Enterprise Java Applications on VMware
Best Practices Guide

vmware®

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

Contents

1. Introduction

1.1 Overview

This guide provides information about best practices for deploying enterprise Java applications on VMware, including key best practice considerations for architecture, performance, design and sizing, and high availability. This information helps IT architects successfully deploy and run Java environments on VMware vSphere®.

1.2 Purpose

This guide provides best practice guidelines for deploying enterprise Java applications on VMware vSphere. The recommendations in this guide are not specific to any particular set of hardware or to the size and scope of any particular implementation. These best practices provide guidance only. They do not represent strict design requirements because enterprise Java application requirements can vary from one implementation to another. However, the guidelines do form a good foundation on which you can build—many VMware customers have used these guidelines to successfully virtualize their enterprise Java applications. This document focuses on details that pertain to the deployment of enterprise Java applications on VMware vSphere and it is not necessarily a best practice guide for pure Java. For specific Java best practices, refer to the vendor documentation for the JVM that you are using.

Virtualizing enterprise Java applications does not require a change in your Java coding paradigm. Any performance enhancements that you have done on physical are transferrable as is to the vSphere deployed instance of your application.

1.3 Target Audience

This guide assumes a basic knowledge and understanding of VMware vSphere and enterprise Java applications.

- Architectural staff can use this document to gain an understanding of how the system works as a whole as they design and implement various components.
- Engineers and administrators can use this document as a catalog of technical capabilities.

1.4 Scope

This guide covers the following topics:

- Enterprise Java Applications on vSphere Architecture – Provides a high-level best practice architecture for running enterprise Java applications on vSphere.
- Enterprise Java Applications on vSphere Best Practices – Provides best practice guidelines for properly preparing the vSphere platform to run enterprise Java applications on vSphere. Best practices for the design and sizing of virtual machines, guest OS tips, CPU, memory, storage, networking, and useful JVM tuning parameters are presented. Also covered are the various high availability features in vSphere, including VMware ESX®/VMware ESXi™ host clusters, resource pools (*horizontal scalability* and *vertical scalability*) along with the VMware vSphere Distributed Resource Scheduler (DRS).
- Enterprise Java on vSphere Troubleshooting Primer – There are times when you must troubleshoot a particular Java application problem when running on vSphere. The vSphere esxtop utility is very informative when troubleshooting.
- Enterprise Java Application on vSphere FAQ – Answers to frequently asked questions about the deployment of the enterprise Java applications on vSphere.

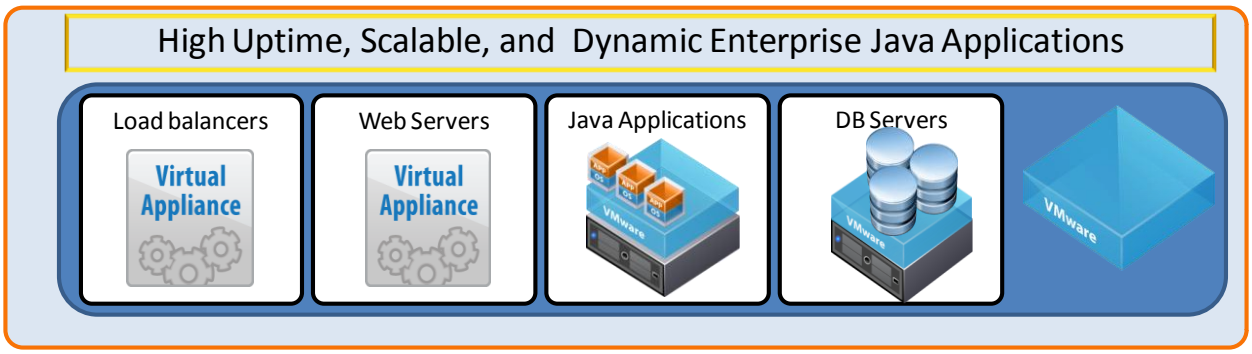
2. Enterprise Java Applications on vSphere – Architecture

Enterprise Java applications are made of four main tiers. These tiers are the:

- Load Balancers tier
- Web Servers tier
- Java Application Server tier
- Database Server tier

A highly scalable and robust Java application has all of these tiers running in vSphere to achieve the full benefits of the scalability features offered by vSphere. Figure 1 shows a multitier, fully virtualized enterprise Java application architecture running on vSphere.

Figure 1. Multitier Virtualized Enterprise Java Application Architecture



Each of these tiers is running in a virtual machine that is managed by vSphere, which forms the foundation. Best practices are discussed in this guide for vSphere features such as VMware vSphere High Availability (HA), DRS, VMware vSphere® vMotion®, resource pools, hot plug/hot add, networking and storage. The following are key architectural attributes of each tier:

- Load Balancer tier – Increasingly feature-rich load balancers are available which provide various load balancing algorithms and API integration with vSphere. This allows the enterprise Java application architecture to scale on demand as traffic bursts occur.
- Web Server tier – The Web Server tier must be appropriately tuned, with the correct number of HTTP threads to service anticipated traffic demands.
- Java Application Server tier – Many commonly used application servers have mechanisms to help tune the JVM to meet traffic demands. If you have already tuned the available Java threads, JDBC configurations, and various JVM and GC parameters on physical machines, the tuning information is transferrable as is to the vSphere deployment of enterprise Java applications.
- Database Server tier – It is critical to meeting the uptime SLA of enterprise Java applications to have the appropriate high availability architecture for the DB server. DB servers can benefit from running on vSphere (see the best practices for your DB server on vSphere). This guide covers best practices for JDBC Connection Pool tuning at the Java application server.

3. Enterprise Java Applications on vSphere – Best Practices

3.1 Virtual Machine Sizing and Configuration Best Practices Overview

Enterprise Java applications are highly customizable, so consequently, a performance test must be conducted to establish best sizing.

