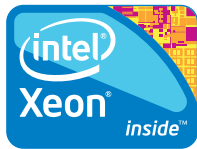


Intel® Virtualization Technology FlexMigration Enablement

Allowing virtualization software vendors to deliver live migration solutions across different generations of Intel® Xeon® processor-based servers



EXECUTIVE SUMMARY

Intel® Virtualization Technology FlexMigration (Intel® VT FlexMigration) allows virtualization software vendors to deliver live migration solutions across different generations of Intel® Xeon® processor-based servers. By dramatically enlarging the generation of servers that can be deployed in a virtual machine (VM) migration compatibility resource pool, Intel VT FlexMigration lets IT managers and CIOs quickly adopt emerging usage models like load balancing and business continuity during unforeseen spikes in data center resource demand. It also helps organizations protect their investments by using both older-generation and new-generation servers in the same resource pool. This white paper explains how Intel VT FlexMigration works, including how and why some instructions are masked by the hypervisor while others are not.

Introduction to Virtualization

Virtualization was first implemented on mainframe computing systems in the early 1960s by IBM Corporation with its IBM System/360* operating system (OS) products. IBM developed this product line because the huge mainframes of the day had a great deal of processor power that was unable to be fully utilized. operating systems were single-user and ran only one application at a time—which meant customers were buying expensive processing power but could not use all of it. With the System/360 OS, IBM devised a way to logically split mainframes into separate, smaller machines and allow them to run multiple projects at the same time. Figure 1 shows IBM's early virtualization model.

IBM unleashed the processing power of mainframes by using hardware configuration managers (HCM) or virtual machine monitors (VMM), also called hypervisors, which it co-developed with Massachusetts Institute of Technology (MIT). These programs interacted directly with the system

hardware, sometimes running on a dedicated component known as a service processor (SP). The VMM software enabled system hardware to be partitioned into multiple smaller machines.

The term “virtual” is used to indicate that access to the physical hardware is abstracted to hide the implementation details. For example, a VMM-controlled container accesses system resources like disks and network cards through interfaces presented by the VMM rather than through direct communication with the device. When running on specialized hardware, these containers can become fully isolated and independent environments capable of being separately powered, configured, booted, and administered. Essentially, the OS does not know it is sharing resources. Figures 2 and 3 show the different approaches taken by HCM VMM-based techniques.

Figures 2 and 3 show the different approaches taken by HCM VMM-based techniques.

Marco Righini
Intel Corporation
marco.righini@intel.com

What is a Virtual Machine Monitor?

VMM is the software component hosting virtual machines. In basic terms, the VMM, or hypervisor, is a layer of software between the underlying physical hardware and applications contained in virtual machines (VM), giving users the appearance of direct access to the real machine environment. The VMM allows operating systems to share hardware by presenting each OS with a hardware interface and tricking the OS into thinking it has full control of the physical hardware. The VMM emulates all functions of the physical hardware relating to the OS and the applications running on it.

Because the VMM also offers complete encapsulation of a virtual machine’s software state, the VMM layer can map and remap virtual machines to available hardware resources.

A VM is a set of files that can be easily moved and copied, making it possible to easily migrate virtual machines across different hardware and hardware vendors, completely decoupling the hardware stack from the software stack.

x86 Virtualization Challenges

Several challenges make it difficult to virtualize in an x86 environment:

Figure 1. IBM's early virtualization model

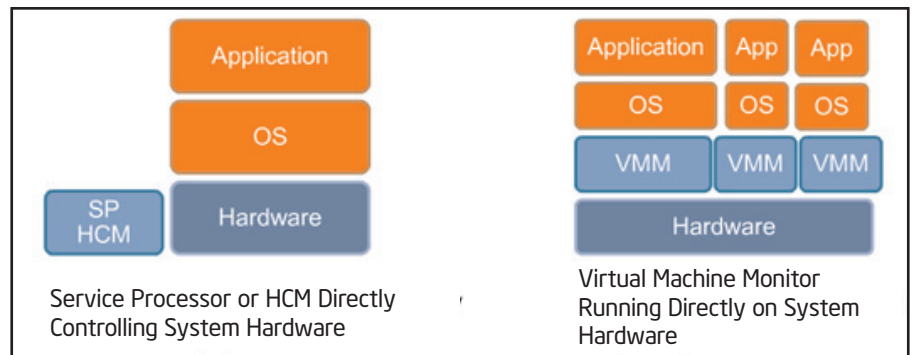
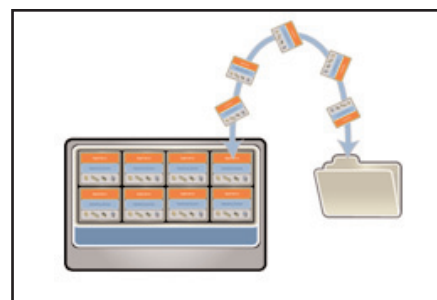


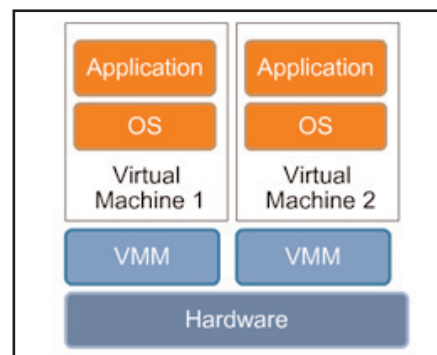
Figure 2. Encapsulation



- A non-virtualizable processor
- Hardware diversity
- Pre-existing software

Traditionally, the mainframe approach runs virtual machines in a less privileged mode to allow the VMM to regain control of privileged instructions. It relies on the VMM to virtualize and interface directly with the I/O devices. That means the VMM is in complete control of the machine.

Figure 3. Virtual machine monitor, also known as hypervisor



This traditional approach does not apply as easily to x86 architecture for several reasons. The 32-bit Intel® architecture is not naturally virtualizable. Popek and Goldberg showed that an architecture can support virtual machines only if all instructions that can inspect or modify privileged machine state will trap when executed from any but the most privileged mode.¹ Because x86

processors do not meet this condition, it is not possible to virtualize the processor by simply executing all virtual machine instructions in a less privileged mode. As a result of the PC's "open" architecture, PCs are a highly diverse group. In a traditional implementation, the virtual machine monitor would manage these devices, requiring a large programming effort in PC architecture to provide device drivers in the VMM for all supported devices. Unlike mainframes, which are configured and managed by experienced system administrators, desktop and workstation PCs are often pre-installed with a standard OS and set up and managed by the end user. In this environment, it is extremely important to allow the user to adopt virtual machine technology without losing the ability to continue using the existing OS and applications. In most cases, it would be unacceptable to completely replace an existing OS with a

VMM. To address the architecture issues discussed above, VMware developed virtualization for x86 computers in the late 1990s.

Any CPU with typical architecture will support at least two distinct operating modes, kernel mode and user mode (Figure 4).

Kernel mode is the mode in which the OS runs. It is also referred to as unrestricted mode, master mode, supervisor mode, or privileged mode. As Figure 4 shows, privileged instructions—such as sensitive register instructions that modify or read values containing information about operating mode, status, and the state of the processor—are executed by the OS under kernel mode. A program in kernel mode is trusted to never fail because, if it does, the whole computer system will likely crash.

In user mode, only non-privileged instructions are executed. In other words, user programs that run in this mode only permit a subset of instructions to be executed and a subset of the hardware features to be accessed. In general, all instructions involving I/O and memory are disallowed in user mode.

Non-privileged indicates that it is forbidden for processes running in this mode to access those portions of memory that have been allocated to the kernel or to other programs. Non-privileged instructions are also forbidden to address the I/O directly or to manipulate the state of memory.

We just discussed CPU modes and how instructions are executed in different privilege levels. A similar concept that is applied to the OS layer uses concentric rings to define privilege levels (Figure 5).

Figure 4. Privilege levels in the CPU layer

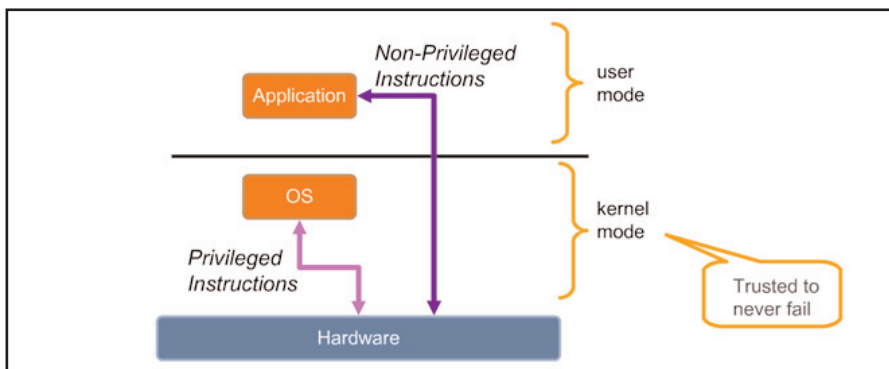
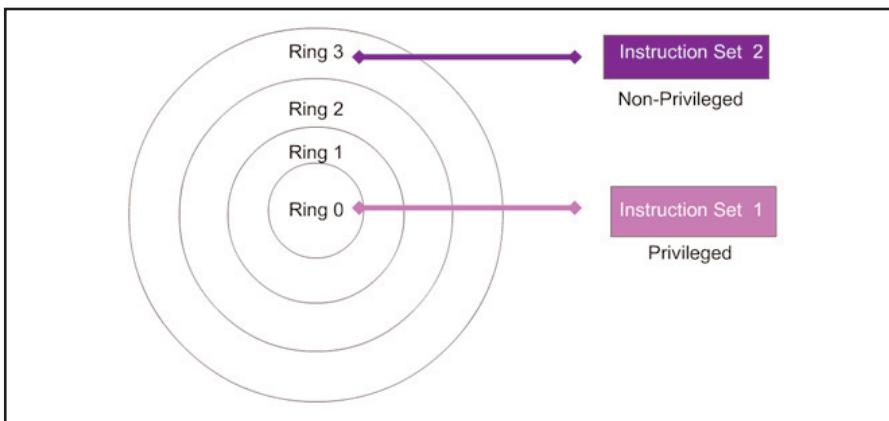


Figure 5. Rings and privilege levels



Intel® Virtualization Technology FlexMigration on Enablement

Within the OS layer, applications talk to the OS for all user interface and file management operations. The talk is a set of instructions such as "read data" or "write to a printer." Each instruction set is associated with a privilege level. General-purpose processor architectures support code execution on multiple privilege levels. However, most operating systems use only two rings:

- Ring 0 for the OS
- Ring 3 for applications

Programs running in ring 0 can do anything with the system. The application code running in ring 3 should be able to fail at any time without impacting the rest of the computer system.

