Performance Best Practices for VMware Cloud on AWS

VMware Cloud on AWS

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

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

Contents

About this Book

This book, *Performance Best Practices for VMware Cloud on AWS*, provides performance tips that cover the most performance-critical areas of VMware Cloud® on AWS. It is not intended as a comprehensive guide for planning and configuring your deployments.

- Chapter 1 Hosts and Virtual Machines, provides guidance regarding VMware Cloud on AWS hosts and the virtual machines that run on them.
- Chapter 2 Guest Operating Systems, provides guidance regarding the guest operating systems running in vSphere virtual machines.
- Chapter 3 Virtual Infrastructure Management, provides guidance regarding infrastructure management best practices.

For planning purposes we recommend reading this entire book before beginning a deployment. Material in the Virtual Infrastructure Management chapter, for example, might influence your initial configuration choices.

Intended Audience

This book is intended for system administrators who are planning a VMware Cloud on AWS deployment and want to maximize its performance. The book assumes the reader is already familiar with VMware vSphere concepts and terminology.

Additional Documentation

The most current VMware Cloud on AWS documentation can be found on the VMware Docs site.

You can access performance and other technical papers on the VMware Technical Papers page.

Hosts and Virtual Machines 1

This chapter provides guidance regarding VMware Cloud on AWS hosts and the virtual machines that run on them.

This chapter includes the following topics:

VMware Cloud on AWS Host General Considerations

This section provides guidance regarding a number of general performance considerations in VMware Cloud on AWS hosts.

- Plan your deployment by allocating enough resources for all the virtual machines you will run, as well as those needed by VMware Cloud on AWS itself.
- Allocate to each virtual machine only as much virtual hardware as that virtual machine requires. Provisioning a virtual machine with more resources than it requires can, in some cases, reduce the performance of that virtual machine as well as other virtual machines sharing the same host.
- Unused or unnecessary virtual hardware devices can impact performance and should be deactivated.

  For example, Windows guest operating systems poll optical drives (that is, CD or DVD drives) quite frequently. When virtual machines are configured to use a physical drive, and multiple guest operating systems simultaneously try to access that drive, performance could be impacted. This impact can be reduced by configuring the virtual machines to use ISO images instead of physical drives, and can be avoided entirely by deactivating optical drives in virtual machines when the devices are not needed.

- VMware Cloud on AWS supports multiple virtual hardware versions. If you plan to move a virtual machine from VMware Cloud on AWS to an on-premises ESXi host (in a hybrid cloud scenario, for example), make sure the machine's virtual hardware version is supported by the ESXi hosts on which you intend to use them. You can see what's supported by each version of ESXi in VMware KB article 2007240.

Host CPU Considerations

This section provides guidance regarding CPU considerations in VMware VMware Cloud on AWS hosts.

CPU virtualization adds varying amounts of overhead depending on the percentage of the virtual machine's workload that can be run on the physical processor as is and the cost of virtualizing the remainder of the workload:

- For many workloads, CPU virtualization adds only a very small amount of overhead, resulting in performance essentially comparable to native.
- Many workloads to which CPU virtualization does add overhead are not CPU-bound—that is, most of their time is spent waiting for external events such as user interaction, device input, or data retrieval, rather than running instructions. Because in this case otherwise-unused CPU cycles are available to absorb the virtualization overhead, these workloads will typically have throughput similar to native, but potentially with a slight increase in latency.
- For a small percentage of workloads, for which CPU virtualization adds overhead and which are CPU-bound, there might be a noticeable degradation in both throughput and latency.

The rest of this section lists practices and configurations recommended by VMware for optimal CPU performance.

- In most environments VMware Cloud on AWS allows significant levels of CPU overcommitment (that is, running more vCPUs on a host than the total number of physical processor cores in that host) without impacting virtual machine performance.

  If a host becomes CPU saturated (that is, the virtual machines and other loads on the host demand all the CPU resources the host has), latency-sensitive workloads might not perform well. In this case Distributed Resource Scheduler (DRS) will attempt to migrate one or more virtual machines to another host if there's one available with sufficient resources. You might also want to reduce the CPU load (by powering off some virtual machines, for example).
- It is a good idea to periodically monitor the CPU usage of the host. This can be done through the vSphere Client, using the VMware vRealize® Operations™ management suite, or by using resxtop. Below we describe how to interpret resxtop data:
  - If the load average on the first line of the resxtop CPU panel is equal to or greater than 1, this indicates that the system is overloaded.

  - The usage percentage for the physical CPUs on the PCPU line can be another indication of a possibly overloaded condition. In general, 80% usage is a reasonable ceiling and 90% should be a warning that the CPUs are approaching an overloaded condition. However organizations will have varying standards regarding the desired load percentage.

  Note Although esxtop can't be used in VMware Cloud on AWS, resxtop can. See the blog post ESXTOP and VMware Cloud on AWS for guidance. For information about using resxtop see Performance Monitoring Utilities: resxtop and esxtop in *vSphere Monitoring and Performance*.

- Configuring a virtual machine with more virtual CPUs (vCPUs) than its workload can use might cause slightly increased resource usage, potentially impacting performance on very heavily loaded systems. Common examples of this include a single-threaded workload running in a multiple-vCPU virtual machine or a multi-threaded workload in a virtual machine with more vCPUs than the workload can effectively use.

  Even if the guest operating system doesn't use some of its vCPUs, configuring virtual machines with those vCPUs still imposes some small resource requirements that translate to real CPU consumption on the host. For example:

  - Maintaining a consistent memory view among multiple vCPUs can consume additional resources, both in the guest operating system and in the host.
  - Most guest operating systems run an idle loop during periods of inactivity. Within this loop, most of these guest operating systems halt by running the HLT or MWAIT instructions. Some very old guest operating systems, however, use busy-waiting within their idle loops. This results in the consumption of resources that might otherwise be available for other uses (other virtual machines, the VMkernel, and so on).