It is a best practice to establish the size of your virtual machine in terms of vCPU, memory, and number of JVMs by conducting a thorough performance test that mimics your production workload profile. The resulting virtual machine from the vertical scalability (scale-up) performance test is referred to as the *building block VM*. The building block virtual machine is a good candidate template on which scaled-out (horizontally scaled) virtual machines can be based.

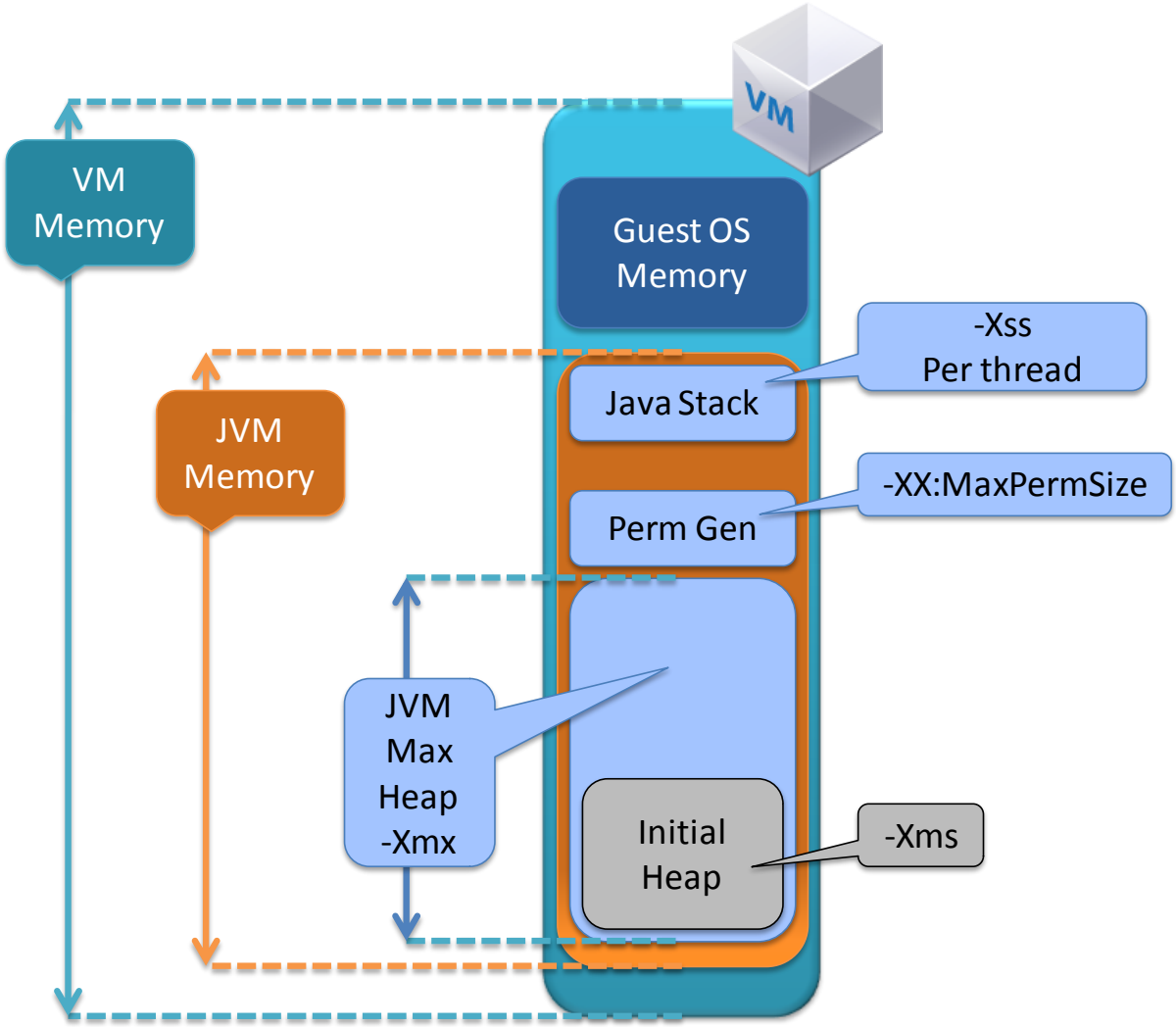
3.2 vCPU for Virtual Machines Best Practices

Best Practice	Description
BP1 VM sizing and VM-to-JVM ratio through a performance load test	Establish a workload profile and conduct a load test to measure how many JVMs you can stack on a particularly sized virtual machine. In this test, establish a best case scenario of how many concurrent transactions you can push through a configuration before it can be safely deemed a good candidate for scaling horizontally in an application cluster.
BP2 VM vCPU CPU overcommit	For performance-critical enterprise Java applications virtual machines in production, make sure that the total number of vCPUs assigned to all of the virtual machines does not cause greater than 80% CPU utilization on the ESX/ESXi host.
BP3 VM vCPU. Do not oversubscribe to CPU cycles that you do not really need.	• For example, if your performance load test determines that 2 vCPU is adequate up to 70% CPU utilization, but instead you allocate 4 vCPU to your virtual machine, then potentially there can be 2 vCPUs idle, which is not optimal. • If the exact workload is not known, size the virtual machine with a smaller number of vCPUs initially and increase the number later if necessary.

3.3 Virtual Machine Memory Size Best Practices

To understand how to size memory for a virtual machine you must understand the memory requirements of Java and various segments within it. Figure 2 provides an illustration of these separate memory areas.