Rings 1 and 2 are typically not used, but could be configured with different levels of access.

So far, we have discussed how the OS is the gatekeeper when performing privileged instructions. Since the OS operates in the kernel mode, it has permissions to perform privileged instructions. However, a VMM will typically have more than one OS (guest OS) installed. With a VMM, the OS operates in user mode. Figure 6 shows how the VM's OS (the guest OS) performs the privileged instructions from the user mode.

To allow coexistence with more than one OS, the VMM is implemented at the kernel mode (privileged level) and the OS of the virtual machine is moved to the user mode (non-privileged level). Doing this allows the VMM to emu-

late all the privileged instructions each virtual machine's OS attempts to execute.

Note that the VMM runs in kernel mode and controls access to the resources shared by all virtual machines on the same physical computer. As the same physical resources are accessed by different virtual machines, the VMM schedules the virtual machines' requests and allocates processor cycles to them as appropriate. Figure 7 shows how privileged instructions are handled in the VM.

To handle the problematic instructions in x86, VMware developed a new virtualization technique that combines traditional direct execution with fast, on-the-fly binary translation. A binary translator traps the non-virtualizable, privileged

Figure 6. The VMM has the control

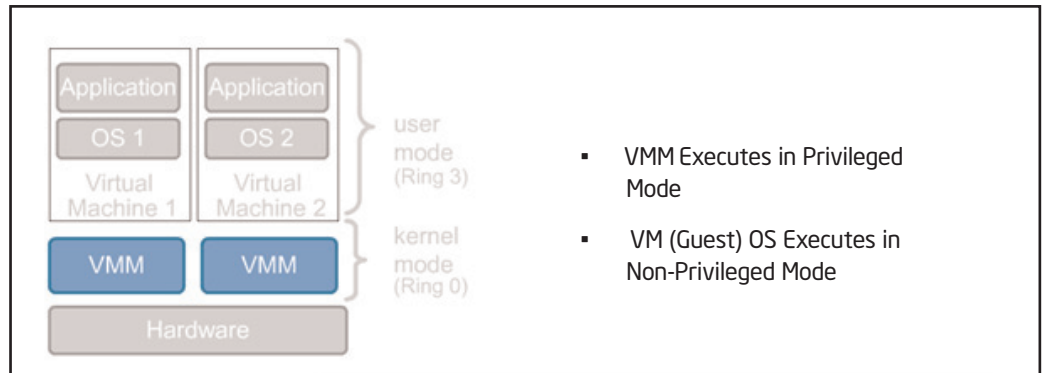
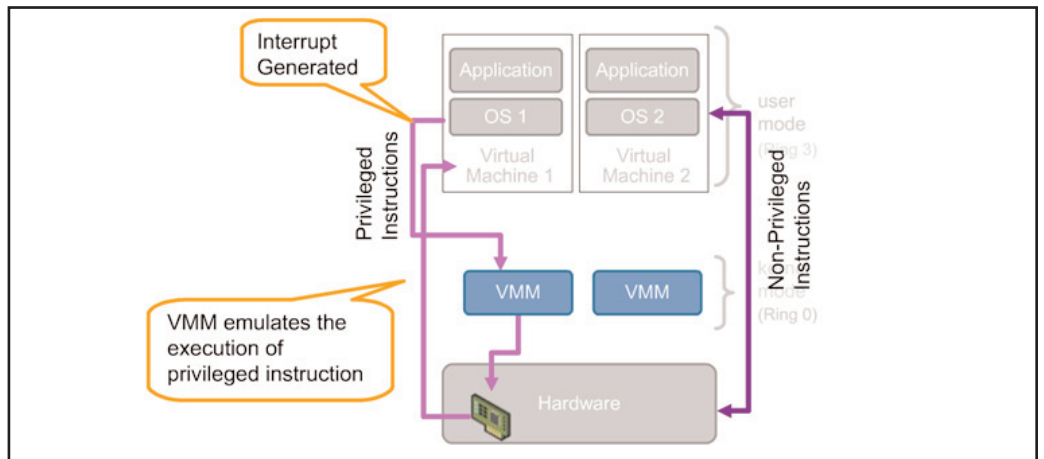


Figure 7. How privileged instructions are handled in the VM



instructions generated by the guest OS and converts them into safe instructions the VMM can emulate. The VMM then uses a direct execution method for handling user-mode, non-privileged code. The result is a high-performance VM that matches the host hardware and maintains total software compatibility.

Direct execution (Figure 8) is a basic virtualization technique for executing the VM on a physical machine while letting the VMM retain ultimate control of the CPU. When the VM attempts to perform a non-privileged instruction, it is directly executed by the host OS with no intervention from the VMM.

Table 1 compares direct execution and binary translation.

To summarize, it is desirable to use direct execution where possible because it is faster; however, in x86 architecture this method cannot handle privileged instructions. Therefore, direct execution in x86 architecture is used for executing user-level (non-privileged) code, and binary translation is used to handle the privileged instructions generated by the guest OS.

Intel® Virtualization Technology

The other possible approach to virtualizing privileged instructions is to provide this capability in hardware, which means having circuits in the processor and memory controller that enhance the running of multiple operating systems. Hardware virtualization is an enabling technology that provides functions to partition memory more effectively, as well as providing or emulating multiple sets of registers. Intel® Virtualization Technology (Intel® VT) is an example of hardware virtualization (VM) for the x86 architecture.

Intel VT provides processor support for virtualization using a new form of processor operation called virtual machine extensions (VMX).

Table 1. Direct Execution vs. Binary Translation

Feature	Direct Execution	Binary Translation
Latency	Minimal start-up latency	Translation overhead
Speed	Native speed	Usually slower than execution on first encounter
Use	Not always possible; typically used for user-level code (non-privileged)	Always possible; all remaining cases including kernel code (privileged)

There are two kinds of VMX operations: root and non-root.

VMX sits at a privilege level below ring 0. (You can think of it as ring -1.) The VMM runs in VMX root operation while the guest OS runs in VMX non-root operation, as shown in Figure 9. By creating this new operating mode, VMMs can run in a software layer that is more privileged than the software layer running the OS. This technology helps to avoid binary translation and para-virtualization techniques for x86 privileged instructions.

The Migration Issue

High-performance virtualization requires delivering virtualization at near-native performance. For this to be technically feasible while maintaining correctness, it is necessary to execute many instructions directly on the underlying hardware. For any virtualization layer, underlying hardware resources are used to facilitate high-performance execution. This ensures applications running inside a virtual machine do not encounter major slowdowns. The virtualization layer intercepts only select instructions that require interception and subsequent emulation.

The x86 instruction set architecture (ISA) allows only instructions belonging to the OS or kernel to be intercepted by the traditional virtualization layer (i.e., a virtualization layer that does not rely on hardware support for virtualization or on para-virtualization). These instructions run at the high-

est privilege level, making them privileged code.

Typically, instructions that run at lower privilege level (non-privileged code) are passed through to the underlying hardware unimpeded.

The CPUID instruction can be executed by the OS (privileged code) as well as by applications (non-privileged code). If this instruction is part of an application, it executes directly on the underlying hardware, thereby giving an application a list of features available on the underlying hardware. If CPUID is part of privileged code, the virtualization layer can intercept it if emulation is needed. This methodology comes from attempting to virtualize the inherently non-virtualizable x86 ISA.

Therefore, even when applications are running within a virtual machine, this may be true:

- Applications might have many of their instructions running directly on underlying hardware.
- Instructions such as CPUID might be executed as non-privileged and therefore run directly on the underlying hardware, potentially tying an application to the set of features supported by the CPU on a particular host.

Live Migration

Being able to migrate an entire VM while it is running is useful only if applications continue to operate normally after migration.

Applications typically include non-privileged instructions, which might execute directly on the underlying hardware CPU. If applications are to continue performing normally after a live migration, they need to execute instructions and utilize features that were available on the source host hardware—even when

they are running on the destination host hardware. This is required even though these applications are executing within the same virtual machine.

Cold migration, on the other hand, ensures that the virtual machine and underlying applications restart after migration. Because the applications restart, they initialize and query the CPU for available features. This implies the applications utilize only instructions and

features supported by the destination host hardware (where the virtual machine is powered on).

Live migration is a required technology for building an agile, dynamic data center based on server virtualization. However, it has not been possible to perform successful live migration between servers based on different generations of processors, each with different instruction sets. As described earlier,

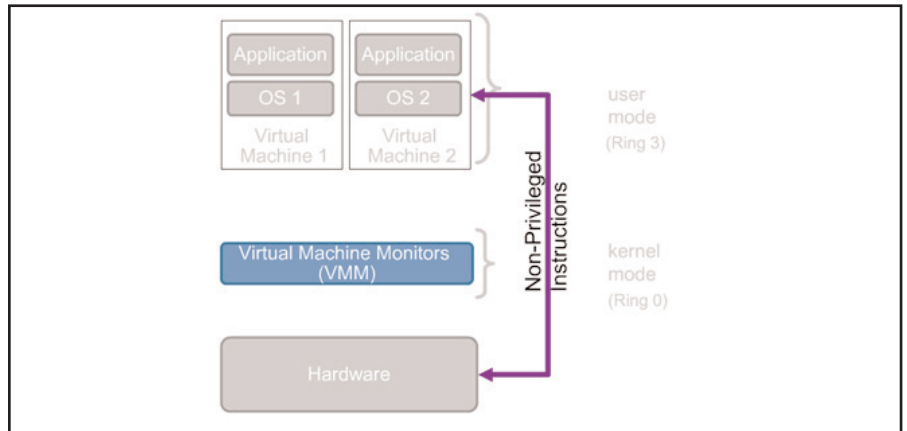


Figure 8. Direct execution

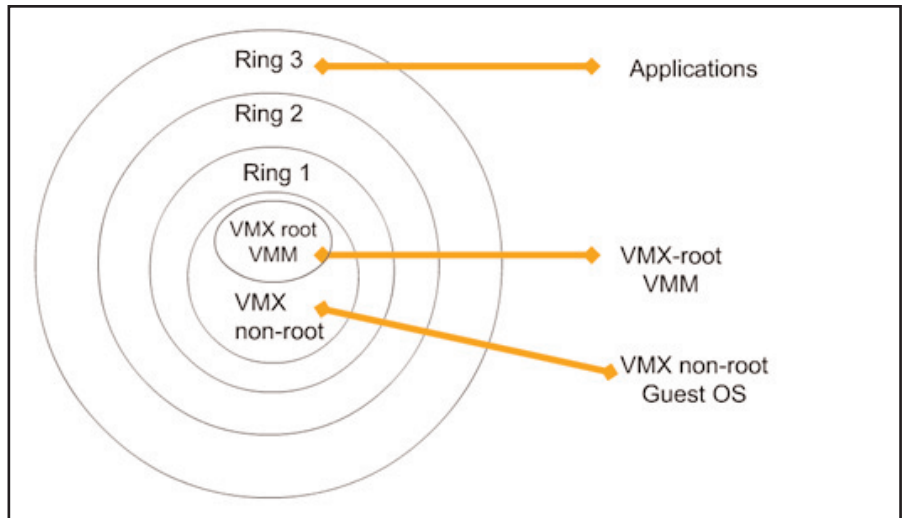


Figure 9. Intel's solution for processor virtualization

this is because the direct execution approach sees the exact instruction set implemented within the CPU. This limits the ability to implement large resource pools, creating islands of servers and hindering the implementation of advanced data center capabilities. Because CPUs change so quickly, this is a growing issue for data center management.

