VMware vSphere Performance FAQ

Q Will auto-tiered storage work with Storage DRS?

A Not at this time for vSphere 5. If you are planning to use Storage DRS, you will have to disable auto-tiering on the storage. Also if you plan to use auto-tiering, do not enable Storage DRS. This may change in a future release.

Q Why does the vCenter memory utilization counter always look so high (with Large Pages and TPS)?

A When clients are using hardware that supports MMU, by default vSphere will use large memory pages (2MB in size) to back all guest memory first. So it's not until the memory allocator runs out of physical memory that it will start to use technologies like TPS (which uses 4KB memory pages) and ballooning; typically when about 92-94% of the memory is allocated. vCenter will report high memory utilization as all physical memory is used up.

Q What is a good DAVG threshold?

A The best way to keep the average SLA is to keep the DAVG under 10ms. If DAVG is consistently between 10-15ms, that is a sign that an issue may be starting. Above 15ms you will notice the slowness and it is time to troubleshoot the problem.

Q What is the difference in performance between physical RDM and vmdk on VMFS?

A The difference is negligible and you should not see a difference between physical RDM and vmdk on VMFS.

Q When and how should I adjust HIMP for VDI and XenApp workloads?

A For vSphere 5, you do not need to make adjustments. For vSphere 4.x and earlier servers that will be hosting VDI or TS workloads only, HIMP can be set to 0.

Q Are there scheduling issues with creating larger VMs?

A No. With the vast improvements made in the scheduler since vSphere 4, this is no longer an issue. VMware now uses a process called descheduling instead of having to support idle loops. Further details can be found in VMware vSphere 4: The CPU Scheduler in ESX 4.

Q What do we recommend for BIOS power management for tuning performance?

A With vSphere 4.x, selecting maximum performance is preferred for high performance workloads. As of vSphere 5, this can now be set to *OS controlled*, which allows vSphere 5 to manage the BIOS.

Q Is there a performance difference if I configure servers for different memory speeds (for example, 1033 vs 800)?

A The difference is negligible and, generally, you should pick capacity over performance (that is, it is best to have enough guest memory than to be short of it).

Q Does enabling EVC on a vSphere cluster affect performance?

A No. EVC masks out extended instructions that only pertain to multimedia type workloads.

Q Are there performance problems virtualizing Java applications?

A No. The performance has shown to be very good as long as best practices are followed and a true apples-to-apples comparison of physical vs. virtual environments is made. Further details can be found in the Enterprise Java Applications on VMware Best Practices Guide.

Q What is the effect of large pages with Java applications?

A 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 informed that you want to use large memory pages, as is the case when using large pages in physical systems.

Set `-XX:+UseLargePages` as the JVM level for Oracle HotSpot.

On the IBM JVM it is `-Xlp`, and JRockit `-XXlargePages`.

You also need to enable this at the guest OS level if the guest does not support transparent large pages.

Q Why does Java consume the entire memory within the heap on startup?

A This is no different in the physical world. A JVM will consume the memory you allocate to it, based on the initial `–Xms` heap memory plus whatever it needs for direct memory access. Java does it s own memory management whether virtualized or not, and it will continue to function the same way from a memory management perspective. To minimize the impact of statically configured Java process memory and Java heap memory, study the sizing formula provided in the

Enterprise Java Application on VMware – Best Practices Guide.

Q What do I do if I suspect I have a Java thread issue?

A It is important to take the thread dump right at the point when problematic symptoms appear. This is especially true if you are conducting a benchmark load test—take the thread dump at max 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.

Q If I over-commit CPU, how much can I over-commit it by?

A For performance-critical enterprise Java application VMs in production, make sure the total number of vCPUs assigned to all of the virtual machines does not cause greater than 80% CPU utilization on the ESX host. Do not over-subscribe to CPU cycles that you don't really need.

Q What host BIOS settings can apply for a Java application environment?

A Follow power management best practices mentioned in the Enterprise Java Application on VMware – Best Practices Guide.

Q Can we get EM4J for environments that do not use tc Server?

A No. Not at this time.

Q Will vMotion impact latency-sensitive application clusters?