    VMware Cloud on AWS automatically detects these loops and de-schedules the idle vCPU. Though this reduces the CPU overhead, it can also reduce the performance of some I/O-heavy workloads. For additional information see VMware KB articles 1077 and 2231.
  - The guest operating system's scheduler might migrate a single-threaded workload amongst multiple vCPUs, thereby losing cache locality.

- Some workloads can easily be split across multiple virtual machines. In some cases, for the same number of vCPUs, using more smaller virtual machines (sometimes called "scaling out") will provide better performance than fewer larger virtual machines (sometimes called "scaling up"). In other cases the opposite is true, and fewer larger virtual machines will perform better. The variations can be due to a number of factors, including NUMA node sizes, CPU cache locality, and workload implementation details. The best choice can be determined through experimentation using your specific workload in your environment.

Hyper-Threading

This section provides guidance regarding hyper-threading in VMware Cloud on AWS.

- Hyper-threading technology (sometimes also called simultaneous multithreading, or SMT) allows a single physical processor core to behave like two logical processors, essentially allowing two independent threads to run simultaneously. Unlike having twice as many processor cores—that can roughly double performance—hyper-threading can provide anywhere from a slight to a significant increase in system performance by keeping the processor pipeline busier.

  Note Some processor types are subject to a concurrent-context attack vector security vulnerability when hyper-threading is active. VMware Cloud on AWS instances that use these processors are configured to use a special Side-Channel Aware Scheduler (SCAv1) to mitigate the vulnerability. When this scheduler is used, VMware Cloud on AWS will only schedule processes on one thread for each core, potentially reducing CPU capacity for some workloads.

  VMC i3en.metal hosts use Cascade Lake processors; these are not subject to this security vulnerability and will thus utilize both hyper-threads on each core for optimum performance.

  VMC i3.metal hosts use Broadwell processors; on these hosts hyper-threading is deactivated, potentially limiting performance when a host is close to full CPU utilization.

- On a system with hyper-threading activated, VMware Cloud on AWS assigns adjacent CPU numbers to logical processors on the same core. Thus CPUs 0 and 1 are on the first core, CPUs 2 and 3 are on the second core, and so on.

  VMware Cloud on AWS manages processor time intelligently to spread load smoothly across all physical cores in the system. If there is no work for a logical processor, it is put into a halted state that frees its execution resources and allows the virtual machine running on the other logical processor on the same core to use the full execution resources of the core.

Non-Uniform Memory Access (NUMA)

This section describes how to obtain the best NUMA performance from VMware Cloud on AWS.

Note A different feature, Virtual NUMA (vNUMA), allowing the creation of NUMA virtual machines, is described in Guest Operating System CPU Considerations.

More information about using NUMA systems can be found in the Using NUMA Systems with ESXi section of *vSphere Resource Management*. Though not specifically addressing VMware Cloud on AWS, most of the mechanisms detailed there are nevertheless relevant.

Manual NUMA Configuration

The intelligent, adaptive NUMA scheduling and memory placement policies in VMware Cloud on AWS can manage all virtual machines transparently, so that administrators don't need to deal with the complexity of balancing virtual machines between nodes by hand. Manual controls are available to override this default behavior, however, and advanced administrators might prefer to manually set NUMA placement (through the numa.nodeAffinity advanced option).

Virtual machines can be separated into the following two categories:

- Virtual machines with a number of vCPUs equal to or less than the number of cores in each physical NUMA node.

  These virtual machines will be assigned to cores all within a single NUMA node and will be preferentially allocated memory local to that NUMA node. This means that, subject to memory availability, all their memory accesses will be local to that NUMA node, resulting in the lowest memory access latencies.

- Virtual machines with more vCPUs than the number of cores in each physical NUMA node (called "wide virtual machines").

  These virtual machines will be assigned to two NUMA nodes and will be preferentially allocated memory local to those NUMA nodes. Because vCPUs in these wide virtual machines might sometimes need to access memory outside their own NUMA node, they might experience higher average memory access latencies than virtual machines that fit entirely within a NUMA node.

  Note This potential increase in average memory access latencies can be mitigated by appropriately configuring Virtual NUMA (described in Guest Operating System CPU Considerations), thus allowing the guest operating system to take on part of the memory-locality management task.

Because of this difference, there can be a slight performance advantage in some environments to virtual machines configured with no more vCPUs than the number of cores in each physical NUMA node.

Conversely, some memory bandwidth bottlenecked workloads can benefit from the increased aggregate memory bandwidth available when a virtual machine that would fit within one NUMA node is nevertheless split across multiple NUMA nodes. This split can be accomplished by limiting the number of vCPUs that can be placed per NUMA node by using the maxPerMachineNode option (do also consider the impact on vNUMA, however, which is described in Guest Operating System CPU Considerations).

As described in Hyper-Threading, i3en.metal instances have hyper-threaded enabled. Therefore on these systems virtual machines with a number of vCPUs greater than the number of cores in a NUMA node but lower than the number of logical processors in each physical NUMA node might benefit from using logical processors with local memory instead of full cores with remote memory. This behavior can be configured for a specific virtual machine with the numa.vcpu.preferHT flag.

VMware Cloud on AWS Memory Considerations

This section provides guidance regarding memory considerations in VMware Cloud on AWS.

Memory Sizing

This section describes choosing how much memory to allocate to your virtual machines.

Consider the following when selecting the amount of memory you allocate to your virtual machines:

- You should be sure to allocate enough memory to hold the working set of applications you will run in the virtual machine, thus minimizing thrashing. Because thrashing can dramatically impact performance, it is very important not to under-allocate memory.
- On the other hand, though the performance impact of over-allocating memory is far less than under-allocating it, you should nevertheless avoid substantially over-allocating as well.