Intel VT FlexMigration and VMware's Enhanced vMotion* are designed to overcome this limitation by enabling all servers to expose the same instruction set to applications—even if they are based on different processor generations from a single CPU supplier.

VirtualCenter* Compatibility Checks for VMotion

Unless you are using Enhanced VMotion compatibility, VirtualCenter* must match up the features of the source and destination host CPUs to ensure that a VM and its applications operate normally after migration with VMotion. VirtualCenter does this using features reported by the CPUID instruction, matching features between the source and destination host CPUs to ensure that all features are compatible.

VirtualCenter also has a set of masks you can apply to hide features reported by the CPUID instruction so that the query can view fewer features. These masks are used only when CPUID is executed as a part of privileged code. They ensure that only features that matter to the execution of the VM are compared.

Because these features are present in hardware, non-privileged code querying CPUID is

not affected by any part of the virtualization layer; thus, these masks are not applied when the non-privileged code makes the query.

Applying CPUID override masks circumvents CPU compatibility checks for VMotion. (An application running in a virtual machine still might crash if it does not find the set of instructions that were available to it on the source host hardware.)

These facts and examples show why CPU compatibility checks are necessary for migration with VMotion. These checks not only ensure that the migration succeeds, they ensure the applications running inside a VM continue to operate normally on the destination host hardware. Intel VT FlexMigration has been supported since VMware Infrastructure* 3.5 update 2.

Compatibility Pools

This section reviews which CPUs are compatible without Enhanced vMotion and Intel VT FlexMigration and the changes with Intel VT FlexMigration and Enhanced vMotion.

Before Intel VT FlexMigration and Enhanced vMotion

Before the support of Intel VT FlexMigration and VMware's Enhanced vMotion, there were barriers to running VMware VMotion between different generations of CPUs. Suppose a VM was running on a dual-core Intel Xeon processor 5160 series and using the streaming SIMD extensions (SSE) of that type of CPU. The VM would crash if we tried to live migrate it to a different type of CPU that did not have the same SSE instruction. This is why VMware Infrastructure prohibits the migration on different type of CPUs.

Some Myths

Many people had the misconception that clock speed, number of cores, caches, and stepping all made the difference in blocking a live migration. In fact, for the reasons we described earlier, just the instruction set matters. Table 2 describes which Intel Xeon processors were compatible together before Intel VT FlexMigration and Enhanced vMotion.

Compatibility Pool with Intel VT FlexMigration and VMware Enhanced vMotion

It is important to establish the compatibility pool with Intel VT FlexMigration and VMware Enhanced vMotion, starting with VMware Infrastructure 3.5 update 2 or later. CPUs enabled with Intel VT FlexMigration adapt to the oldest CPU for the ring 3 instructions. This means:

- The VM running in such a cluster will default to core instructions. These are the most compatible in the cluster. (This is true in VMware Infrastructure 3.5 update 2 or later. It will be different in VMware vSphere* products.)
- You cannot live migrate an up-and-running VM from a cluster that is not Enhanced vMotion-enabled to one that is enabled.

All these CPUs will be compatible. Also, all servers based on these CPU families may work within the same compatibility pool. Servers based on the Intel Xeon processor 5500 series also work in an Enhanced vMotion-enabled cluster with VMware ESX* 3.5 update 4.

Table 2. Intel® Xeon® Processor Compatibility Before Intel VT FlexMigration and Enhanced vMotion

Intel® Xeon® Processor Series	No. Cores	Frequency	Cache (MB)	System (No. Sockets)	Intel® Core Microarchitecture
5100	2	4 - 8	4	Dual	65 nm
5300	4	1.6 - 3.0	8	Dual	65 nm
7300	4	1.6 - 2.93	4 - 8	Quad or more	65 nm
7200	2	2.4 - 2.93	8	Quad or more	65 nm
5200	2	1.86 - 3.4	6	Dual	45 nm
5400	4	2.33 - 3.40	12	Dual	45 nm
7400	6	2.13 - 2.66	8 - 16	Quad or more	45 nm
7500	8	1.86 - 2.36	24 maximum	Quad or more	45 nm
5500	4 + SMT	1.86 - 2.93	4 - 8	Dual	45 nm
5600	6 + SMT	2.13 - 3.33	12 maximum	Dual	32 nm

Table 3. Intel® Xeon® Processor Compatibility After Intel VT FlexMigration and Enhanced vMotion

Intel® Xeon® Processor Series	No. Cores	Frequency	Cache (MB)	System (No. Sockets)	Intel® Core Microarchitecture
5100	2	1.6 - 3.0	4	Dual	65 nm
5300	4	1.6 - 3.0	8	Dual	65 nm
7300	4	1.6 - 2.93	4 - 8	Quad or more	65 nm
7200	2	2.4 - 2.93	8	Quad or more	65 nm
5200	2	1.86 - 3.4	6	Dual	45 nm
5400	4	2.33 - 3.40	12	Dual	45 nm
7400	6	2.13 - 2.66	8 - 16	Quad or more	45 nm
7500	8	1.86 - 2.26	25 maximum	Quad or more	45 nm
5500	4 + SMT	1.86 - 2.93	4 - 8	Dual	45 nm
5600	6 + SMT	2.13 - 3.33	12 maximum	Dual	32 nm

Please note, Intel NetBurst® architecture-based CPUs (e.g., Intel® Pentium® 4 processor) are out of the scope and are compatible.

The best way to check which instructions are available in each EVC mode is to use CPUID within a VM, and use the bits to see which instructions are available. Figure 10 shows an example.

You can find updated documentation for the newest CPUs and instructions at <http://www.intel.com/Assets/PDF/app-note/241618.pdf>.

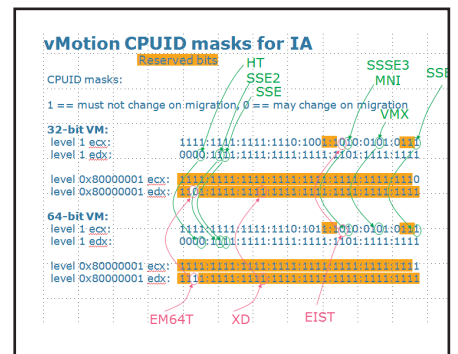
Using and Implementing Intel® Virtualization Technology FlexMigration, Step by Step

Setup and configuration on VMware ESX 3.5 update 2 or later:

- **Install or upgrade** the existing virtual data center server to meet the requirements for VMware Infrastructure 3.5 update 2 or later. This has no impact on the existing infrastructure. Please note that since the database structure is changing, it is essential to make a backup.

- **Upgrade or install VMware ESX 3.5 update 2 or later.** If the server is already in a production environment and in a VMware DRS* cluster, put it into maintenance mode to make sure that all VMs are

