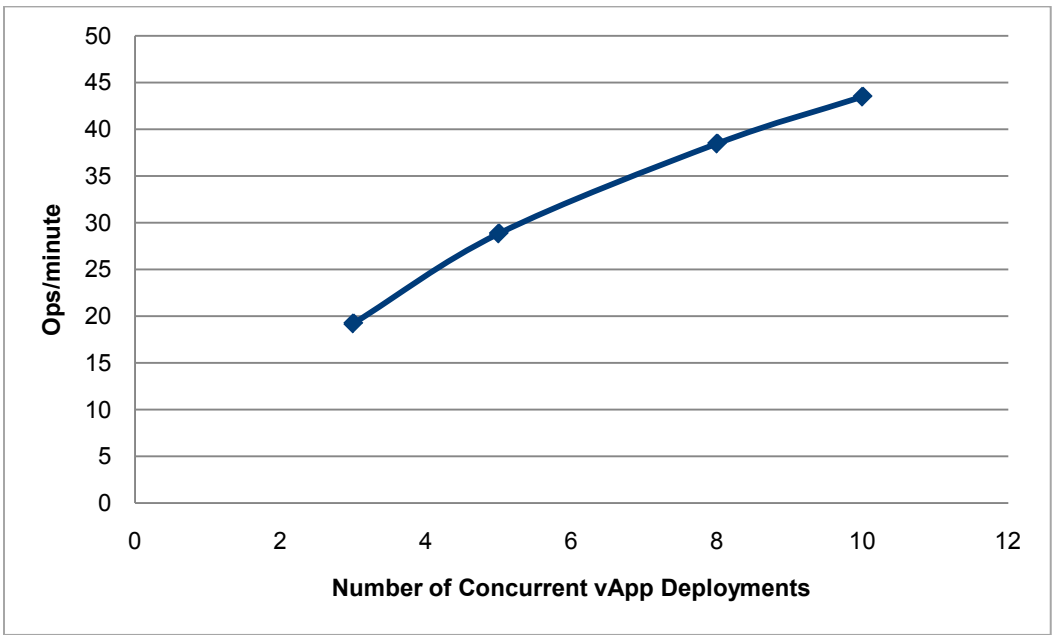

Figure 20. Concurrently Deploy vApps without Fence Mode

For deploying without fence mode, the throughput is higher than with fence mode because less configuration needs to occur. Here, too, throughput keeps increasing when the concurrency level increases.

Performance Tuning Notes

The latency to deploy a vApp in fence mode does not increase much when there are more VMs in the vApp. This is because fence mode adds a fixed cost to the deployment time (deploy and configure a vShield Edge VM one time) and the increase in time is not proportional to the number of VMs in the vApp.

Deploying multiple vApps concurrently can achieve high throughput.

Inventory Sync

When a vCloud Director cell is initialized, the current vCenter Server inventory information is retrieved and the relevant data is stored in the empty vCloud Director database. This process is called **initial sync**. The vCloud Director server may be shut down and restarted. When it is restarted, all the current vCenter Server inventory information is retrieved again. Changes to the inventory data are stored in the vCloud Director Database. This process is called **restart-cell-sync**. The difference between initial sync and restart-cell-sync is the state of the vCloud Director database. Initial sync starts with an empty vCloud Director database, whereas restart-cell-sync starts with a database that already contains the inventory information. After restart-cell-sync, the vCloud Director database is periodically updated.

The vCenter Server process may also be shut down and restarted. When this occurs, the vCloud Director server tries to reconnect to the vCenter Server and re-sync the inventory information. This process is called **reconnect-vCenter-sync**. Reconnect-vCenter-sync takes less time than restart-cell-sync because, in reconnect-vCenter-sync, the vCloud Director server has a cache that stores the inventory information in memory.

Sync Latency for Different Inventory Sizes

Figure 21 shows that the latencies for initial sync and restart-cell-sync increase linearly as the number of virtual machines in the system grow. For reconnect-vCenter-sync latency, the default inventory cache is not big enough to hold 3000 inventory objects, so some inventory objects must be fetched from the vCloud Director database. This process causes the significant latency increase seen between 2000 and 3000 inventory sizes.