  Memory allocated to a virtual machine beyond the amount needed to hold the working set will typically be used by the guest operating system for file system caches. If memory resources at the host level become low, VMware Cloud on AWS can generally use memory ballooning (described in Memory Overcommit Techniques) to reclaim the portion of memory used for these caches.

  But over-allocating memory also unnecessarily increases the virtual machine memory overhead. While VMware Cloud on AWS can typically reclaim the over-allocated memory, it can’t reclaim the overhead associated with this over-allocated memory, thus consuming memory that could otherwise be used to support more virtual machines.

Memory Overcommit Techniques

This section describes the techniques VMware Cloud on AWS uses to allow memory overcommitment.

VMware Cloud on AWS uses four memory management mechanisms to dynamically reduce the amount of machine physical memory required for each virtual machine. These are page sharing, ballooning, memory compression, and swapping.

Page Sharing

VMware Cloud on AWS can use a proprietary technique to transparently share memory pages between virtual machines, thus eliminating redundant copies of memory pages. While pages are shared by default within virtual machines, pages are not shared by default between virtual machines for security reasons.

Ballooning

If the host memory begins to get low and the virtual machine's memory usage approaches its memory target, VMware Cloud on AWS will use ballooning to reduce that virtual machine's memory demands. Using a VMware-supplied vmmemctl module installed in the guest operating system as part of the VMware Tools suite, VMware Cloud on AWS can cause the guest operating system to relinquish the memory pages it considers least valuable. Ballooning provides performance closely matching that of a native system under similar memory constraints. To use ballooning, the guest operating system must be configured with sufficient swap space.

Memory Compression

If the virtual machine's memory usage approaches the level at which host-level swapping will be required, VMware Cloud on AWS will use memory compression to reduce the number of memory pages it will need to swap out. Because the decompression latency is much smaller than the swap-in latency, compressing memory pages has significantly less impact on performance than swapping out those page

Host-Level Swapping

VMware Cloud on AWS will next forcibly reclaim memory from the virtual machine by swapping out pages to a swap file. Some of the swaped pages might be active, which can cause virtual machine performance to degrade significantly due to its high access latency. (Note that this swapping is distinct from the swapping that can occur within the virtual machine under the control of the guest operating system.)

For further information about memory management, see Understanding Memory Resource Management in VMware vSphere 5.0 (though this paper specifically addresses vSphere 5.0, most of the concepts are still applicable).

While VMware Cloud on AWS uses page sharing, ballooning, and memory compression to allow significant memory overcommitment, usually with little or no impact on performance, you should avoid overcommitting memory to the point that host-level swapping is used to swap out active memory pages.

If you suspect that memory overcommitment is beginning to affect the performance of a virtual machine you can take the following steps:

Note The point at which memory overcommitment begins to affect a workload's performance depends heavily on the workload. The areas we describe in this section will be relevant for most workloads. For some of those workloads, however, performance will be noticeably impacted earlier than for others.

1 In the vSphere Client, select the virtual machine in question, select **Monitor > Performance > Advanced**, in the drop down at the upper right select **Memory**, then look at the value of Ballooned memory (Average).

An absence of ballooning suggests that the host is not under heavy memory pressure and thus memory overcommitment is not affecting the performance of that virtual machine.

Note This indicator is only meaningful if the balloon driver is installed in the virtual machine and is not prevented from working.

Some ballooning is quite normal and not indicative of a problem.

2 In the vSphere Client, select the virtual machine in question, select **Monitor > Performance > Advanced**, in the drop down at the upper right select **Memory**, then compare the values of Consumed memory and Active memory. If consumed is higher than active, this suggests that the guest is currently getting all the memory it requires for best performance.

3 In the vSphere Client, select the virtual machine in question, select **Monitor > Utilization**, expand the Guest Memory pane, then look at the values of Swapped and Compressed. Swapping and compressing at the host level indicate more significant memory pressure.

4 Check for guest operating system swap activity within that virtual machine.

This can indicate that ballooning might be starting to impact performance, though swap activity can also be related to other issues entirely within the guest operating system (or can be an indication that the guest memory size is simply too small).

Memory Swapping Optimizations

This section describes ways to avoid or reduce host-level swapping, and presents techniques to minimize its impact on performance when it is unavoidable.

As described in Memory Overcommit Techniques, VMware Cloud on AWS supports a bounded amount of memory overcommitment without host-level swapping. If the overcommitment is so large that the other memory reclamation techniques are not sufficient, however, VMware Cloud on AWS uses host-level memory swapping, with a potentially significant impact on virtual machine performance.

- Because VMware Cloud on AWS uses page sharing, ballooning, and memory compression to reduce the need for host-level memory swapping, don’t deactivate these techniques.
- If you choose to overcommit memory with VMware Cloud on AWS, be sure you have sufficient swap space on your hosts. At the time a virtual machine is first powered on, VMware Cloud on AWS creates a swap file for that virtual machine equal in size to the difference between the virtual machine's configured memory size and its memory reservation. The available disk space must therefore be at least this large (plus the space required for VMX swap, as described in Memory Overhead).

  Note that in VMware Cloud, swap files inherit the storage policy associated with the VM, and are thus thin-provisioned by default. If left at the default value, this swap file will start out having slightly reduced performance, but will reach full performance once it is fully utilized.

- The swap file location for a specific virtual machine can be set in the vSphere Client (right-click the virtual machine, select **Edit Settings**, choose the **VM Options** tab, expand **Advanced**, and select a Swap file location). If this option is not set, the default behavior is that the swap file will be created in the virtual machine's working directory.
- Host-level memory swapping can, in many cases, be reduced for a specific virtual machine by reserving memory for that virtual machine at least equal in size to the machine's active working set. Host-level memory swapping can be eliminated for a specific virtual machine by reserving memory equal in size to the virtual machine's entire memory.