Figure 10. Checking which instructions are available



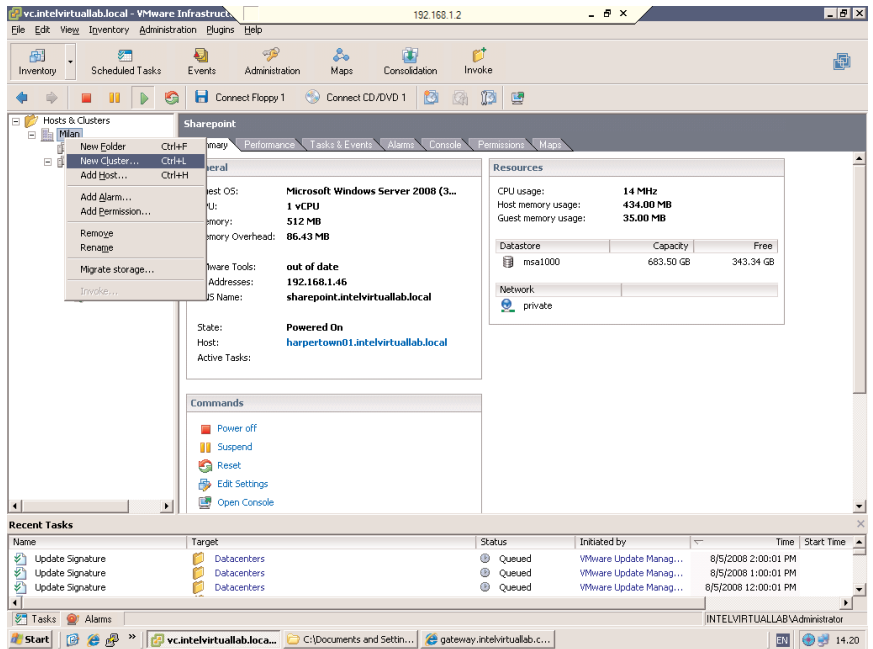


Figure 11. Create a new cluster

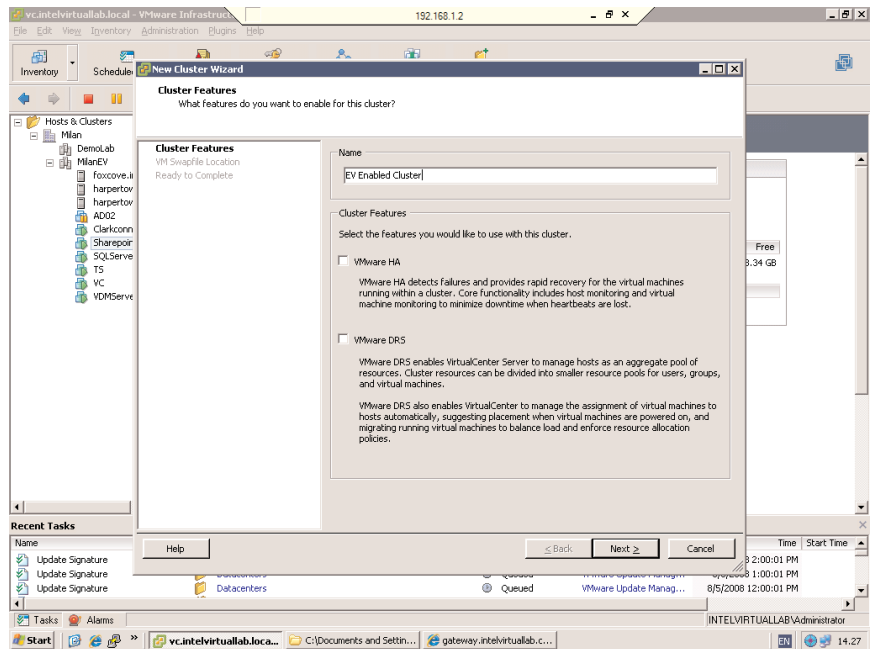


Figure 12. Cluster creation wizard

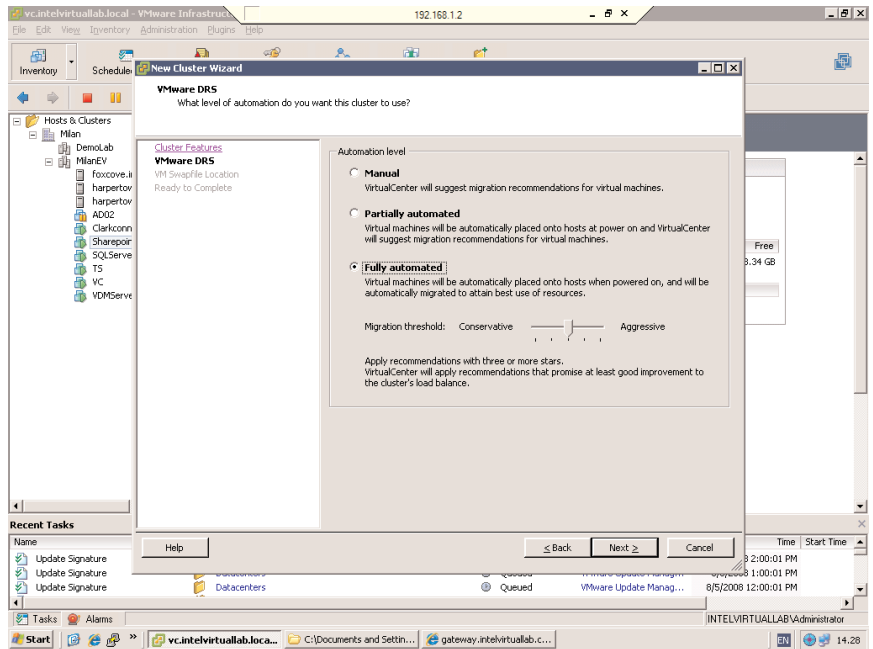


Figure 13. Choose whether you want DRS to be fully automated, partially automated, or manual

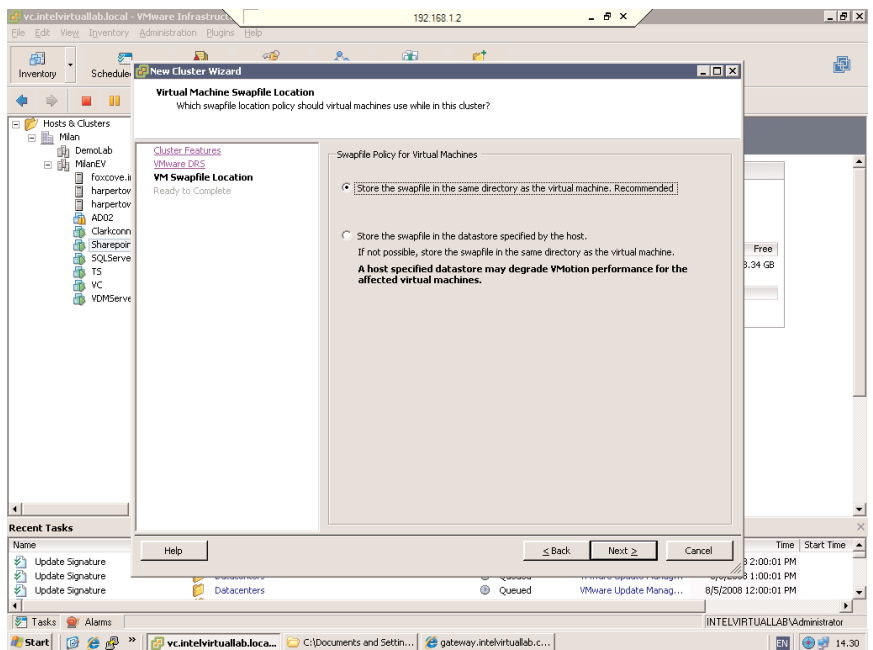


Figure 14. Choose "Default" when asked where the VM swap file should be stored

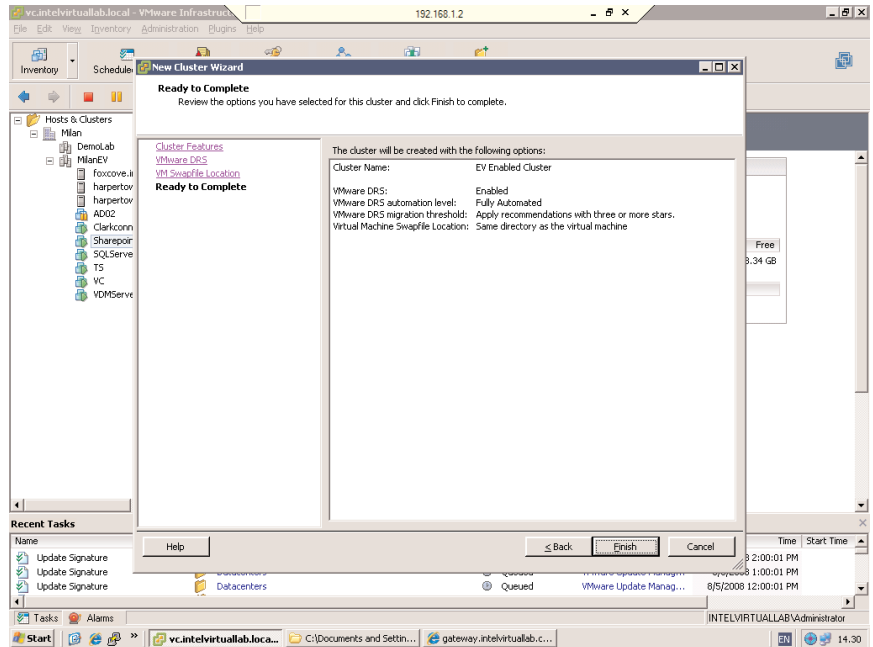


Figure 15. Confirm everything by pressing the finish button

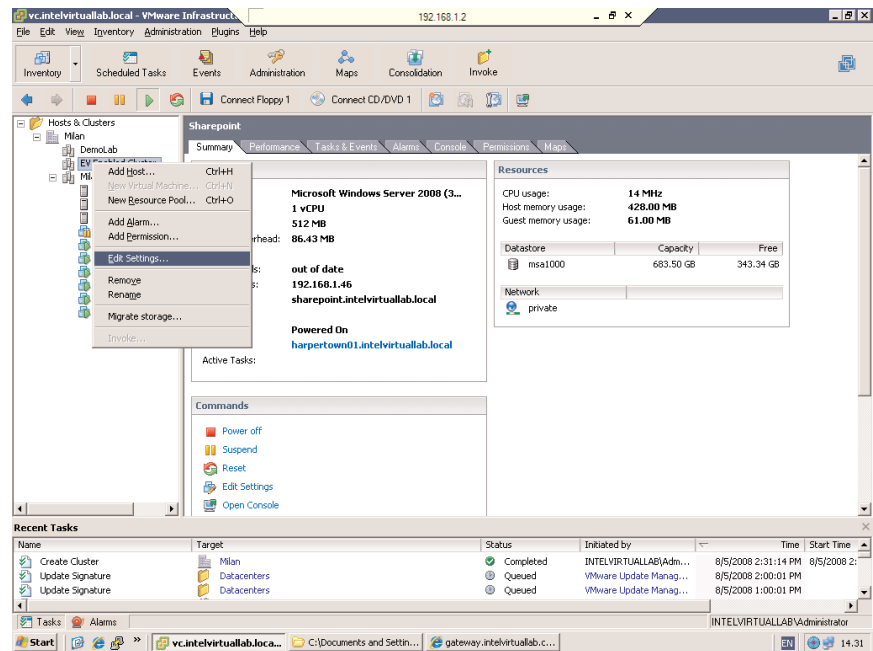


Figure 16. Right-click on the cluster and go to the edit settings menu

evacuated. Roll-upgrade all the VMware ESX servers that participate to Enhanced vMotion clusters.

- Once all VMware ESX servers are upgraded, recreate a new cluster with the virtual infrastructure using these six steps:

1. Right-click on the data center to create a new cluster (Figure 11).
2. A wizard will pop up asking for the name of the new cluster and what type of cluster this will be: HA cluster, DRS cluster, or both (Figure 12).
3. Choose whether you want DRS to be fully automated, partially automated, or manual (Figure 13).
4. Choose "Default" when asked where the VM swap file should be stored (Figure 14).
5. Confirm everything by pressing the finish button (Figure 15).

6. Once created, the cluster will appear in the infrastructure. Right-click on it and go to the edit settings menu (Figure 16).
7. Click on the VMware Enhanced vMotion Compatibility (EVC) and enable EVC for Intel hosts and click OK (Figure 17). (Note that it is not possible to enable EVC on any existing cluster.)
8. Once the cluster is created, you need to move to hosts (the one with VMware ESX 3.5 Update 2) in the EVC-enabled cluster by first putting them into maintenance mode (Figure 18).
9. Once in maintenance mode, you can just drag and drop the host into the new cluster (Figure 19).

To enable an Enhanced vMotion cluster, power off all running VMs (Figure 20). Do not suspend VMs, since this will keep instructions, processes, and related allocated memory dumped to disk.

This might end in a non-resumable VM once it is migrated into the Enhanced vMotion-enabled cluster (Figure 21).

Setup and Configuration on vSphere* Cluster Creation and Enabling Enhanced Vmotion*

This section describes the steps to follow to create a cluster and enable VMware Enhanced VMotion*, which uses Intel VT FlexMigration technology.

There are two pre-requisites before adding any host into an Enhanced VMotion-enabled cluster:

1. Enable the execute disable bit in the BIOS.
2. Enable Intel's VT bit in the BIOS and select an AC power cycle.

To create and configure the cluster:

- After logging into the vSphere client, create the high-level data center node (Figure 22).