Figure 2. Single JVM Deployed on One VM



VM Memory = Guest OS Memory + JVM Memory

JVM Memory = JVM Max Heap (`-Xmx`* value)* *+ JVM Perm Gen* (`-XX:`*MaxPermSize)*

*+ NumberOfConcurrentThreads * (*`-Xss`*)*

In Figure 2, the following terms are used:

- *Perm Gen* is an area that is in addition to the `-Xmx` (*Max Heap*) value; it holds the class level information about the code. IBM JVMs do not have a Perm Gen area.
- The above VM Memory formula is an approximation of the main areas allocated. To more accurately size you must load test the Java application for additional memory that may be allocated due to NIO buffers, JIT code cache, classloaders, and verifiers. In particular, some Java applications use NIO buffers, which can add huge memory demands.
- The contents of a direct buffer are allocated from the guest operating system memory instead of the Java heap, and non-direct buffers are copied into direct buffers for native I/O operations. Use load testing to appropriately size the effect of these buffers.
- If you have multiple JVMs (*N* JVMs) on a VM, then *VM memory = guest OS memory + N * JVM memory*.

Best Practice	Description
BP4 VM memory sizing	• Whether you are using Windows or Linux as your guest OS, refer to the technical specification of your vendor for memory requirements. It is common to see the guest OS allocated about 0.5GB to 1GB in addition to the JVM memory size. However, each installation may have additional processes running on it, for example, monitoring agents, and you must also accommodate their memory requirements. Figure 2 shows the various segments of JVM and virtual machine memory, and the formula summarizes virtual machine memory as: *VM Memory (needed) = guest OS memory + JVM Memory* where *JVM Memory = JVM Max Heap (*`-Xmx` *value) + Perm Gen (*`-XX`*:MaxPermSize) + NumberOfConcurrentThreads * (*`-Xss`*)* • The `-Xmx` value is the value that you determined during load testing for your application on physical servers. This value does not have to change when moving to a virtualized environment. Load testing your application when deployed on vSphere helps to confirm the best `-Xmx` value. • It is recommended that you do not overcommit memory because the JVM memory is an active space where objects are constantly being created and garbage collected. Such an active memory space requires its memory to be available all the time. If you overcommit, memory ballooning or swapping may occur and impede performance. • An ESX/ESXi host employs two distinct techniques for dynamically expanding or contracting the amount of memory allocated to virtual machines. The first method is known as memory *balloon driver* (vmmemctl). This is loaded from the VMware Tools package into the guest operating system running in a virtual machine. The second method involves paging from a virtual machine to a server swap file without any involvement by the guest operating system. In the page swapping method, when you power on a virtual machine, a corresponding swap file is created and placed in the same location as the virtual machine configuration file (`VMX` file). The virtual machine can power on only when the swap file is available. ESX/ESXi hosts use swapping to forcibly reclaim memory from a virtual machine when no balloon driver is available. The balloon driver might be unavailable either because VMware Tools is not installed or because the driver is disabled or not running. For optimal performance, ESX/ESXi uses the balloon approach whenever possible.

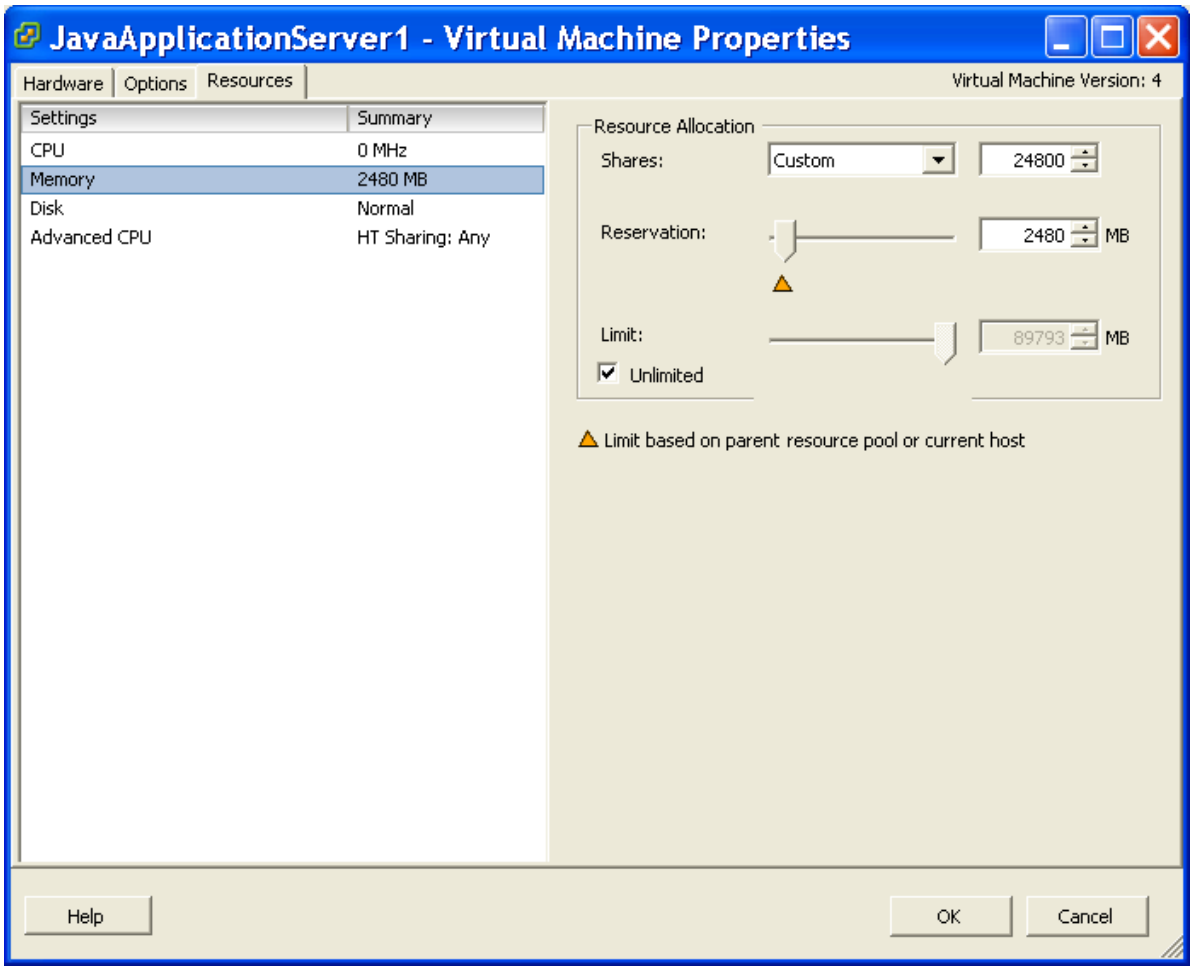
	However, swapping is used when the driver is temporarily unable to reclaim memory quickly enough to satisfy current system demands. Because the memory is being swapped out to disk, there is a significant performance penalty when the swapping technique is used. Therefore, it is recommended that the balloon driver is always enabled, but monitor it to verify that it is not being invoked when that memory is overcommitted. • Both ballooning and swapping should be prevented for Java applications. To prevent ballooning and swapping, refer to BP5 – Set memory reservation for virtual machine needs.
BP5 Set memory reservation for virtual machine memory needs	• JVMs running on virtual machines have an active heap that must always be present in physical memory. Use the VMware vSphere® Client™ to set the reservation equal to the needed virtual machine memory. *Reservation Memory = VM Memory = guest OS Memory + JVM Memory* • If for example, you have a 4GB heap, then it is likely that the JVM Memory is approximately 4.5GB, with another 0.5GB needed for guest OS. Therefore, the total of virtual machine memory needed is 5GB, so a memory reservation of 5GB must be configured for the virtual machine. **Note** This is an example only. Conduct your sizing and load testing exercise to verify the appropriate sizing for your workload. Likely this type of virtual machine needs 2 vCPUs as your starting point, then adjust according to the workload behavior.
BP6 Use large memory pages	• Large memory pages help performance by optimizing the use of the translation look-aside buffer (TLB), where virtual to physical address translations are performed. Use large memory pages as supported by your JVM and your guest operating system. The operating system and the JVM must be aware that you want to use large memory pages, as is the case when using large pages in physical systems. • Set the `-XX:+UseLargePages` at the JVM level for Sun HotSpot. • On the IBM JVM, it is `-Xlp`, on JRockit, `-XXlargePages`. • You must also enable this at the guest OS level. For more information, see *Large Page Performance: ESX Server 3.5 and ESX Server 3i v3.5* at http://www.vmware.com/files/pdf/large_pg_performance.pdf. • When sizing memory for large pages to be consumed by the JVM, you must leave a certain amount of small memory pages for other processes that cannot use large pages.