  Note When applied to a running virtual machine, the effect of these memory reservations might appear only gradually. To immediately see the full effect you would need to power-cycle the virtual machine.

  Be aware, however, that configuring resource reservations will reduce the number of virtual machines that can be run on a system. This is because VMware Cloud on AWS will keep available enough host memory to fulfill all reservations (both for virtual machines and for overhead) and won't power-on a virtual machine if doing so would reduce the available memory to less than the reserved amount.

  Note The memory reservation is a guaranteed lower bound on the amount of physical memory VMware Cloud on AWS reserves for the virtual machine. It can be configured through the vSphere Client in the settings window for each virtual machine (right-click the virtual machine, select **Edit Settings**, choose the **Virtual Hardware** tab, expand **Memory**, then set the desired reservation).

Memory Overhead

Virtualization causes an increase in the amount of physical memory required due to the extra memory needed by VMware Cloud on AWS for its own code and for data structures.

This additional memory requirement can be separated into two components:

1. A system-wide memory space overhead for the VMkernel and various host agents (hostd, vpxa, vSAN, etc.).
2. An additional memory space overhead for each virtual machine.

   The per-virtual-machine memory space overhead can be further divided into the following categories:

   - Memory reserved for the virtual machine executable (VMX) process.

     This is used for data structures needed to bootstrap and support the guest (i.e., thread stacks, text, and heap).
   - Memory reserved for the virtual machine monitor (VMM).

     This is used for data structures required by the virtual hardware (i.e., TLB, memory mappings, and CPU state).

- Memory reserved for various virtual devices (i.e., mouse, keyboard, SVGA, USB, etc.).
- Memory reserved for other subsystems, such as the kernel, management agents, etc.

The amounts of memory reserved for these purposes depend on a variety of factors, including the number of vCPUs, the configured memory for the guest operating system, whether the guest operating system is 32-bit or 64-bit, the VMM execution mode, and which features are activated for the virtual machine. For more information about these overheads, see vSphere Resource Management.

While the VMM and virtual device memory needs are fully reserved at the time the virtual machine is powered on, a feature called VMX swap can significantly reduce the per-virtual-machine VMX memory reservation, allowing more memory to be swapped out when host memory is overcommitted. This represents a significant reduction in the overhead memory reserved for each virtual machine.

The creation of a VMX swap file for each virtual machine (and thus the reduction in host memory reservation for that virtual machine) is automatic. By default, this file is created in the virtual machine's working directory, but a different location can be set with sched.swap.vmxSwapDir.

The amount of disk space required varies, but even for a large virtual machine is typically less than 100MB (typically less than 300MB if the virtual machine is configured for 3D graphics). The system retains a small working set memory reservation, and does not swap under ordinary usage. Extraordinary usage (for example, taking screenshots every second, or collecting a support dump) might cause swapping.

Note The VMX swap file is entirely unrelated to swap to host-level swapping, which is described in Memory Overcommit Techniques.

In addition, VMware Cloud on AWS also provides optimizations, such as page sharing (see Memory Overcommit Techniques), to reduce the amount of physical memory used on the underlying server. In some cases these optimizations can save more memory than is taken up by the overhead described in this section.

2MB Large Memory Pages for Hypervisor and Guest Operating System

This section describes how VMware Cloud on AWS handles large memory pages.

In addition to the usual 4KB memory pages, VMware Cloud on AWS also provides 2MB memory pages (commonly referred to as "large pages"). (Note that although some CPUs support 1GB memory pages, VMware Cloud on AWS supports only 2MB large pages.)

VMware Cloud on AWS assigns these 2MB machine memory pages to guest operating systems whenever possible; it does this even if the guest operating system doesn't request them (though the full benefit of large pages comes only when the guest operating system and applications use them as well, as described in Guest Operating System Memory Considerations). The use of large pages can significantly reduce TLB misses, improving the performance of most workloads, especially those with large active memory working sets. In addition, large pages can slightly reduce the per-virtual-machine memory space overhead.

Use of large pages can also change page sharing behavior. While VMware Cloud on AWS ordinarily uses page sharing regardless of memory demands, it does not share large pages. Therefore with large pages, page sharing might not occur until memory overcommitment is high enough to require the large pages to be broken into small pages. For further information see VMware KB article 78544. In VMware Cloud on AWS, disabling large pages can be done only on a per VM basis.

VMware Cloud on AWS Storage Considerations

This section provides guidance regarding storage considerations in VMware Cloud on AWS.

Virtual Disk Modes

VMware Cloud on AWS supports three virtual disk modes: Independent persistent, Independent nonpersistent, and Dependent.

Note An independent disk does not participate in virtual machine snapshots. That is, the disk state will be independent of the snapshot state; creating, consolidating, or reverting to snapshots will have no effect on the disk.

These modes have the following characteristics:

Independent persistent

In this mode changes are persistently written to the disk, providing the best performance.

Independent nonpersistent

In this mode disk writes are appended to a redo log. The redo log is erased when you power off the virtual machine or revert to a snapshot, causing any changes made to the disk to be discarded. When a virtual machine reads from an independent nonpersistent mode disk, VMware Cloud on AWS first checks the redo log (by looking at a directory of disk blocks contained in the redo log) and, if the relevant blocks are listed, reads that information. Otherwise, the read goes to the base disk for the virtual machine. Because of these redo logs, which track the changes in a virtual machine's file system and allow you to commit changes or revert to a prior point in time, performance might not be as high as independent persistent mode disks.