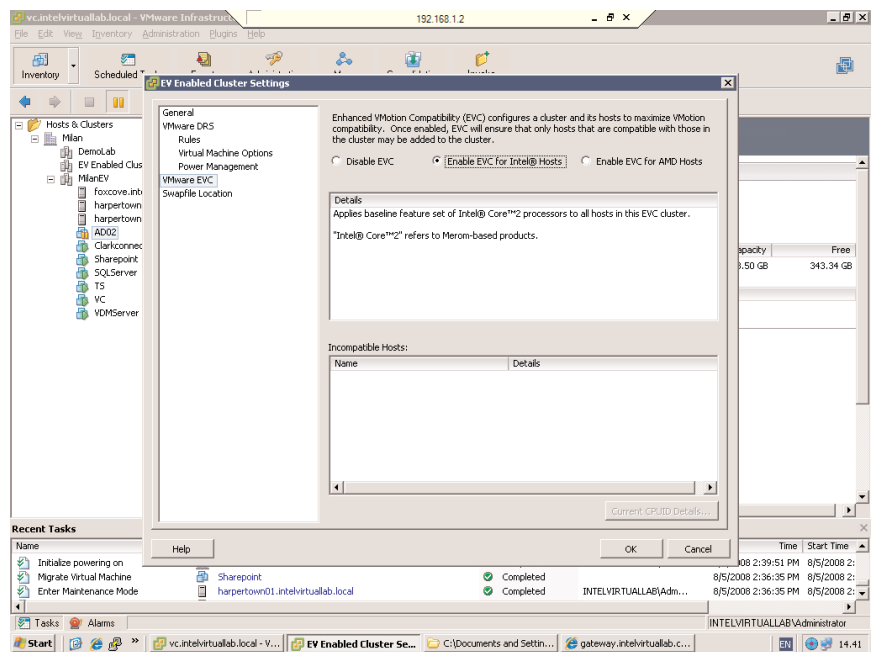


Figure 17. Click on the VMware EVC and Enable EVC for Intel hosts and click OK.

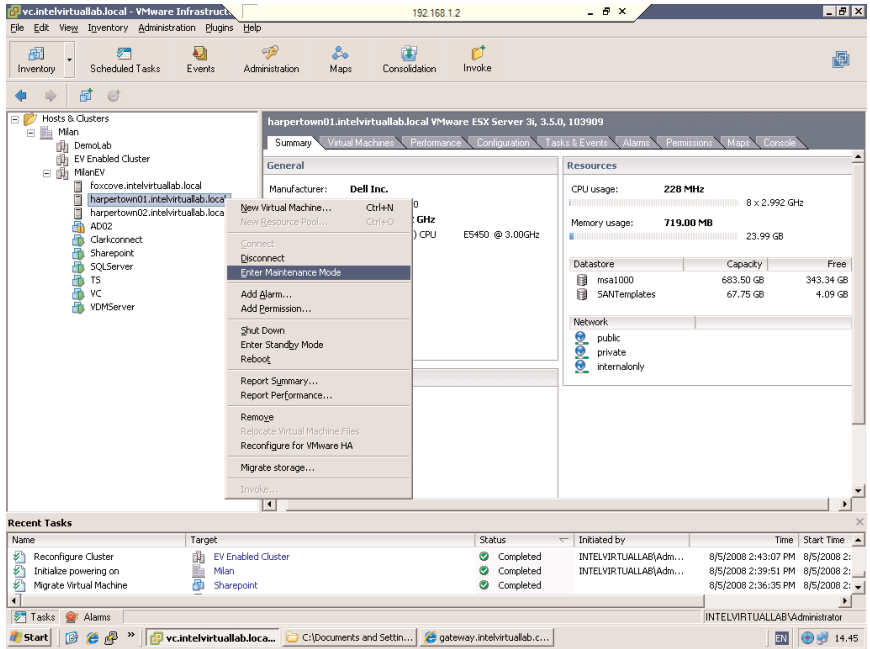


Figure 18. Select maintenance mode

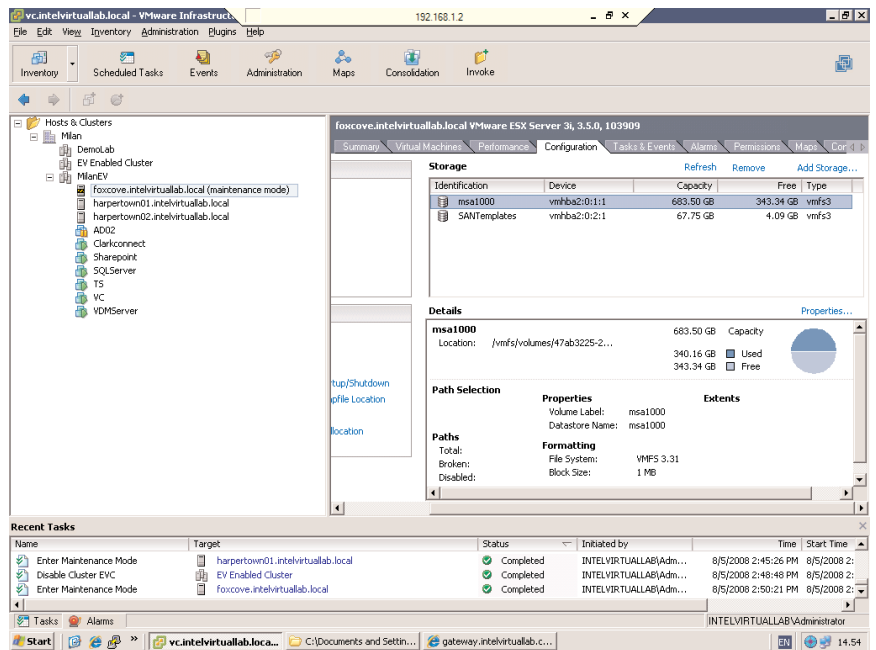


Figure 19. Drag and drop the host into the new cluster

Intel® Virtualization Technology FlexMigration on Enablement

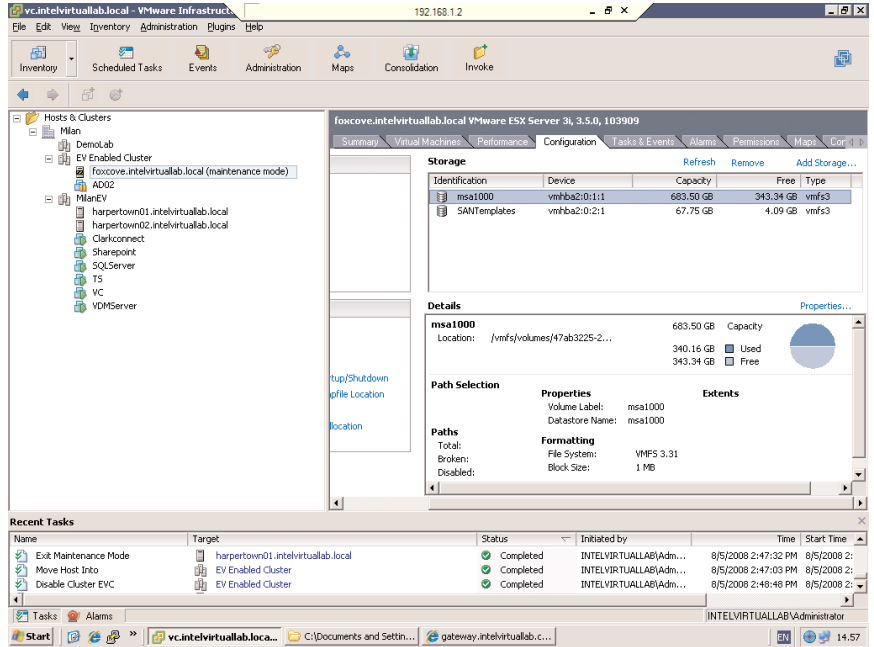


Figure 20. Power off all running VMs

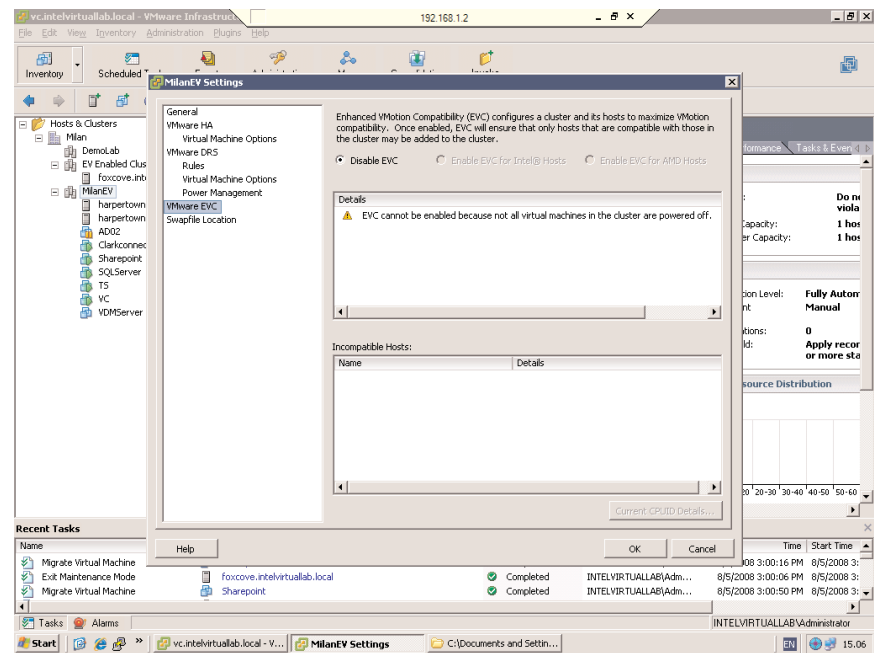


Figure 21. Error message saying that EVC cannot be enabled because there are some running VMs