3.3.1 BP5 – How to Set Memory Reservation

Set the memory reservation value in the vSphere Client to the size of memory for the virtual machine. In Figure 3, the memory reservation is set to 2480MB. This virtual machine is always allocated this amount of memory on any ESX/ESXi host on which it runs.

To set the memory reservation, select the virtual machine, right-click, and select the **Edit Settings** > **Resources** tab. Figure 3 shows how to set the memory reservation for a virtual machine in the vSphere Client.

Figure 3. Setting the Memory Reservation



3.4 Virtual Machine Timekeeping Best Practices

Best Practice	Description
BP7 Timekeeping Use NTP source	• Timekeeping can be different in virtual machines than on physical machines for a variety of reasons. See *Timekeeping in VMware Virtual Machines: VMware ESX 4.0/ESXi 4.0, VMware Workstation 7.0* at http://www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf. • Timekeeping can have an effect on Java programs if they are sensitive to accurate measurements over periods of time, or if they require a timestamp that is within an exact tolerance (such as a timestamp on a shared document or data item). VMware Tools contains features that are installable on the guest operating system to enable time synchronization. Use of those tools is recommended. The frequency of timer interrupts can also affect the performance of your Java application.

3.4.1 Configuration for Timekeeping Best Practices

- Synchronize the time on the ESX/ESXi host with an NTP source. See Timekeeping in VMware Virtual Machines: VMware ESX 4.0/ESXi 4.0, VMware Workstation 7.0 (http://www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf).
- Synchronize the time in the virtual machine's guest operating system:
  - For Linux guest operating systems use an external NTP source. See the preceding reference.
  - For Windows guest operating systems use W32Time. Refer to your Windows administration guide for detailed information.

Lower the clock interrupt rate on the virtual CPUs in your virtual machines by using a guest operating system that allows lower timer interrupts. Examples of such operating systems are RHEL 4.7 and later, RHEL 5.2 and later, and SUSE Linux Enterprise Server 10 SP2. For more information on timekeeping best practices for Linux, see *Timekeeping best practices for Linux guests* at http://kb.vmware.com/kb/1006427.

- Use the Java features for lower resolution timing that are supplied by your JVM, such as the option for the Sun JVM on Windows guest operating systems:

  `-XX:+ForceTimeHighResolution`

- You can also set the `_JAVA_OPTIONS` variable to this value on Windows operating systems using the technique given (useful in cases where you cannot easily change the Java command line).
- The following is an example of how to set the Sun JVM option. To set the `_JAVA_OPTIONS` environment variable:

Select **Start** > **Settings** > **Control Panel** > **System** > **Advanced** > **Environment Variables**.

Select **New** under **System Variables**. The variable name is `_JAVA_OPTIONS`. The variable value is `-XX:+ForceTimeHighResolution`.

Restart the guest operating system to propagate the variable.

- For Windows guest operating systems that use an SMP HAL, avoid using the `/usepmtimer` option in the `boot.ini` system configuration.

3.5 Vertical Scalability Best Practices

If an enterprise Java application deployed on vSphere experiences heavy CPU utilization and you have determined that an increase in the vCPU count can help resolve the situation, use vSphere hot add to add additional vCPU.