For the restart-cell-sync experiment, there are no changes for the inventory objects, and so there are no database inserts required. In contrast, the latency for initial sync is higher because inventory objects need to be inserted into the vCloud Director database. The latency for initial sync can be longer than the latency for restart-cell-sync even when the inventory size is relatively small. The majority of time is spent comparing the difference between the stored inventory object in the vCloud Director database with the one in vCenter Server.

For reconnect-vCenter-sync, because the in-memory inventory cache can potentially hold all or most of the inventory objects, no extra time is needed to fetch these objects from the vCloud Director database. This is why reconnect-vCenter-sync gives the best performance among the three types of inventory syncs. The following sections describe more about the inventory cache effects and how to tune the inventory cache for better performance.

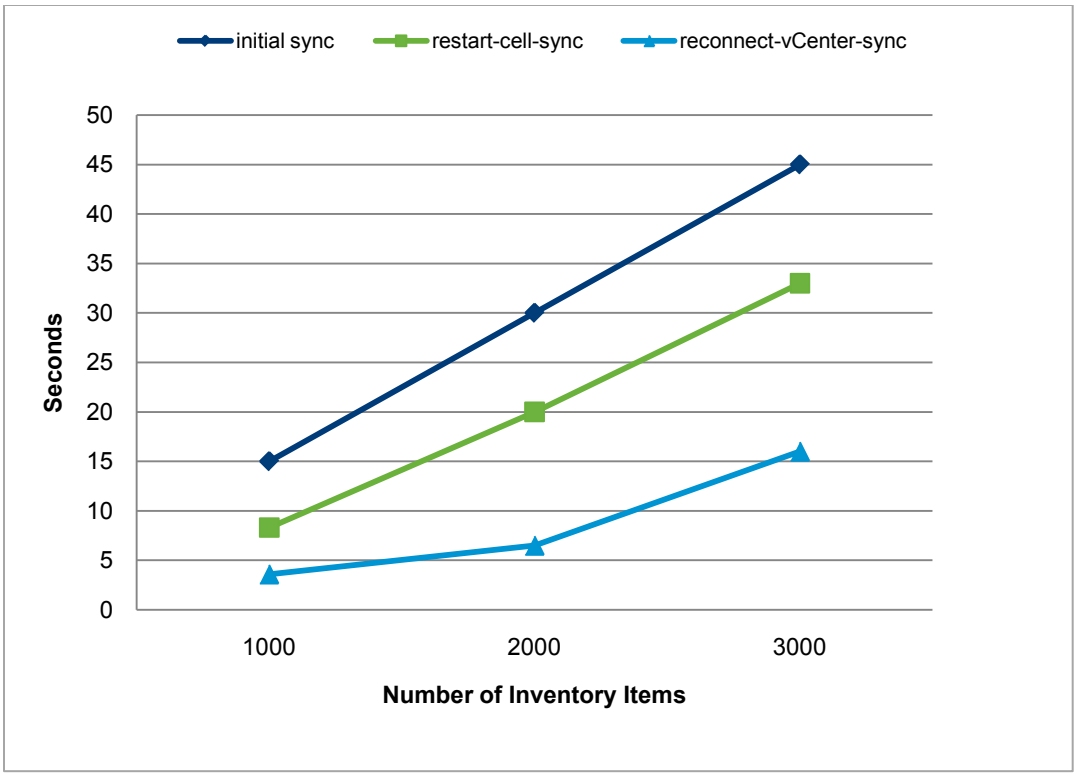

Figure 21. Inventory Sync Latencies

In-Memory Inventory Cache

An in-memory inventory cache is implemented in vCloud Director 1.0. The cache saves the time to fetch inventory information from the database and the time to de-serialize the database record. Figure 22 demonstrates the effectiveness of the inventory cache for reconnect-vCenter-sync usage. When the cache size is set to 10,000 inventory items, the cache-hit ratio[1] is much higher. The latency to sync 5000 VMs is also much faster when the cache-hit ratio is higher.

[1] Cache-hit ratio is the percentage of accesses to the cache that result in hits.