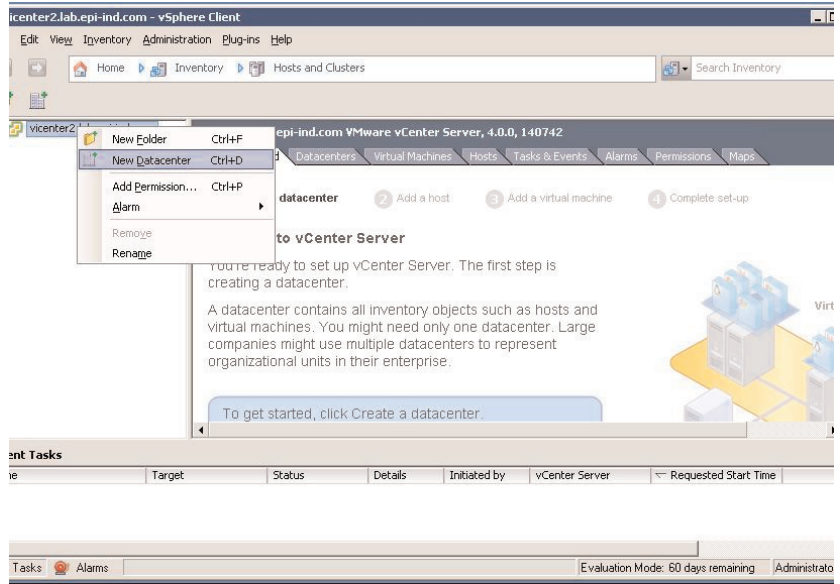


Figure 22. Create the high-level data center node

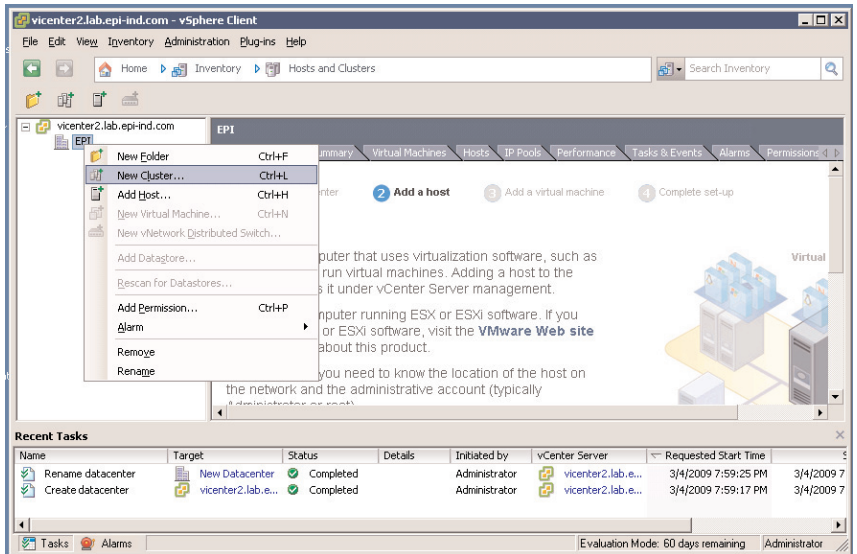


Figure 23. Create the cluster

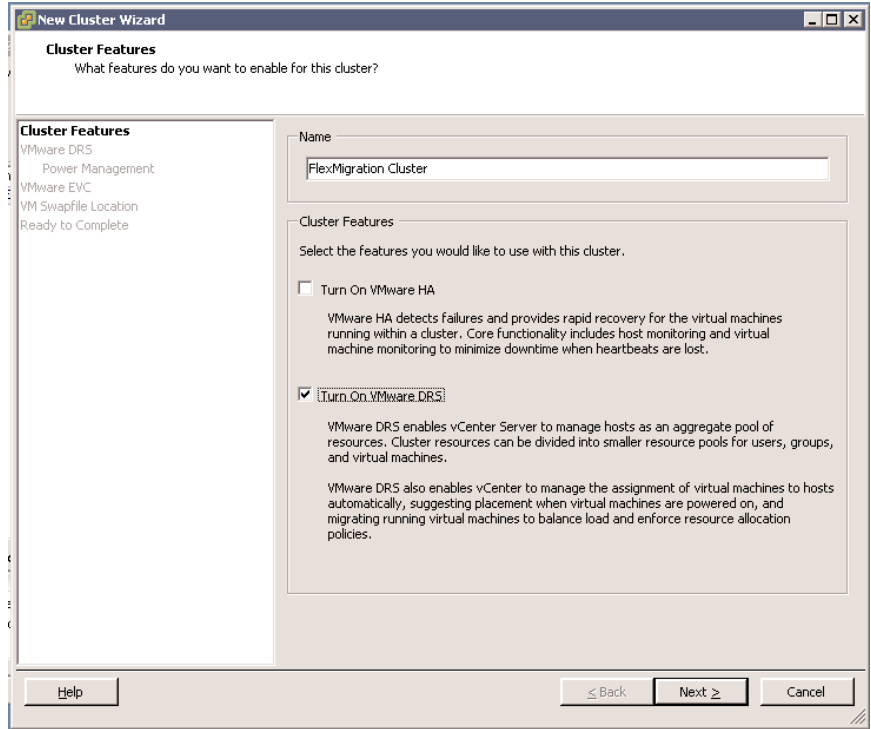


Figure 24. Enable the flags for HA and DRS

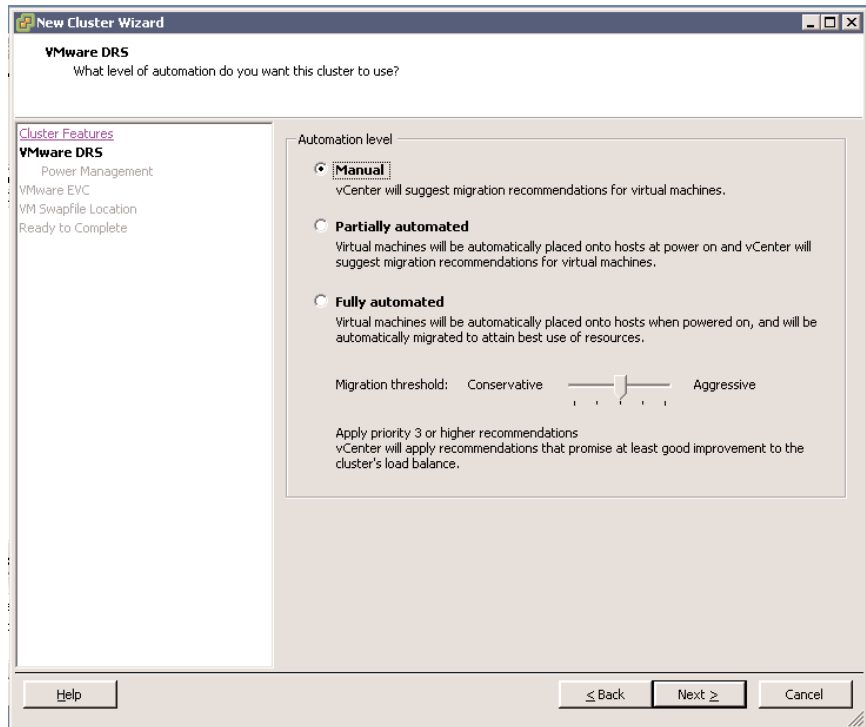


Figure 25. Provide additional settings for the DRS

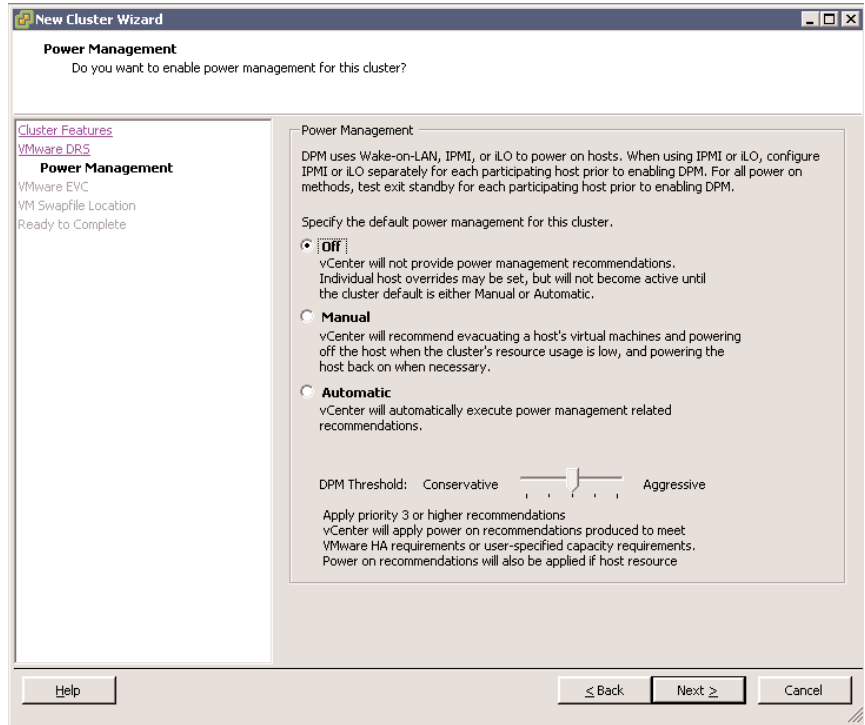


Figure 26. Enable or disable Dynamic Power Management

After creating the data center, create the cluster (Figure 23).

Depending on your requirements, enable the appropriate flags for HA and DRS. You can change this setting any time by editing the cluster settings (Figure 24). (For this example, we will enable on the DRS feature.)

Since we enabled the DRS feature, we need to provide additional settings for the DRS. Note that the number of live migrations depends on this setting. For example, if the mode is set to fully automated and aggressive, then, depending on the workload, if there is much variation, you would see many migrations happening to balance the cluster (Figure 25).

Dynamic Power Management (DPM) is a new feature in VMware ESX 4.0. This was introduced earlier as a test feature. Depending on the requirement, enable or disable this option. Note that additional hardware is

required to enable this option, which is specified in Figure 26.

The EVC configuration setting depends on the different kinds of hardware on the VMware ESX hosts that need to be added into the cluster.

Table 4 summarizes the settings based on the hardware of the ESX hosts present in the cluster.

Note that if all the servers in the cluster belong to the Intel Xeon processor 5500 series and the EVC is set to Intel® Core™2 processor, then the applications running on the server with in the virtual machines will not be able to utilize the new instructions present in the Intel Xeon 5500 series processors.

Table 5 shows a high-level summary of the instructions present in these families of processors.

For optimal performance, choose the recommend option of storing the swap file in the same directory as the VM (Figure 27).

Ensure that all the settings are correct and complete the creation of the cluster (Figure 28).