Best Practice	Description
BP8 Hot add CPU/memory	• Virtual machines with a guest OS that supports hot add CPU and hot add memory can take advantage of the ability to change the virtual machine configuration at runtime without any interruption to virtual machine operations. This is particularly useful when you are trying to increase the ability of the virtual machine to handle more traffic. • Plan ahead and enable this feature. The virtual machine must be turned off to have the hot plug feature enabled, but after enabling it you can hot add CPU and hot add memory at runtime without virtual machine shutdown (if the guest OS supports it). • See Section 3.3, Virtual Machine Memory Size Best Practices. • When you need to increase Java heap space, you typically must increase vCPU count to get the best GC cycle performance. There are many other aspects to GC tuning, so refer to your JVM documentation.

3.6 Horizontal Scalability, Clusters, and Pools Best Practices

Enterprise Java applications deployed on VMware vSphere can benefit from using vSphere features for horizontal scalability using ESX/ESXi host clusters, resource pools, host affinity and DRS.

Best Practice	Description

BP9
Use ESX/ESXi host cluster
	• To enable better scalability, use ESX/ESXi host clusters.
	• When creating clusters, enable VMware HA and VMware DRS:
	  ○ VMware HA – Detects failures and provides rapid recovery for the VM running in a cluster. Core functionality includes host monitoring and VM monitoring to minimize downtime.
	  ○ VMware DRS – Enables VMware vCenter Server™ to manage hosts as an aggregate pool of resources. Cluster resources can be divided into smaller pools for users, groups, and virtual machines. It enables VMware vCenter™ to manage the assignment of virtual machines to hosts automatically, suggesting placement when virtual machines are powered on, and migrating running virtual machines to balance loads and enforce allocation policies.
	• Enable EVC (for either Intel or AMD). EVC is c; it configures a cluster and its hosts to maximize vMotion compatibility. When EVC is enabled, only hosts that are compatible with those in the cluster may be added to the cluster.

BP10 Use resource pools	Multiple resource pools can be used within a cluster to manage compute resource consumption by either reserving the needed memory for the virtual machines within a resource pool or by limiting/restricting it to a certain level. This feature also helps you meet quality of service and requirements. For example, you can create a Tier 2 resource pool for the less critical applications and a Tier 1 resource pool for business critical applications.
BP11 Affinity rules	The VM-Host affinity rule provides the ability to place virtual machines on a subset of hosts in a cluster. This is useful in honoring ISV licensing requirements. Rules can be created so that virtual machines run on ESX/ESXi hosts in different blades for higher availability. Conversely, you can limit the ESX/ESXi host to one blade in case network traffic between the virtual machines must be optimized by keeping them in one chassis location.
BP12 Use vSphere-aware load balancers	vSphere makes it easy to add resources such as host and virtual machines at runtime. It is possible to provision these ahead of time. However, it is simpler if you use a load balancer that is able to integrate with vSphere APIs to detect the newly added virtual machines and add them to its application load balancing pools without downtime.

3.7 Inter-Tier Configuration Best Practices

The configuration for compute resources at each of the four critical technology tiers (Load Balancer tier, Web Server tier, Java Application Server tier, and Database Server tier) that are layered on vSphere must translate to an equitable configuration at the next tier. For example, if the Web Server tier is configured to handle 100 HTTP requests per second, then of those requests you must determine how many Java application server threads are needed, and in turn how many DB connections are needed in the JDBC pool configuration.

Best Practice	Description
BP13 Establish appropriate thread ratios that prevent bottlenecks (HTTP threads:Java threads:DB connections)	• This is the ratio of HTTP threads to Java threads to database connections. • Establish initial setup by assuming that each layer requires a 1:1:1 ratio of HTTP threads:Java Threads:DB-connections, and then based on the response time and throughput numbers, adjust each of these properties until you have satisfied your SLA objectives. For example, if you initially submit 100 HTTP requests to the Web Server, assume that all of these will have an interaction with Java threads, and in turn, DB connections. During the benchmark testing, you will likely find that not all HTTP threads are submitted to the Java application server, and in turn, not all Java application server threads each require a DB connection. So the ratio for 100 requests translates to 100 HTTP threads:25 Java threads:10 DB connections, and this depends on the nature of your enterprise Java application behavior. Benchmarking helps to establish this ratio.

BP14 Load balancer algorithm choice and virtual machine symmetry	• Know the available algorithms of your load balancer. Make sure that when using the scale-out approach that all of your virtual machines are receiving an equal share of the traffic. Some industry standard algorithms are *round robin*, *weighted round robin*, *least connections*, and *least response time*. You might want to initially default to least connections and then adjust in your load test iterations. • Keep the size of compute resources in your virtual machines symmetrical. For example, using 2 vCPU virtual machines as a repeatable, horizontally scalable building block might help the load balancing algorithm work more effectively, as opposed to a pool of non-symmetrical virtual machines for one particular application. That is, mixing 2 vCPU virtual machines with 4 vCPU virtual machines in one load balancer-facing pool is non-symmetrical and the load balancer has no awareness of weighing this unless you configure for it at the Load Balancer level. This is time consuming.

3.8 High-Level vSphere Best Practices

The following sections summarize some of the key networking, storage and hardware-related best practices that are commonly used.

3.8.1 vSphere Networking Best Practices

Best Practice	Description
BP15 vSphere networking	Follow vSphere networking best practices. In addition to *Performance Best Practices for VMware vSphere 4.0* at http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.0.pdf, and *Performance Best Practices for VMware vSphere 5.0* at http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.0.pdf, see VMworld *Virtual Networking Concepts and Best Practices* at http://www.vmworld.com/docs/DOC-5122.