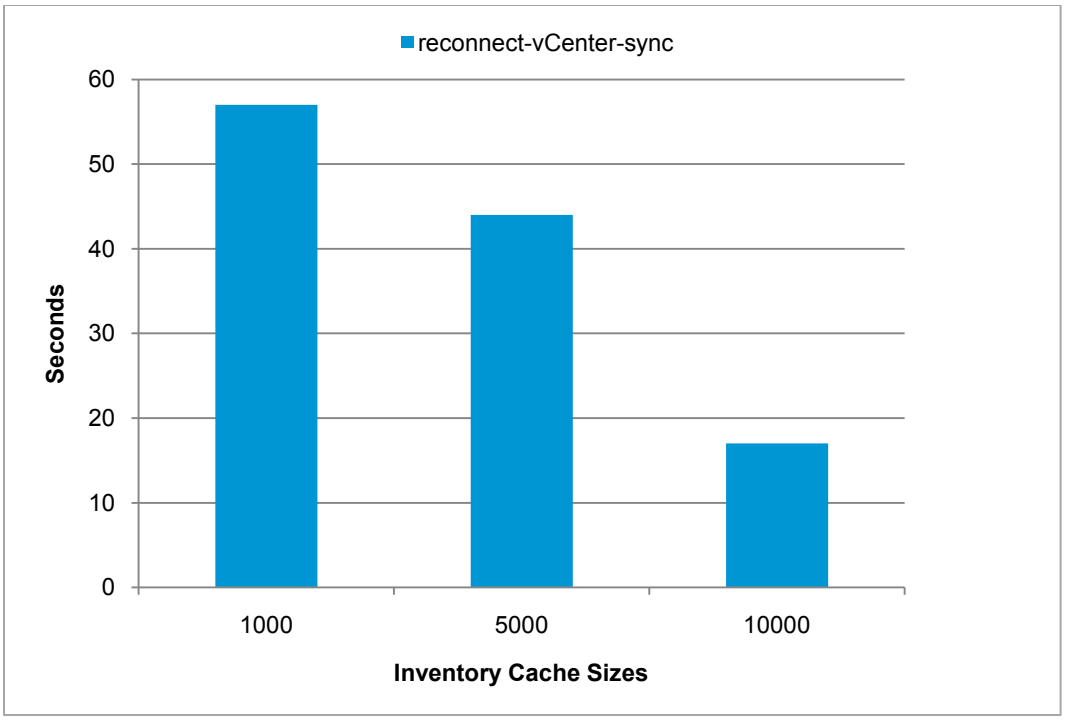

Figure 22. Sync Time for Inventory Cache Sizes Set to 1000, 5000 and 10,000 with 5000 VMs

Inventory Cache and JVM Heap Size Tuning

vCloud Director maintains an inventory of all vCenter Server entities it manages. A part of this inventory is cached by each vCloud Director cell. This caching process depends on the size of the cache configured for each cell. There is one sizing configuration for inventory cache for all cells of the vCloud Director installation. Serialization and deserialization of vCenter Server entities and querying the database for these objects are expensive. So, the overall performance of vCloud Director is greatly enhanced by proper sizing. This section describes how this sizing matters and how inventory cache prioritizes objects.

By default, each vCloud Director cell is configured for 5000 inventory items (total inventory cache entries including hosts, networks, folders, resource pools, and so on). We estimate that this sizing is optimal for 2000 VMs. Our measurements show that on average, each inventory cache item is about 30KB in size. So, with the default configuration, an installation supporting 2000 VMs may end up using 5000 × 30KB = ~150MB.[2]

Note: For steps to change the inventory cache, see "Adjusting Thread Pool and Cache Limits" on page 28.

Because the number of supported VMs increases in the vCD when additional vCloud Director cells are added, proper tuning of this inventory size will help boost performance. We recommend the following formula to help figure out what number to use for the cache size on each cell:

Inventory Cache Size = 2.5 × (Total Number of VMs in vCloud Director)

We assume here that most VMs in one or more instances of vCenter Server that are managed by vCloud Director are the ones created by vCloud Director. If that is not the case, substitute the *Total number of VMs in vCloud Director* with *Total number of VMs in each vCenter Server*.

As the number of VMs managed by vCloud Director increases, you can either add new vCenter Server instances to support more VMs, or you can utilize vCenter Server to its full scale (3000 VMs for vSphere 4.0 and 10,000 VMs for vSphere 4.1). Because the inventory in vCloud Director is the aggregate of all vCenter Server inventories

[2] 5000 × (30KB) = 5000 × (30 × 1024) bytes = 153,600,000 bytes ÷ 1,048,576 = 146MB

managed by it, the above sizing formula still holds.