A Many of our customers vMotion tier-1 Java workloads successfully, and have not reported specific vMotion issues. It is difficult to make a general statement about all workloads, and which of those workloads truly qualify as latency-sensitive. If you have an application that requires sensitivity to a few microseconds, then additional tuning to your infrastructure might be required to have vMotion not impact your system. Utilizing a 10GbE network and using VMXNET 3 drivers will be helpful. For further details, refer to Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs.

Q Do you have many customers running WebLogic or WebSphere in virtual machines and how does it perform?

A We have many large customers running both WebLogic and WebSphere very successfully.

Q What NUMA and vNUMA considerations should I be aware of? Should NUMA be disabled? Do I need to turn on –XX:UseNUMA?

A Size your VMs to be within NUMA nodes of RAM for optimal performance. The total server RAM is divided by the number of sockets/processors on your server machine, and hence each configured VM should be sized to fit within this. However, in ESXi 5, there is vNUMA support in addition to many NUMA performance enhancements that will help localize the VM to the NUMA node as much as possible. Due to the many enhancements of ESX NUMA locality mechanisms, we don't typically find customers setting `-XX:UseNUMA` because the ESX NUMA scheduler does a good job. The ESX NUMA scheduler doesn't replace guest NUMA for wide VMs, so configuring NUMA inside of guests can still be advantageous. You may want to conduct your own performance study to see the effects of `-XX:UseNUMA`.

Q How many and what size virtual machines will I need for a Java application environment?

A This depends on the nature of your application. But the overhead of running multiple JVMs under a single OS is essentially the same whether or not the JVMs run natively or in a VM. We most often see 2-vCPU VMs as a common building block for Java applications. One of the guidelines is to tune your system for more scale out as opposed to scale up. This rule is not absolute as it depends on your organization's architectural best practices. Smaller, more scaled-out VMs may provide better overall architecture, but you will incur additional guest OS licensing costs. If this is a constraint, then you can tune towards larger 4-vCPU VMs.

Q What is the correct number of JVMs per virtual machine?

A There is no definite answer. This largely depends on the nature of your application. The benchmarking you conduct can determine the limit of the number of JVMs that can be stacked up on a single VM.

The more JVMs you put on a single VM, the more JVM overhead/cost of initializing a JVM is incurred. Alternately, instead of stacking up multiple JVMs within a VM, you can increase the JVM size by adding more threads and heap size. This can be achieved if your JVM is within an application server such as Tomcat. Then, instead of increasing the number of JVMs, you can increase the number of concurrent threads available and resources to accurately size for your traffic such that a single Tomcat JVM services your n-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 afford your JVM heap size to be and performance. The trade-off of a very large JVM heap size beyond 4GB needs to be tested for performance and GC cycle impact. This concern is not specific to virtualization as it equally applies to physical server setup.

Q I have conducted extensive GC sizing and tuning for our current enterprise Java application running on physical machines. Do I have to adjust anything related to sizing when moving this Java application to a virtualized environment?

A No. All tuning that you would perform for your Java application on a physical machine is transferrable to your virtual environment. However, because virtualization projects are typically about driving a high consolidation ratio, it is advisable that you conduct adequate load testing to establish your ideal compute resource configuration for individual VMs, number of JVMs within a VM, and overall number of VMs on the ESX host.

Q What is a good approach to moving my distributed Enterprise Java application to a virtualized environment?

A You have to determine the size of the repeatable building block VM. This is established by benchmarking, along with the total scale-out factor. Determine how many concurrent users each single vCPU-configuration of your application can handle, and extrapolate that to your production traffic to determine the overall compute resource requirement. Having a symmetrical building block; for example, every VM having the same number of vCPUs, helps keep load distribution from your load balancer even. Essentially, your benchmarking test helps you determine how large a single VM should be (scaling up) and how many of these VMs you will need (scaling out).

You need to 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 multi-tier and bottlenecks can appear at any point along the scale-out performance line and quickly cause non-linear results. The assumption of linear scalability may not always be true, and it is essential to load test a pre-production replica (production to be) of your environment.

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

A One of the most important steps is to conduct a load test to help you determine the ideal individual VM size and how many JVMs you can stack up (scale up). Based on this repeatable building block VM, you can scale out to more VMs to determine what is best for your application traffic profile.

Review the VMware vSphere maximums. See Configuration Maximums: VMware vSphere 5.

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