Dependent

In this mode, if a snapshot has been taken, disk writes are appended to a redo log that persists between power cycles. Thus, like the independent nonpersistent mode disks described above, dependent mode disk performance might not be as high as independent persistent mode disks. If a snapshot has not been taken, however, dependent disks are just as fast as independent disks.

Storage I/O Resource Allocation

The maximum storage I/O resources available to an object, such as a VMDK, can be set using Limits. These limits, set in I/O operations per second (IOPS), can be used to provide strict isolation and control of certain workloads. By default, these are set to unlimited; when set to any other value, VMware Cloud on AWS enforces the limits even if the underlying datastores are not fully utilized.

Limits are a cap on the object's resource usage, not a guarantee that the object will receive that amount. Limiting some workloads can allow other, more critical workloads to maintain their performance even during peak load periods when there is contention for I/O resources.

To set Limits, from the vSphere Client right-click a virtual machine, select **Edit Settings**, choose the **Virtual Hardware** tab, expand **Hard disk 1** (or whichever hard disk you want to configure), then set the Limit - IOPs to the desired value.

Running Storage Latency Sensitive Applications

By default, the VMware Cloud on AWS storage stack is configured to drive high storage throughput at low CPU cost. While this default configuration provides better scalability and higher consolidation ratios, it comes at the cost of potentially higher storage latency. Workloads that are highly sensitive to storage latency might therefore benefit from the following:

- If you are using an LSILogic vHBA or a PVSCSI vHBA (with hardware version 11 or later) in the virtual machine, you can adjust the reqCallThreshold value.

  The lower the reqCallThreshold value, the less time the I/O requests are likely to stay in the vHBA's queue. For instance, if reqCallThreshold is set to 1, it means when there is even one I/O request in the vHBA's queue, the I/Os will be dispatched to the lower layer. (The default reqCallThreshold value is 8.)

  Reducing the reqCallThreshold value will typically increase CPU usage. It also, perhaps counter-intuitively, has the potential to reduce the maximum possible throughput on the virtual disk(s).

  The system-wide configuration can be overridden for a specific vHBA. To change this variable through the vSphere Client:

  a Right-click the virtual machine with the vHBA you wish to change, then select **Edit Settings**.

  b Under the **VM Options** tab, expand **Advanced**, then click **EDIT CONFIGURATION**.

c Look for scsiX.reqCallThreshold (where X is the target SCSI number) and set it to the desired reqCallThreshold value. If the variable isn't present, click **ADD CONFIGURATION PARAMS** and enter it as a new parameter.

d Click **OK** to close the **Configuration Parameters** window.

e Click **OK** to close the **Edit Settings** window.

The change will not take effect until the virtual machine has been restarted or the relevant vHBA is disconnected from the virtual machine and connected again.

For further information on running storage latency sensitive workloads, the white paper Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs, while addressing primarily network latency-sensitive workloads, might prove helpful.

VMware Cloud on AWS Networking Considerations

This section provides guidance regarding networking considerations in VMware Cloud on AWS.

SplitRx Mode

SplitRx mode uses multiple physical CPUs to process network packets received in a single network queue. This feature can significantly improve network performance for certain workloads. These workloads include:

- Multiple virtual machines on one VMware Cloud on AWS host all receiving multicast traffic from the same source. (SplitRx mode will typically improve throughput and CPU efficiency for these workloads.)
- Traffic via the vSphere Network Appliance API (DvFilter) between two virtual machines on the same host. (SplitRx mode will typically improve throughput and maximum packet rates for these workloads.)

This feature is automatically activated for a VMXNET3 virtual network adapter (the only adapter type on which it is supported) when VMware Cloud on AWS detects that a single network queue on a physical NIC is both (a) heavily utilized and (b) getting more than 10,000 broadcast/multicast packets per second.

Note SplitRx mode will be automatically activated as described above only if the network traffic is coming in over a physical NIC, not when the traffic is entirely internal to the VMware Cloud on AWS host. In cases where the traffic is entirely internal, however, SplitRx mode can be manually activated if desired, as described in "Activating or Deactivating SplitRx Mode for an Individual Virtual NIC," below.

Activating or Deactivating SplitRx Mode for an Individual Virtual NIC

The SplitRx mode feature can be individually configured for each virtual NIC (overriding the global NetSplitRxMode setting) using the ethernetX.emuRxMode variable for each virtual machine (where X is replaced with the network adapter's ID).

The possible values for this variable are:

- `ethernetX.emuRxMode = "0"`

  This value deactivates splitRx mode for ethernetX.

- `ethernetX.emuRxMode = "1"`

  This value activates splitRx mode for ethernetX.

To change this variable through the vSphere Client:

1. Right-click the virtual machine you wish to change, then select **Edit Settings**.
2. Under the **VM Options** tab, expand **Advanced**, then click **EDIT CONFIGURATION**.
3. Look for ethernetX.emuRxMode (where X is the number of the desired NIC) and configure it as you wish. If the variable isn't present, click **ADD CONFIGURATION PARAMS** and enter it as a new parameter.
4. Click **OK** to close the **Configuration Parameters** window.
5. Click **OK** to close the **Edit Settings** window.

The change will not take effect until the virtual machine has been restarted or the relevant vNIC is disconnected from the virtual machine and connected again.

Virtual Network Interrupt Coalescing

Virtual network interrupt coalescing can reduce the number of interrupts, thus potentially decreasing CPU utilization. Though this could increase network latency, many workloads are not impacted by additional network latency of anywhere from a few hundred microseconds to a few milliseconds, and the reduction in virtual networking overhead can potentially allow more virtual machines to be run on a single VMware Cloud on AWS host.

By default, this feature is activated for all virtual NICs in VMware Cloud on AWS. For VMXNET3 virtual NICs, however, this feature can be set to one of three schemes or deactivated by changing the ethernetX.coalescingScheme variable (where X is the number of the virtual NIC to configure).