After you have created the cluster, you can start adding hosts. If the hosts have already been added directly to the data center, you need to move them into the cluster making sure that all VMs in the hosts are shut down and the BIOS settings mentioned at the start of the section are enabled (Figure 29). Next:

- Specify the connection string for the host, which can either be an IP address or a fully qualified domain name (FQDN). If any conditions are not met (e.g., BIOS changes), appropriate errors/warnings come up. If the host is being added to the cluster for the

Table 4. Enhanced vMotion Configuration Settings

Intel® Microarchitecture	EVC Setting	Example
Intel® Core™ microarchitecture	Intel® Xeon® Core 2	Intel Xeon processor 5365, 5450, and 5570
Next-generation Intel Core microarchitecture		
Next-generation Intel® microarchitecture		
Next-generation Intel Core microarchitecture	Intel Xeon Core 2 (45nm)	Intel Xeon processor 5450, 7460, and 5570
Next-generation Intel microarchitecture		
Next-generation Intel microarchitecture	Intel Xeon Core i7	Intel Xeon processor 5570
Next-generation Intel microarchitecture	Intel Xeon Core i7 (32nm)	Intel Xeon processor 5670

Table 5. Instruction Sets

Intel® Microarchitecture	Instruction Set
Intel® Core™ microarchitecture	Baseline that includes x87, SSE, SSE2, SSE3, MMX, etc.
Next-generation Intel Core microarchitecture	SSE4.1 (40+ media/video instructions)
Next-generation Intel® microarchitecture	SSE4.2 (7 instructions [POPCNT, CRC32, PCMPGTQ, STTNI])
Next-generation Intel microarchitecture	AES-NI encryption instruction