An additional detail that you should pay attention to is the JVM heap size for each vCloud Director cell process. As mentioned previously, an increase in the total number of inventory items cached requires more memory on each cell. So, it is essential to increase the JVM heap equivalently. By default, JVM heap size is configured to be 1GB for the process. This is good for supporting 5000 inventory cache entries. If the number of cached entries is changed, for example, to 15,000 (an additional 10,000) for a large vCloud Director installation, you might need to increase the JVM heap size to support additional cache entries. In this example, the JVM heap size should be increased to 1.5GB (1GB original + 10,000 x 30KB = 300MB + 200MB as some buffer for other caches/objects with an increase in total VMs) to support the additional 10,000 cache entries per vCloud Director cell.

For large installations, if the JVM heap size is not increased, it is likely that the vCloud Director cell will run out of memory, resulting in poor performance and a possible cell failure. For more information, refer to knowledge base article 1026355 to change vCloud Director JVM Heap size [9].

Inventory Sync Resource Consumption

During inventory sync, up to 25% out of 4 vCPUs can be consumed because inventory sync is implemented as a single thread task. For the vCloud Director database, reconnect-vCenter-sync consumes the least CPU among all three types of syncs because most objects are held in the inventory cache, and there is no need to query and insert the backend database. For initial sync and restart-cell-sync, the Oracle database consumes less than 20% of 4 vCPUs. As we have discussed, most of the CPU usage is consumed by database inserts in the initial sync case. However, for restart-cell-sync, most of the CPU consumed is for database queries.

All three types of syncs depend on vCenter Server to provide the inventory information. As the number of inventory objects gets bigger, the resource consumption in vCenter Server is more significant. Figure 23 shows vCenter Server CPU usage for reconnect-vCenter-sync with a different number of VMs.

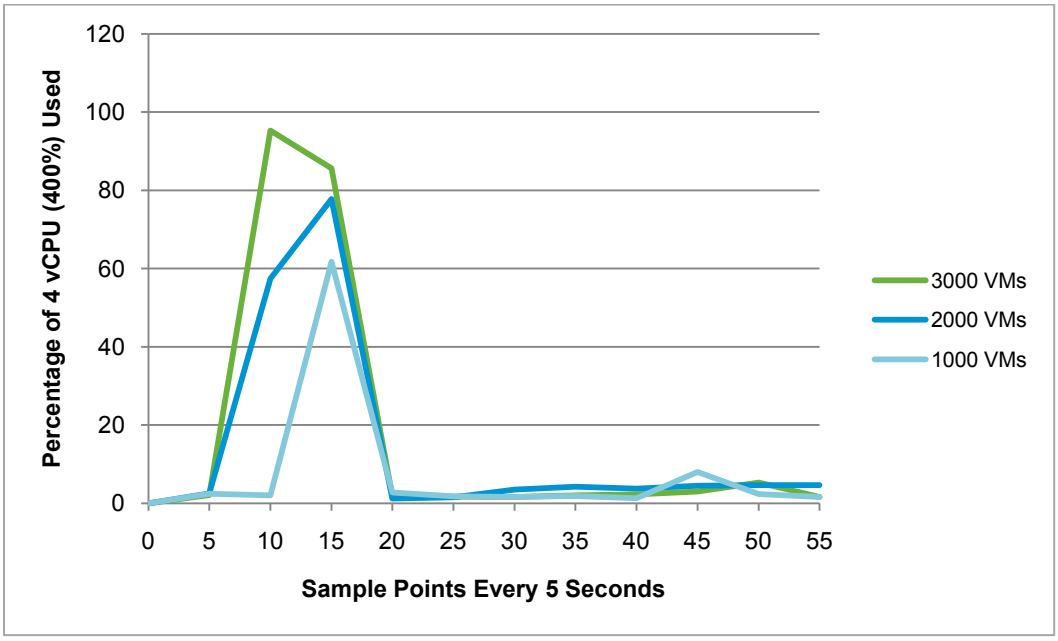

Figure 23. vCenter CPU Usage for Reconnect-vCenter-Sync with 1000, 2000 and 3000 VMs

Load Balancing VC Listener

vCloud Director has a service called VC Listener (Inventory Service) through which it monitors and aggregates changes of vCenter Server entities. Each VC Listener service is capable of listening for changes from one or more vCenter Server instances. However, there is only one VC Listener service that is listening to a given vCenter Server. vCloud Director manages which cell will listen for changes to which vCenter Server. If any cell that was listening to vCenter Server changes fails, vCloud Director automatically picks another live cell to continue listening for changes.