- The feature will be activated (the default) if the ethernetX.coalescingScheme variable is not present or if the variable is present and set to rbc (which stands for rate-based coalescing).

  In this case, the ethernetX.coalescingParams variable can be used to set the virtual network interrupt rate in interrupts per second. The value can range from 100 to 100,000, with a default of 4,000.

- The feature can be set to queue a predefined number of packets before interrupting the virtual machine or transmitting the packets by setting ethernetX.coalescingScheme to static.

In this case, the ethernetX.coalescingParams variable can be used to set the number of packets VMware Cloud on AWS will queue. The value can range from 1 to 64, with a default of 64. A larger queue size can reduce the number of context switches between the virtual machine and the VMkernel, potentially reducing CPU utilization both in the virtual machine and in the VMkernel.

Regardless of the number of packets queued, however, VMware Cloud on AWS waits no longer than 4 milliseconds before sending an interrupt or transmitting the packets. Other events, such as the virtual machine being idle, can also trigger virtual machine interrupts or packet transmission, so packets are rarely delayed the full 4 milliseconds.

- The feature can be set to be adaptive by setting ethernetX.coalescingScheme to adapt. This setting bases the interrupt rate on both the virtual machine load and the overall system load; it uses a lower interrupt rate when the load is high and a higher interrupt rate when the load is low.
- The feature can be deactivated by setting ethernetX.coalescingScheme to disabled. Deactivating this feature will typically result in more interrupts (and thus higher CPU utilization), but will typically also lead to lower network latency.

To configure VMXNET3 virtual interrupt coalescing through the vSphere Client:

1. Right-click the virtual machine you wish to change, then select **Edit Settings**.
2. Under the **VM Options** tab, expand Advanced, then click **EDIT CONFIGURATION**.
3. Look for and set the variable you wish to change:
   - If selecting a virtual interrupt coalescing scheme, look for ethernetX.coalescingScheme (where X is the number of the virtual NIC to configure) and set it to your desired value.

     If the variable isn't present, click **ADD CONFIGURATION PARAMS** and enter it as a new parameter.
   - If configuring the virtual network interrupt rate or queue size, look for ethernetX.coalescingParams (where X is the number of the virtual NIC to configure) and set it to your desired value.

     If the variable isn't present, click **ADD CONFIGURATION PARAMS** and enter it as a new parameter.
4. Click **OK** to close the **Configuration Parameters** window.
5. Click **OK** to close the **Edit Settings** window.

The change will not take effect until the virtual machine has been restarted.

Guest Operating Systems

2

This chapter provides guidance regarding the guest operating systems running in virtual machines.

This chapter includes the following topics:

Guest Operating System General Considerations

This section provides guidance regarding a number of general performance considerations in the guest operating system.

Guest Operating System Miscellaneous Considerations

- Use guest operating systems that are supported by VMware Cloud on AWS. Refer to the VMware Compatibility Guide for a list (go to http://www.vmware.com/resources/compatibility then, in **What are you looking for**: choose **Guest OS**).

  Note VMware Tools might not be available for unsupported guest operating systems.

- Install the latest version of VMware Tools in the guest operating system. Make sure to update VMware Tools after each VMware Cloud on AWS upgrade.

  Installing VMware Tools in Windows guests updates the BusLogic SCSI driver included with the guest operating system to the VMware-supplied driver. The VMware driver has optimizations that guest-supplied Windows drivers do not.

  VMware Tools also includes the balloon driver used for memory reclamation. Ballooning (described in Memory Overcommit Techniques) will not work if VMware Tools is not installed.

- Deactivate screen savers and Window animations in virtual machines. On Linux, if using an X server is not required, deactivate it. Screen savers, animations, and X servers all consume extra physical CPU resources, potentially affecting consolidation ratios and the performance of other virtual machines.
- Schedule backups and virus scanning programs in virtual machines to run at off-peak hours. Avoid scheduling them to run simultaneously in multiple virtual machines on the same host. In general, it is a good idea to evenly distribute CPU usage not just across CPUs, but also across time. For workloads such as backups and virus scanning, where the load is predictable, this is easily achieved by scheduling the jobs appropriately.
- For the most accurate timekeeping, consider configuring your guest operating system to use NTP, Windows Time Service, the VMware Tools time-synchronization option, or another timekeeping utility suitable for your operating system.

  We recommend, however, that within any particular virtual machine you use either the VMware Tools time-synchronization option or another timekeeping utility, but not both.

  For additional information about best practices for timekeeping within virtual machines, see VMware KB articles 1318 and 1006427.

Measuring Performance in Virtual Machines

Be careful when measuring performance from within virtual machines.

- vSphere allows you to take advantage of virtualized CPU performance counters to use performance tuning tools inside the guest operating system. For more information, see Monitoring Guest Operating System Performance in *vSphere Monitoring and Performance* and Enable Virtual CPU Performance Counters in *vSphere Virtual Machine Administration*.
- Timing numbers measured from within virtual machines can be inaccurate, especially when the processor is overcommitted.

  One possible approach to this issue is to use a guest operating system that has good timekeeping behavior when run in a virtual machine, such as a guest that uses the NO_HZ kernel configuration option (sometimes called “tickless timer”). More information about this topic can be found in the white paper Timekeeping in VMware Virtual Machines.
- Measuring performance from within virtual machines can fail to take into account resources used by VMware Cloud on AWS for tasks it offloads from the guest operating system, as well as resources consumed by virtualization overhead.

Measuring resource utilization using tools at the host level (such as the vSphere Client, VMware Tools, or resxtop) can avoid these problems.

Note Although esxtop can’t be used in VMware Cloud on AWS, resxtop can. See the blog post ESXTOP and VMware Cloud on AWS for guidance. For information about using resxtop see Performance Monitoring Utilities: resxtop and esxtop in *vSphere Monitoring and Performance*.