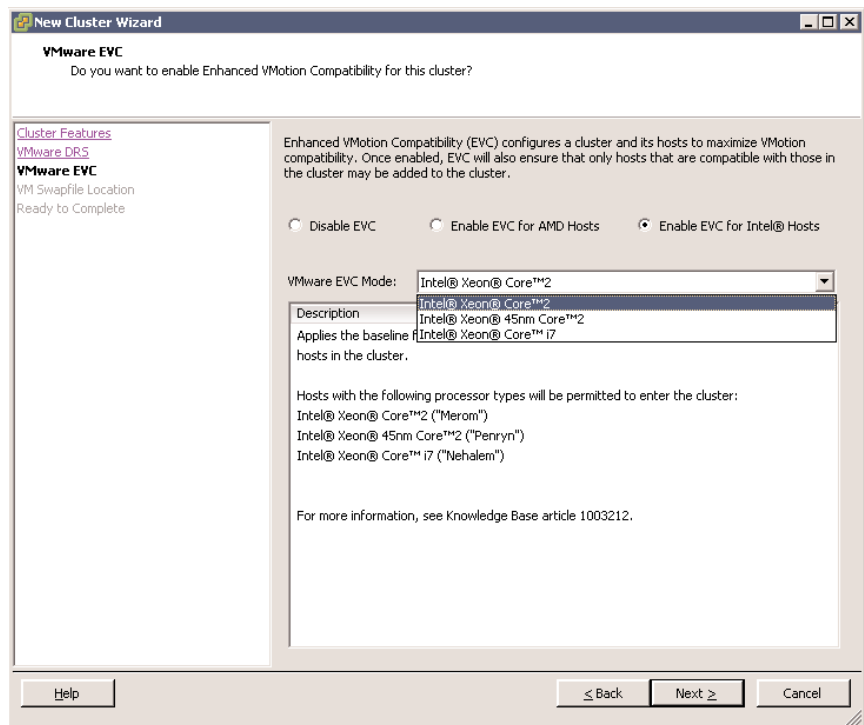


Figure 27. EVC set to Intel® Core™2 processor

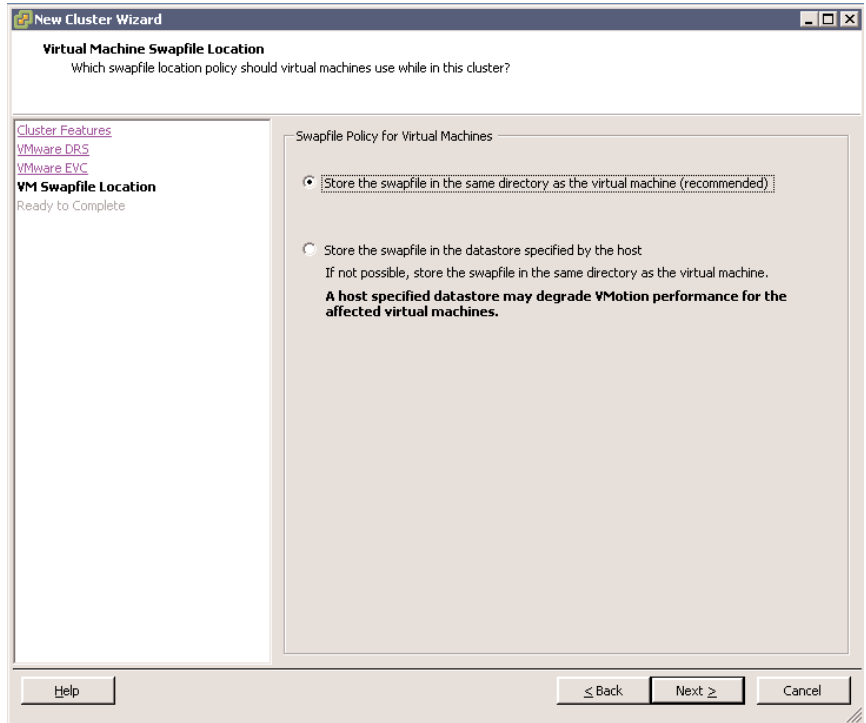


Figure 28. Store the swap file in the same directory as the VM

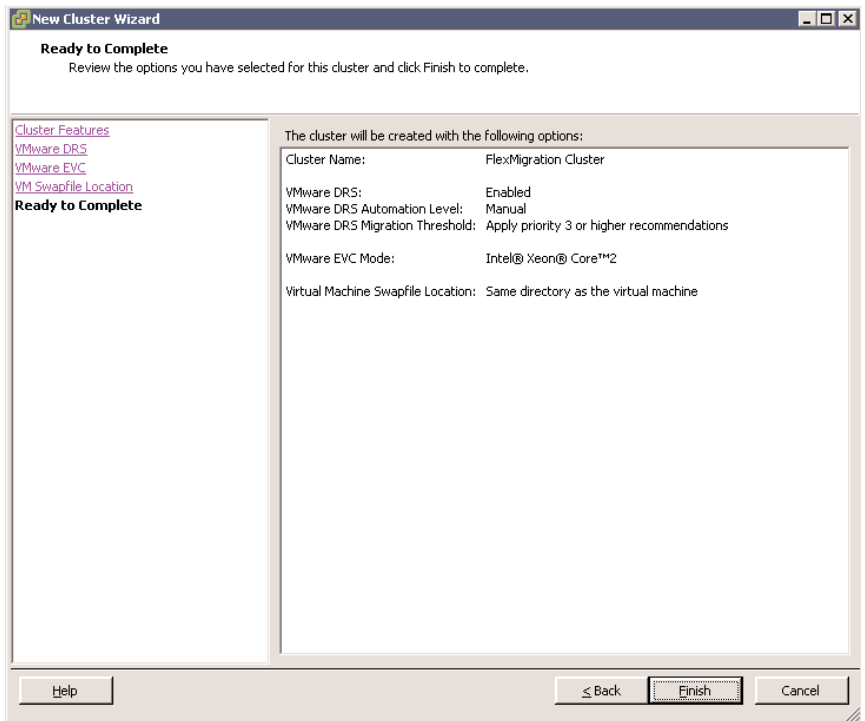


Figure 29 Complete the creation of the cluster

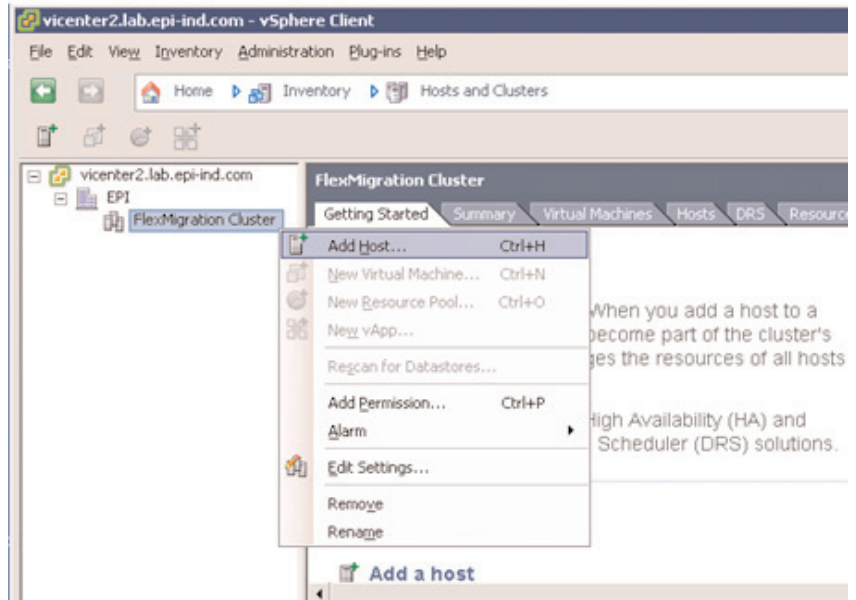


Figure 30. Add hosts into the cluster

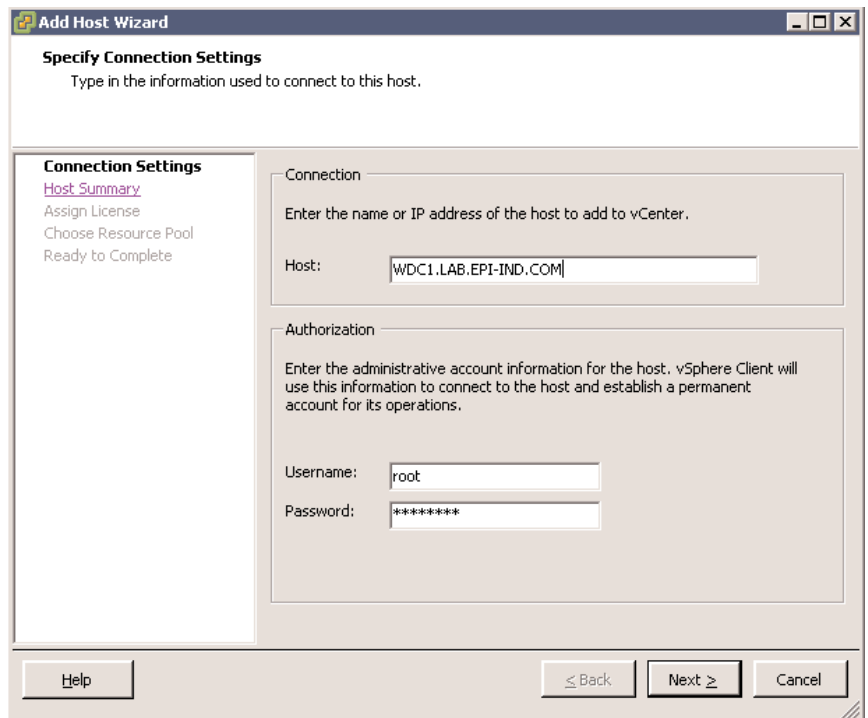


Figure 31. Specify the connection string for the host

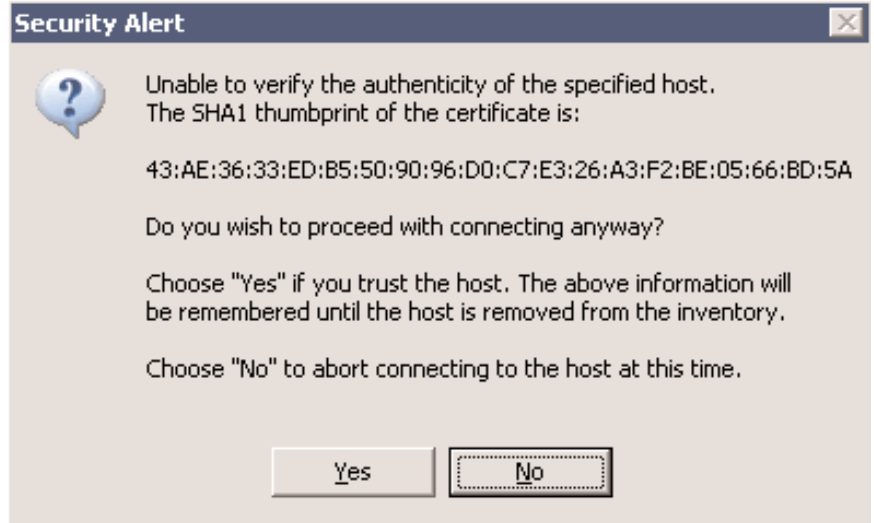


Figure 32. Confirm the authenticity of the server

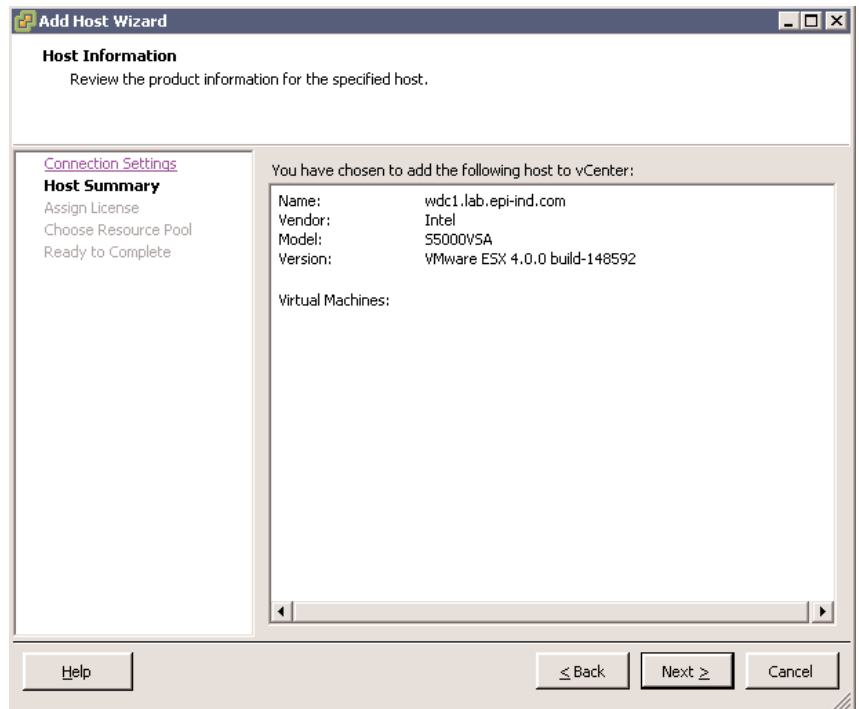


Figure 33. Review the product information for the host

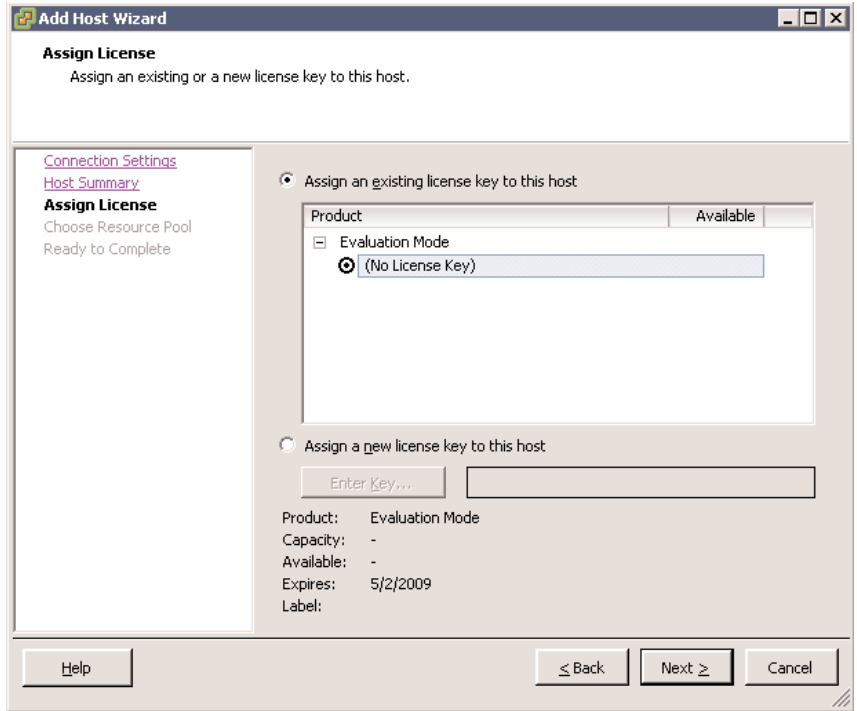


Figure 34. Use the host in evaluation mode

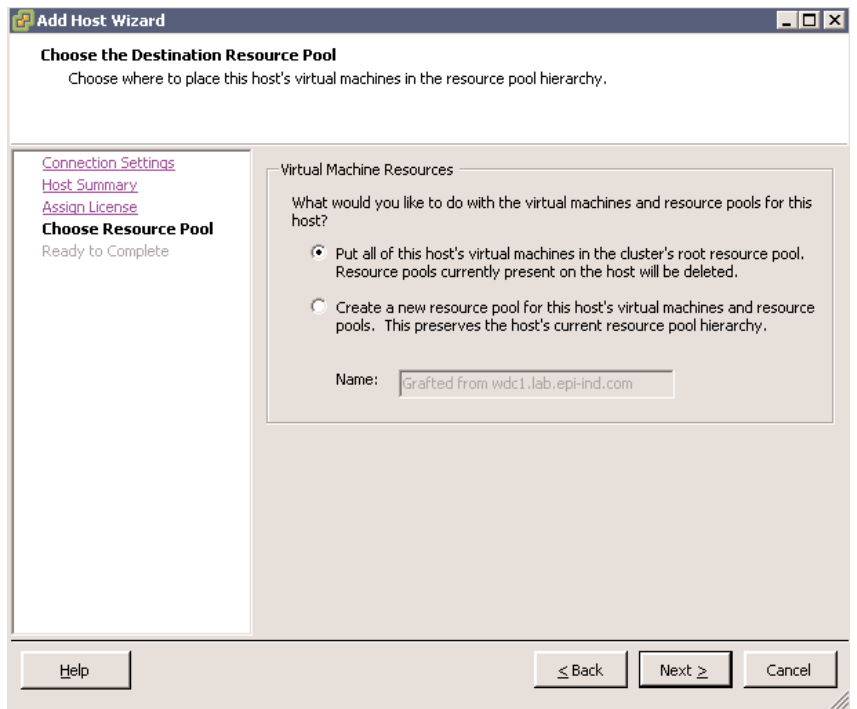


Figure 35. Choose the destination resource pool

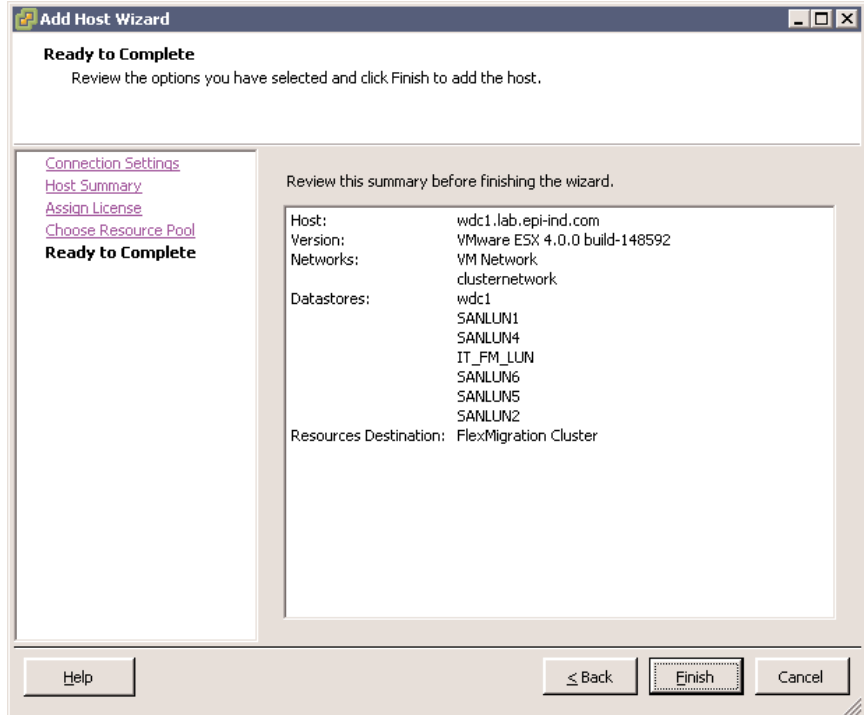


Figure 36. Click finish to add the host

- first time, you will be alerted to confirm the authenticity of the server (Figure 30).
- Confirm the host creation settings and go to the next screen (Figures 31 through 33).
- Enter the license information if you have it, or continue to the next step to use the host in evaluation mode (Figure 34).

Based on your needs, choose the resource pool configuration and complete the final step of the host creation. Repeat the steps for adding additional hosts into the cluster (Figures 35 and 36).

Pain Points

- It is necessary to create a new cluster to enable EVMotion.

- Never migrate suspended VMs. Not considering this may cause loss of data and even complete corruption of the VM.
- Never disable on an enabled Enhanced vMotion cluster. If this happens, you will have to recreate the complete cluster.

Summary

Virtualization has three key aspects that make portability of VMs very easy:

- Partitioning:** Helps to make better usage of resources
- Encapsulation:** VMs are basically a set of files that can be easily copied or managed and moved around the data center.
- Isolation:** Each VM is separated from the others. VMs think they own the hardware for themselves. Isolation is not only between VMs, but is also abstract-

ing the hardware underneath—which makes VMs portable across different hardware vendors.

Intel VT FlexMigration and Extended vMotion remove barriers to managing a virtualized data center and give CIOs investment protection across many years.

Depending on the type of hypervisor, version compatibility may be different across CPU generations. Also, the baseline instruction set may vary.

Intel VT FlexMigration, used in conjunction with other hardware assist features and hypervisors, can also help enable cloud computing.

To learn more about Intel® Virtualization Technology FlexMigration, visit www.intel.com/business.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web site at www.intel.com.

Copyright © 2010 Intel Corporation. All rights reserved. Intel, the Intel logo, and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others. Printed in USA SS/MR/0510 Please Recycle