By default, vCloud Director load balances evenly among available live cells when failure occurs. However, vCloud Director does not have automatic load balancing when things are working fine and new cells are added to an existing cluster or failed cells are brought online.

For example, consider a vCloud Director installation with two cells, A and B, and two vCenter Server instances (vCenters), VC1 and VC2. By default, vCloud Director tries to assign each cell one vCenter to listen for changes. Suppose cell A gets assigned to VC1 and cell B to VC2. When cell B fails, VC2 is failed over to cell A. Now if cell B is brought back online, cell A continues to listen for both vCenters and vCloud Director does not rebalance the load. This does not mean that all interactions with VC2 or VC1 have to go through cell B. Cell B will be equally functional, but cell A will be more loaded in terms of CPU usage.

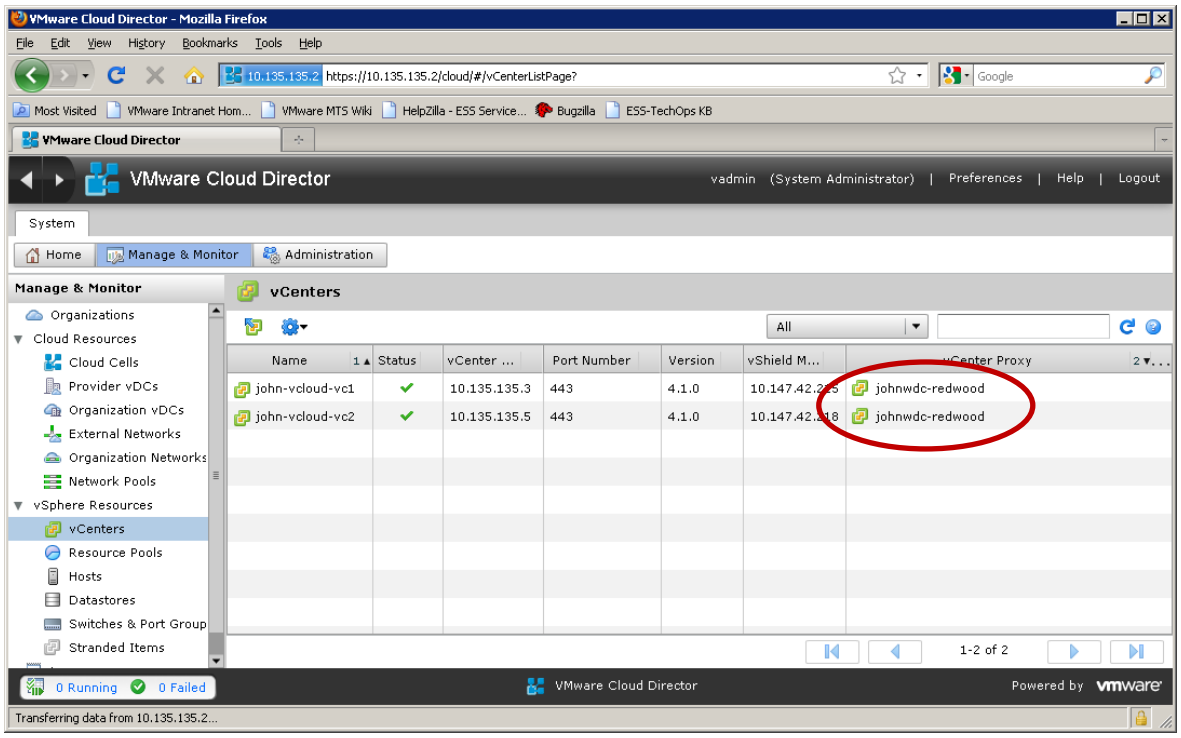

Figure 24. Detection vCenter Listener Location in vCloud Director User Interface

Very large installations of vCloud Director (greater than or equal to 10,000 VMs) are likely to have more than one vCloud Director cell and one vCenter. Under this circumstance, it is essential to monitor how many vCenters are managed by each cell and determine if there is any room on other cells. You can view the assignments of vCenters to cells on the vCenters list page, as shown in Figure 24.

vCloud Director has functionality that allows you to manually load balance existing load. In the previous example, by manually reconnecting any one vCenter (that is, VC1 or VC2) from the vCenters list page, vCenter Director will reassign the vCenter to cell B since it had no assigned vCenter to which to listen.

vCloud Director considers all cells to be the same in terms of physical resources available.