Guest Operating System CPU Considerations

This section addresses CPU considerations in the guest operating system.

Side-Channel Vulnerability Mitigation in Guest Operating Systems

A class of security vulnerabilities collectively known as “side-channel vulnerabilities” have been discovered in many modern CPUs. These include vulnerabilities commonly called Spectre, Meltdown, Foreshadow, L1TF, and others. Mitigation for some side-channel vulnerabilities takes place in the guest operating system. These mitigations address serious security vulnerabilities, but they can also have a significant impact on performance, especially on systems that are CPU resource constrained.

Because of the complexity of this topic, as well as its ever-changing nature, it isn’t thoroughly addressed in this book. For the latest information about these vulnerabilities as they relate to VMware products, see the operating system-specific mitigations sections in VMware KB articles 52245, 54951, and 55636, all of which are updated regularly.

Virtual NUMA (vNUMA)

Virtual NUMA (vNUMA) exposes NUMA topology to the guest operating system, allowing NUMA-aware guest operating systems and applications to make the most efficient use of the underlying hardware’s NUMA architecture.

Note For more information about NUMA, see Non-Uniform Memory Access (NUMA).

Virtual NUMA, which requires virtual hardware version 8 or later, can in some cases provide significant performance benefits for wide virtual machines (as defined in Non-Uniform Memory Access (NUMA)), though the benefits depend heavily on the level of NUMA optimization in the guest operating system and applications.

- You can obtain the maximum performance benefits from vNUMA if your clusters are composed entirely of hosts with matching NUMA architecture. The hosts in a single VMware Cloud on AWS clusters always have matching NUMA architecture, but when VMs are moved to a different VMware Cloud on AWS cluster, or moved to an on-premises ESXi host, the NUMA architecture could be different.

  The benefit obtained from matching NUMA architecture is because the very first time a vNUMA-activated virtual machine is powered on, its vNUMA topology is set based in part on the NUMA topology of the underlying physical host on which it is running. By default, once a virtual machine’s vNUMA topology is initialized it doesn’t change unless the number of vCPUs in that virtual machine is changed. This means that if a vNUMA virtual machine is moved to a host with a different NUMA topology, the virtual machine’s vNUMA topology might no longer be optimal for the underlying physical NUMA topology, potentially resulting in reduced performance.

- When sizing your virtual machines, take into account the size of the physical NUMA nodes:
  - For the best performance, try to size your virtual machines to stay within a physical NUMA node. For example, if you have a host system with six cores per NUMA node, try to size your virtual machines with no more than six vCPUs.
  - When a virtual machine needs to be larger than a single physical NUMA node, try to size it such that it can be split evenly across as few physical NUMA nodes as possible.
  - Use caution when creating a virtual machine that has a vCPU count that exceeds the physical processor core count on a host. Because hyper-threads are considered logical threads, this is sometimes permissible, but will potentially create contention when used.
- Changing the corespersocket value doesn't influence vNUMA or the configuration of the vNUMA topology. The configuration of vSockets and corespersocket now only affects the presentation of the virtual processors to the guest OS, something potentially relevant for software licensing. vNUMA will automatically determine the proper vNUMA topology to present to the guest OS based on the underlying host.

  This decoupling of the corespersocket setting from vNUMA allows vSphere to automatically determine the best vNUMA topology.

  To disable this behavior and directly control the vNUMA topology, see the numa.vcpu.followcorespersocket setting in the Virtual NUMA Controls section of *vSphere Resource Management*.

  Note Although corespersocket no longer directly sets the vNUMA topology, some corespersocket values could result in sub-optimal guest OS topologies; that is, topologies that are not efficiently mapped to the physical NUMA nodes, potentially resulting in reduced performance.

  For more information about this, see VMware KB article 81383 and the Virtual Machine vCPU and vNUMA Rightsizing – Guidelines blog post. In addition, the Virtual Machine Compute Optimizer tool can provide guidance on configuring optimal topologies.

- By default, vNUMA is activated only for virtual machines with more than eight vCPUs. This feature can be activated for smaller virtual machines, however, while still allowing VMware Cloud on AWS to automatically manage the vNUMA topology. This can be useful for wide virtual machines (that is, virtual machines with more vCPUs than the number of cores in each physical NUMA node) with eight or fewer vCPUs.

  To activate vNUMA for virtual machines with eight or fewer vCPUs, use the vSphere Client to set numa.vcpu.min to the minimum virtual machines size (in vCPUs) for which you want vNUMA activated as follows:

  a Right-click the virtual machine you wish to change, then select **Edit Settings...**.

  b Under the **VM Options** tab, expand **Advanced**, then click **EDIT CONFIGURATION...**.

  c Look for numa.vcpu.min and configure it as you wish. If the variable isn't present, click **ADD CONFIGURATION PARAMS** and enter it as a new parameter.

d Click **OK** to close the **Configuration Parameters** window.

e Click **OK** to close the **Edit Settings** window.

Alternatively, you can take full manual control of a virtual machine's vNUMA topology using the maxPerVirtualNode option. For more details, see the Virtual NUMA Controls section of *vSphere Resource Management*.

- CPU Hot Add is a feature that allows the addition of vCPUs to a running virtual machine. Activating this feature, however, deactivates vNUMA for that virtual machine, resulting in the guest OS seeing a single vNUMA node. Without vNUMA support, the guest OS has no knowledge of the CPU and memory virtual topology of the host. This in turn could result in the guest OS making sub-optimal scheduling decisions, leading to reduced performance for applications running in large virtual machines.