3.8.2 vSphere Storage Best Practices

Best Practice	Description
BP16 vSphere storage	• VMware recommends a minimum of four paths from an ESX/ESXi host to a storage array, which means the host requires at least two HBA ports. • Follow vSphere storage best practices. For a detailed description of VMware storage best practices, see the *SAN System Design and Deployment Guide* at http://www.vmware.com/files/pdf/techpaper/SAN_Design_and_Deployment_Guide.pdf

3.8.3 vSphere Server Hardware Configuration Best Practices

Best Practice	Description
BP17 ESX/ESXi host hardware	• For hardware configuration best practices see *Performance Best Practices for VMware vSphere 5.0* at http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.0.pdf and for VMware vSphere 4.1 see *VMware vCenter Server Performance and Best Practices* at http://www.vmware.com/files/pdf/techpaper/vsp_41_perf_VC_Best_Practices.pdf. • Also see *VMware vSphere: The CPU Scheduler in VMware ESX 4.1* at http://www.vmware.com/files/pdf/techpaper/VMW_vSphere41_cpu_schedule_ESX.pdf. • Disable any other power-saving mode in the BIOS. • NUMA considerations – IBM (X-Architecture), AMD (Opteron-based), and Intel (Nehalem) non-uniform memory access (NUMA) systems are supported by ESX/ESXi. On AMD Opteron-based systems, such as the HP ProLiant DL585 Server, BIOS settings for node interleaving determine whether the system behaves like a NUMA system or like a uniform memory accessing (UMA) system. In NUMA architecture, the total memory on ESX/ESXi host is divided by the number of processor sockets. • By default, ESX/ESXi NUMA scheduling and related optimizations are enabled only on systems with a total of at least four CPU cores and with at least two CPU cores per NUMA node. • Virtual machines with a number of vCPUs equal to or less than the number of cores in each NUMA node are managed by the NUMA scheduler and have the best performance. • Hardware BIOS – Verify that the BIOS is set to enable all populated sockets, and enable all cores in each socket. • Enable Turbo Mode if your processors support it. • Make sure that hyper-threading is enabled in the BIOS.

3.8.4 vSphere Configuration for Latency-Sensitive Workloads

Best Practice	Description
BP18 Latency-sensitive best practices	Workloads that are sensitive to response times in the range of 10's of microseconds will benefit from the best practices discussed in *Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs* at http://www.vmware.com/resources/techresources/10220.

4. Troubleshooting Primer

Investigate vSphere and each of the four tiers when troubleshooting enterprise Java applications problems. Note that vSphere has a dependency on networking and storage, which you might also have to investigate. The next few sections provide information about how to begin troubleshooting, and describe effective utilities you can use.

4.1 Opening a Support Request Ticket

If you suspect the VMware vSphere is not configured optimally and is cause of a performance problem, file a support request at http://www.vmware.com/support/contacts/file-sr.html. In addition, you might want to:

- Follow the troubleshooting steps outlined in the Performance *T*roubleshooting *G*uide for *VMware vSphere 4 and* ESX4.0 at http://communities.vmware.com/docs/DOC-10352.
- Verify that you have applied all of the best practices discussed in this guide.
- Run the `vm-support` utility. Execute the following command at the service console:

  ```
  vm-support -s
  ```

  This collects necessary information so that VMware can help diagnose the problem. It is best to run this command at the time the symptoms occur.

4.2 Troubleshooting Techniques for vSphere with esxtop

4.2.1 Performance Guides and References

- *Performance Troubleshooting for VMware vSphere 4 and ESX 4.0*
  http://communities.vmware.com/docs/DOC-10352
- *Interpreting esxtop 4.1 Statistics*
  http://communities.vmware.com/docs/DOC-11812

4.2.2 esxtop Primer

See *Interpreting esxtop 4.1 Statistics* at http://communities.vmware.com/docs/DOC-11812 for detailed information about metrics. The following table summarizes some of the metrics you can use when troubleshooting.

Display	Metric	Threshold	Description
CPU	%RDY	10	Over-provisioning of vCPU, excessive usage of vSMP or a limit (check %MLMTD) has been set. This %RDY value is the sum of all vCPUs %RDY for a virtual machine. For example, if the maximum value of %RDY of 1 vCPU is 100% and 4 vCPU is 400%. If %RDY is 20 for 1 vCPU then this is problematic, as it means 1 vCPU is waiting 20% of the time for VMkernel to schedule it.
CPU	%CSTP	3	Excessive usage of vSMP. Decrease amount of vCPUs for this particular virtual machine.

Display	Metric	Threshold	Description
CPU	%MLMTD	0	If larger than 0, then *worlds* are being throttled. Possible cause is a limit on CPU. (A *world* is an ESX Server VMkernel schedulable entity, similar to a process or thread in other operating systems.)
CPU	%SWPWT	5	Virtual machine waiting on swapped pages to be read from disk. Memory might be overcommitted.
MEM	MCTLSZ	1	If larger than 0, host is forcing the virtual machine to inflate the balloon driver to reclaim memory as the host is overcommitted.
MEM	SWCUR	1	If larger than 0, the host has swapped memory pages in the past. You might have overcommitted.
MEM	SWR/s	1	If larger than 0, the host is actively reading from swap. This is caused by excessive memory overcommitment.
MEM	SWW/s	1	If larger than 0, the host is actively writing to swap. This is caused by excessive memory overcommitment.
MEM	N%L	80	If less than 80, virtual machine experiences poor NUMA locality. If a virtual machine has memory size greater than the amount of memory local to each processor, the ESX/ESXi scheduler does not attempt to use NUMA optimizations for that virtual machine.
NETWORK	%DRPTX	1	Dropped packages transmitted, hardware is overworked due to high network utilization.
NETWORK	%DRPRX	1	Dropped packages received, hardware is overworked due to high network utilization.
DISK	GAVG	25	Look at DAVG and KAVG as GAVG = DAVG + KAVG.
DISK	DAVG	25	At this level, there is disk latency that is likely caused by storage array.
DISK	KAVG	2	Disk latency caused by the VMkernel. High KAVG usually means queuing. Check QUED.
DISK	QUED	1	Queue has maxed out. Possibly queue depth is set too low. Check with array vendor for optimal queue value.
DISK	ABRTS/s	1	Aborts issued by virtual machine because storage is not responding. For Windows virtual machines this happens after 60-second default. Can be caused by path failure, or storage array is not accepting I/O.
DISK	RESET/s	1	The number of commands resets per second.