Note: Reconnecting a vCenter will incur some down time for actions performed against that particular vCenter, so we recommend you reconnect the vCenter when overall vCloud Director activity is low. It usually takes between 5-30 seconds to move the VC Listener from one cell to another because vCloud Director has to disconnect from the current cell and connect back from a different cell. This process will not affect currently running VMs in any way; they will continue to function without any interruption.

Adjusting Thread Pool and Cache Limits

A vCloud Director installation has preconfigured limits for concurrent running tasks, various cache sizes, and other thread pools. These all are configured with good default values, which are tested to work within an environment of 10,000 VMs. Some of these numbers are also configurable in case you want to adjust them. You must restart the cell for the changes to go into effect.

Table 4. Thread Pool Limits

THREAD POOL	DEFAULT SIZE (FOR EACH CELL)	USAGE/INFORMATION	HOW TO ADJUST
Tasks	128	Maximum number of concurrent tasks that can be executed per cell of a vCloud Director installation. This is a global task count and is not scoped per user or organization. Different cells of the same vCloud Director installation can have different values.	org.quartz.threadPool.threadCount = N Where N is some number of concurrent tasks you want to run on a given cell.
VM Thumbnails	32	Maximum number of concurrent threads that can fetch VM thumbnail images from vCloud Director Agent running on an ESX host. Only thumbnail images for running (powered on) VMs are collected. Thumbnails are also retrieved in batches, so all VMs residing on the same datastore or host will be retrieved in batches. vCloud Director only fetches thumbnails if they are requested and once fetched, also caches them. Thumbnails are requested when a user navigates to various list pages or the dashboard that displays the VM image.	Not configurable.

Table 5. Cache Limits in vCloud Director 1.0

CACHE	DEFAULT SIZE (FOR EACH CELL)	USAGE/INFORMATION	HOW TO ADJUST
VM Thumbnail Cache	1000	Maximum number of VM thumbnails that can be cached per cell. Each cached item has a time to live (TTL) of 180 seconds.	cache.thumbnail.maxElementsInMemory = N cache.thumbnail.timeToLiveSeconds = T cache.thumbnail.timeToIdleSeconds = X
Security Context Cache	500	Holds information about the security context of logged in users. Each item has a TTL of 3600 seconds and idle time of 900 seconds.	cache.securitycontext.maxElementsInMemory = N cache.securitycontext.timeToLiveSeconds = T cache.securitycontext.timeToIdleSeconds = X
User Session Cache	500	Holds information about the user sessions for logged-in users. Each item has a TTL of 3600 seconds and idle time of 900 seconds.	cache.usersessions.maxElementsInMemory = N cache.usersessions.timeToLiveSeconds = T cache.usersessions.timeToIdleSeconds = X
Inventory Cache	5000	Holds information about vCenter Server entities managed by vCloud Director. Each item has a LRU (least recently used) policy of 120 seconds.	inventory.cache.maxElementsInMemory = N

In order to adjust any of these configured values, perform the following steps:

1. Stop the cell. This releases the lock that the cell has on the file you will edit.
2. Edit the `global.properties` file, which is usually found in `<vcloud director install directory>/etc/`.
3. Add the desired configuration lines. For example, `org.quartz.threadPool.threadCount = 256`.
4. Save the file.
5. Start the cell.

vCenter Server configuration limits are very important because vCloud Director utilizes vCenter Server for many operations. For vCenter Server 4.0 configuration limits, refer to *VMware vCenter Server 4.0 Configuration Limits* [4]. For vCenter Server 4.1 configuration limits, refer to *VMware vCenter Server 4.1 Configuration Limits* [5].

Conclusion

In this paper, we discuss some of the features of the vCloud Director 1.0 release, performance characterizations including latency breakdown, latency trends, resource consumption, sizing guidelines and hardware requirements, and performance tuning tips. The following are some of the vCloud Director best practices for performance presented in the paper:

- Ensure the inventory cache size is big enough to hold all inventory objects.
- Ensure the JVM heap size is big enough to satisfy the memory requirement for the inventory cache and memory burst so the vCloud Director server does not run out of memory.
- Import LDAP users by groups instead of importing individual users one by one.
- Ensure the system is not running LDAP sync too frequently. The vCloud database is updated at regular intervals.
- In order to help load balance disk I/O, separate the storage location for OVF uploads from the location of vCloud Director server logs.
- Have a central datastore to hold the most popular vApp templates and media files and have this datastore mounted to at least one ESX host per cluster.
- Be aware that the latency to deploy a vApp in fence mode has a static cost and does not increase proportionately with the number of VMs in the vApp.
- Deploy multiple vApps concurrently to achieve high throughput.
- For load balancing purposes, it is possible to move a VC Listener to another vCloud Director instance by reconnecting the vCenter Server through the vCloud Director user interface.

References

[1] Open Virtualization Format Specification
http://www.dmtf.org/standards/published_documents/DSP0243_1.0.0.pdf

[2] Open Virtualization Format White Paper
http://www.dmtf.org/standards/published_documents/DSP2017_1.0.0.pdf

[3] WANem Simulator Web site
http://wanem.sourceforge.net/

[4] VMware vCenter 4.0 Configuration Limits
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_config_max.pdf

[5] VMware vCenter 4.1 Configuration Limits
http://www.vmware.com/pdf/vsphere4/r41/vsp_41_config_max.pdf

[6] VMware vCenter 4.1 Performance and Best Practice
http://www.vmware.com/files/pdf/techpaper/vsp_41_perf_VC_Best_Practices.pdf

[7] vCloud API Programming Guide
http://www.vmware.com/pdf/vcd_10_api_guide.pdf

[8] vCloud Director 1.0 Release Note
http://www.vmware.com/support/vcd/doc/rel_notes_vcloud_director_10.html

[9] Changing vCloud Director Java heap size to prevent java.lang.OutOfMemoryError messages
http://kb.vmware.com/kb/1026355

[10] vCloud Director Installation and Configuration Guide
http://www.vmware.com/pdf/vcd_10_install.pdf

About the Authors

John Liang is a Staff Performance Engineer at VMware. Since 2007, John has been working as a technical lead for performance projects on VMware products such as VMware vCloud Director, VMware vCenter Update Manager, VMware vCenter Site Recovery Manager (SRM), VMware vCenter Converter, and VMware vCenter Lab Manager. Prior to VMware, John was a Principal Software Engineer at Openwave Systems Inc., where he specialized in large-scale directory development and performance improvements. John received a M.S. degree in Computer Science from Stony Brook University.

Ritesh Tijoriwala is a Senior Member of Technical Staff at VMware. Ritesh is a lead engineer for the Performance and Scalability for VMware vCloud Director and a key engineer for Interoperability with vCenter, Failover and Inventory related functionality for VMware vCloud Director. Ritesh has received his M.S. in Computer Science from California State University, Fresno.

Xuwen Yu is a Member of Technical Staff at VMware where he works on several projects including VMware vCloud Director, VMware vCenter Update Manager and ESX host simulator. Xuwen received his Ph.D. in Computer Science and Engineering from the University of Notre Dame in 2009.

Qiang Ma is a Member of Technical Staff at VMware where he mainly works on VMware vCloud Director and VMware vCenter Lab Manager. Qiang received his M.S. degree in Computer Engineering (ECE) from Stony Brook University.

Acknowledgements

We want to thank *Rajit Kambo* and *Jennifer Anderson* for their valuable guidance and advice. Their willingness to motivate us contributed tremendously to this white paper. We would also like to thank *Andrew Dong*, *Eddie Dinel*, *Hang Cheng*, *Stephen Evanchik*, *Ravi Soundararajan*, *Reza Taheri*, *Borislav Deianov*, *Paul Herrera*, *Mahdi Ben Hamida*, *Snorri Gylfason*, *Bhavesh Mehta*, *Dushyanth Nataraj*, *David Baldwin*, *Kaushik Banerjee*, and *Jason Carolan*. Without their help, this white paper would not have been possible. Finally, we would like to thank *Julie Brodeur* for help editing and improving this white paper.

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