4.3 Java Troubleshooting Primer

Refer to your JVM documentation for troubleshooting guides.

The following sections provide information about how to begin troubleshooting. Information is given about some severe Java application problems which are GC/memory leakage-related, and some that are thread contention-based. For JDBC-based errors, refer to the JDBC driver provided to you by the database vendor. Of particular importance for performance are errors leading to OutofMemory, Stackoverflow, and Thread Deadlock.

4.3.1 Java Memory Problem Troubleshooting

Consider an example where you observe load increases and decreases over time. If memory continues to build without reclamation (in the worst case), or a GC reclamation occurs but not everything is reclaimed, there may be a memory leak. This is very likely if these symptoms persist to where the application suffers from an OutofMemory error. In this case, investigate the GC frequency and setting.

- To turn on GC verbose mode:
  - `verbose:gc` – Prints basic information about GC to the standard output.
  - `-XX:+PrintGCTimeStamps` – Prints the times that GC executes.
  - `-XX:+PrintGCDetails` – Prints statistics about different regions of memory in the JVM.
  - `-Xloggc:<`*`file`*`>` – Logs the results of GC in the specified file.
- Re-inspect the `-Xmx`, `-Xms`, `-Xss` settings.
- If you are using JDK 6, use the `jmap` tool on any platform. Running `jmap` can add additional load on your environment, so plan for the best time to run it.
- If you are using JDK 5:
  - If you are running Linux with JDK 5, use `jmap`.
  - If you are using JDK 5 update 14 or later, use the `-XX:+HeapDumpOnCtrlBreak` option when starting JVM, then use the Ctrl+Break key combination on Windows to dump the heap.

4.3.2 Java Thread Contention Problem

If you suspect that your enterprise Java application is suffering from long pauses, or just has general response time issues to the point where the JVM needs to be restarted to resolve the issue, you might need also to inspect the Java thread dump. You can obtain a Java thread dump by pressing Ctrl+Break for a Windows OS or in Linux by issuing `kill -3` on the Java process ID. Take the thread dump at the point where problematic symptoms appear. This is especially true if you are conducting a benchmark load test—take the thread dump at maximum peak load and inspect the behavior of the various application threads.

There are many widely used thread analysis tools that interpret the thread dump and highlight in red the hot threads or threads waiting for a lock. You can begin your code investigation from that point and follow the call stack.

5. Enterprise Java Applications on vSphere – FAQ

Are there any performance papers and other references that discuss Java workloads running on VMware?

See Section 6, *References*.

With UNIX-based hardware I have very large machines running all of my Java applications. What should be my migration sizing strategy and what are the VMware vSphere maximums that I need to know about?

- One of the most important steps is to conduct a load test to help you determine the ideal individual virtual machine size and how many JVMs you can stack up (vertical scalability). Based on this repeatable building block virtual machine you can scale out to determine what is best for your application traffic profile.
- Know the VMware vSphere maximums. See *Configuration Maximums for vSphere 5.0* at https://www.vmware.com/pdf/vsphere5/r50/vsphere-50-configuration-maximums.pdf and *Configuration Maximums: VMware vSphere 4.1* at http://www.vmware.com/pdf/vsphere4/r41/vsp_41_config_max.pdf.

The following table provides a summary of maximums per virtual machine, per Host, and per vCenter.

vSphere Configuration	Maximum for vSphere 5.0
Per VM	• 32 vCPUs • 1TB • 2TB of storage minus 512 bytes
Per Host	• 2048 vCPUs • 512 virtual machines • 25 vCPUs per core
Per vCenter	• 1000 hosts • 10000 powered on virtual machines • 15000 registered virtual machines • 10 linked vCenter servers • 3000 hosts in linked vCenter servers • 30000 powered on virtual machines in linked vCenter servers • 50000 registered virtual machines in linked vCenter servers • 100 concurrent vSphere clients • 500 hosts per datacenter

What decisions must be made due to virtualization?

Determine the optimal size of the repeatable building block virtual machine. Establish this by benchmarking, along with total scale-out factor. Determine how many concurrent users each single vCPU configuration of your application can handle. Then extrapolate that to your production traffic to determine overall compute resource requirements for vCPU, memory, storage, and networking. Having a symmetrical building block, for example every virtual machine having the same number of vCPUs, helps keep load distribution from your load balancer even. The benchmarking tests help you determine how large a single virtual machine should be (*vertical scalability*) and how many virtual machines you will need (*horizontal scalability*).

Pay special attention to the scale-out factor and see up to what point it is linear within your application running on top of VMware. Enterprise Java applications are multitier and bottlenecks can occur at any tier during scale-out and quickly cause non-linear results. The assumption of linear scalability may not always be valid. It is essential to load test your intended configuration before going into production so that you have sized correctly.

I have conducted extensive GC sizing and tuning for our current enterprise Java application running on physical. Do I have to adjust any of these parameters when moving this Java application to a virtualized environment?

No. All tuning that you perform for your Java application on physical is transferrable to your virtual environment. However, because virtualization projects are typically about driving a high consolidation ratio, it is advisable that you follow the guidelines in *Enterprise Java Applications on VMware Design and Sizing Guidelines* to determine the ideal compute resource configuration for your individual virtual machines, the optimal number of JVMs within a virtual machine, and the overall number of virtual machines on the ESX/ ESXi host.

Additionally, because this type of migration involves an OS/platform change and possibly a JVM vendor change, it is advisable to review Section 2 of this document along with your vendor's tuning advice for both OS and JVM.

How many and what size of virtual machines will I need?

This depends on your application; however, 2 vCPU virtual machines are a common building block for Java applications. One of the guidelines from *Enterprise Java Applications on VMware Design and Sizing Guidelines* is to tune your system more for scale-out than scale-up. This is not an inflexible rule because it depends on your organization's architectural best practices. Smaller, more scaled-out virtual machines might provide better overall architecture, but can incur additional guest OS licensing costs. If this is a constraint, tune towards larger 4 vCPU virtual machines and stack more JVMs on them.

What is the correct number of JVMs per virtual machine?

- There is no one definitive answer because this largely depends on the nature of your application. The benchmarking you conduct can reveal the limit of the number of JVMs that can be stacked up on a single virtual machine.
- The more JVMs that you put on a single virtual machine, the more that JVM overhead/cost is incurred when initializing a JVM. Alternatively, instead of stacking up multiple JVMs within a virtual machine, increase the JVM size vertically by adding more threads and enlarging the heap. This can be achieved if your JVM is within an application server such as Apache Tomcat, so instead of increasing the number of JVMs, you can increase the number of concurrent threads available and resources that a single Tomcat JVM can service for the number of applications deployed and their concurrent requests per second. The limitation of how many applications you can stack up within a single application server instance/JVM is bounded by how large you can make your JVM heap and by the performance impact. A large heap (beyond 4GB) must be tested for performance and GC cycle impact, and you must examine the trade-offs. This concern is not specific to virtualization—it applies equally to physical server setup.

We have 200 JVMs running on a farm of virtual machines that have widely varying demand curves. Does VMware provide any tools to help me use the memory allocated to those virtual machines more efficiently?

Yes. Starting with version 2.6, VMware vFabric™ tc Server™ ships with technology called *Elastic Memory for Java* (EM4J), which allows Java workloads to overcommit memory on an ESX/ESXi host with fewer negative effects than in the past. This is a variant of *memory ballooning* specifically for Java applications.

Can I run EM4J with any Java program, such as Oracle WebLogic or IBM WebSphere?

No. EM4J is only available when running the vFabric tc Server application server, the VMware commercial version of the Apache Tomcat application server.

How do I know if EM4J will benefit my application in its current virtualized configuration?

The only real way to know is by trying it, but in general, EM4J works well when you have a number of Java application servers on a single ESX/ESXi host, each one with a varying load cycle that is out of phase with the others, such as servers serving different geographic territories. Servers for the geographies where it is currently daytime will have a higher load and take memory from the servers serving geographies where it is nighttime.

Is EM4J limited to a particular version of vSphere?

Yes. You must be on vSphere 5 or later to use EM4J.

Where can I learn more about EM4J?

- The VMware vFabric tc Server product documentation at http://pubs.vmware.com/vfabric5/index.jsp?topic=/com.vmware.vfabric.tc-server.2.6/em4j/about.html
- Videos are available on YouTube at http://www.youtube.com/watch?v=kyz7J-FQUSM http://www.youtube.com/watch? v=C2Z5-HPYIEM

6. References

The following is a list of available performance papers and other references:

- *Performance of Enterprise Java Applications on VMware vSphere 4.1 and SpringSource tc Server*
  http://www.vmware.com/resources/techresources/10158
- Harold Rosenberg's Performance Blog
  http://communities.vmware.com/blogs/haroldr
- *Large Page Performance*
  http://www.vmware.com/resources/techresources/1039
- *vMotion Architecture, Performance, and Best Practices in VMware vSphere 5*
  http://www.vmware.com/resources/techresources/10202
- *High Performance Data with VMware vFabric GemFire Best Practices Guide*
  http://www.vmware.com/resources/techresources/10231
- *Performance Brief for IBM WebSphere Application Server 7.0 with VMware ESX 4 on HP ProLiant DL380 G6 Servers*
  http://www.vmware.com/resources/techresources/10095
- *Enterprise Java Applications Architecture on VMware* book
  https://www.createspace.com/3632131
- *Configuration Maximums: VMware vSphere 4.1*
  http://www.vmware.com/pdf/vsphere4/r41/vsp_41_config_max.pdf
- *Configuration Maximums for vSphere 5.0*
  https://www.vmware.com/pdf/vsphere5/r50/vsphere-50-configuration-maximums.pdf
- *Performance Best Practices for VMware vSphere 4.0*
  http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.0.pdf
- *Performance Best Practices for VMware vSphere 5.0*
  http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.0.pdf
- *vCenter Server Performance and Best Practices Guide*
  http://www.vmware.com/files/pdf/techpaper/vsp_41_perf_VC_Best_Practices.pdf
- *Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs*
  http://www.vmware.com/resources/techresources/10220
- *Performance Troubleshooting for VMware vSphere 4 and ESX 4.0*
  http://communities.vmware.com/docs/DOC-10352
- *VMware vSphere: The CPU Scheduler in VMware ESX 4.1* at
  http://www.vmware.com/files/pdf/techpaper/VMW_vSphere41_cpu_schedule_ESX.pdf
- *Interpreting esxtop 4.1 Statistics*
  http://communities.vmware.com/docs/DOC-11812
- VMware vFabric tc Server product documentation
  http://pubs.vmware.com/vfabric5/index.jsp?topic=/com.vmware.vfabric.tc-server.2.6/em4j/about.html
- *Large Page Performance: ESX Server 3.5 and ESX Server 3i v3.5*
  http://www.vmware.com/files/pdf/large_pg_performance.pdf
- *Timekeeping in VMware Virtual Machines: VMware ESX 4.0/ESXi 4.0, VMware Workstation 7.0*
  http://www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf
- *Timekeeping best practices for Linux guests*
  http://kb.vmware.com/kb/1006427

- VMworld *Virtual Networking Concepts and Best Practices*
  http://www.vmworld.com/docs/DOC-5122
- *SAN System Deployment and Design Guide*
  http://www.vmware.com/files/pdf/techpaper/SAN_Design_and_Deployment_Guide.pdf
- YouTube videos
  http://www.youtube.com/watch?v=kyz7J-FQUSM
  http://www.youtube.com/watch? v=C2Z5-HPYIEM