  For this reason, activate CPU Hot Add only if you expect to use it. Alternatively, plan to power down the virtual machine before adding vCPUs, or configure the virtual machine with the maximum number of vCPUs that might be needed by the workload. If choosing the latter option, note that unused vCPUs incur a small amount of unnecessary overhead. Unused vCPUs could also cause the guest OS to make poor scheduling decisions within the virtual machine, again with the potential for reduced performance.

  For additional information see VMware KB article 2040375.

Guest Operating System Memory Considerations

This section addresses memory considerations in the guest operating system.

As described in 2MB Large Memory Pages for Hypervisor and Guest Operating System, VMware Cloud on AWS can make large memory pages available to the guest operating system.

If an operating system or application can benefit from large pages on a native system, that operating system or application can potentially achieve a similar performance improvement on a virtual machine backed with 2MB machine memory pages. Consult the documentation for your operating system and application to determine how to configure each of them to use large memory pages.

Guest Operating System Storage Considerations

The virtual storage adapter presented to the guest operating system can influence storage performance, as can that device's driver, driver settings, and other factors within the guest. This section addresses those considerations.

- For most guest operating systems, the default virtual storage adapter in VMware Cloud on AWS is either LSI Logic Parallel or LSI Logic SAS, depending on the guest operating system and the virtual hardware version. However, VMware Cloud on AWS also includes a paravirtualized SCSI storage adapter, PVSCSI (also called VMware Paravirtual). The PVSCSI adapter offers a significant reduction in CPU utilization as well as potentially increased throughput compared to the default virtual storage adapters, and is thus the best choice for environments with very I/O-intensive guest applications.

  Note In order to use PVSCSI, your virtual machine must be using virtual hardware version 7 or later, as described under VMware Cloud on AWS Host General Considerations.

  Note PVSCSI adapters are supported for boot drives in only some operating systems. For additional information see VMware KB article 1010398.

- If you choose to use the BusLogic Parallel virtual SCSI adapter, and are using a Windows guest operating system, you should use the custom BusLogic driver included in the VMware Tools package.
- The Non-Volatile Memory Express (NVMe) virtual storage adapter (virtual NVMe, or vNVMe) allows recent guest operating systems that include a native NVMe driver to use that driver to access storage through VMware Cloud on AWS. Compared to virtual SATA devices, the vNVMe virtual storage adapter accesses local PCIe SSD devices with much lower CPU cost per I/O and significantly higher IOPS.
- The depth of the queue of outstanding commands in the guest operating system SCSI driver can significantly impact disk performance. A queue depth that is too small, for example, limits the disk bandwidth that can be pushed through the virtual machine. See the driver-specific documentation for more information on how to adjust these settings.
- In some cases large I/O requests issued by applications in a virtual machine can be split by the guest storage driver. Changing the guest operating system’s registry settings to issue larger block size I/O requests can eliminate this splitting, thus enhancing performance. For additional information see VMware KB article 9645697.
- Make sure the disk partitions within the guest are aligned. For further information you might want to refer to the literature from the operating system vendor regarding appropriate tools to use as well as recommendations from the array vendor.

- VMware Cloud on AWS uses drives with 4KB sector size (that is, 4KB native, or 4Kn), but presents storage to the guest with 512B sector size (512 native). You can obtain the best storage performance if your workload issues mostly 4K-aligned I/Os. For more information on this subject, see VMware KB article 2091600 or Device Sector Formats in the *vSphere Storage Guide*. See SQL Server Configuration for specific recommendations for SQL Server.

Guest Operating System Networking Considerations

This section describes the various types of virtual network adapters, how to select one, and how to obtain the best performance from it.

When a virtual machine is first created, VMware Cloud on AWS allows selection of the virtual network adapter (vNIC) it will use. The available options are based on the specified guest operating system and the virtual hardware version. Because some operating systems don't have built-in drivers for the best-performing virtual network adapters, the list of available vNICs won't always include those that might provide the best performance. In some of these cases, though, a driver can later be installed in the guest operating system (typically as part of VMware Tools) and the vNIC can then be changed to a different type.

Types of Virtual Network Adapters

The possible virtual network adapters include three emulated types, three paravirtualized types, and a hybrid adapter.

The emulated virtual network adapters are:

- The Vlance virtual network adapter, which emulates an AMD 79C970 PCnet32 NIC. Drivers for this NIC are found in most 32-bit operating systems.
- The E1000 virtual network adapter, which emulates an Intel 82545EM NIC. Drivers for this NIC are found in many recent operating systems.
- The E1000E virtual network adapter, which emulates an Intel 82574 NIC. Drivers for this NIC are found in a smaller set of recent operating systems.

The VMXNET family of paravirtualized network adapters provide better performance in most cases than the emulated adapters. These paravirtualized network adapters implement an idealized network interface that passes network traffic between the virtual machine and the physical network interface card with minimal overhead. The VMXNET virtual network adapters (especially VMXNET3) also offer performance features not found in the other virtual network adapters. Drivers for VMXNET-family adapters are available for many guest operating systems supported by VMware Cloud on AWS; for optimal performance these adapters should be used for any guest operating system that supports them.

The VMXNET family of paravirtualized virtual network adapters includes:

- The VMXNET virtual network adapter. (This adapter is not available in VMware Cloud on AWS; it is included here for context.)

- The VMXNET2 virtual network adapter (also called “Enhanced VMXNET”). This adapter is based on the VMXNET adapter, but adds a number of performance features.
- The VMXNET3 virtual network adapter (also called VMXNET Generation 3). This adapter has all the features of the VMXNET2 adapter, along with several new ones.

Lastly, there’s a hybrid virtual network adapter:

- The Flexible virtual network adapter, which starts out emulating a Vlance adapter, but can function as a VMXNET adapter with the right driver.

Note The network speeds reported by the guest network driver in the virtual machine do not reflect the actual speed of the underlying physical network interface card. For example, the Vlance guest driver in a virtual machine reports a speed of 10Mb/s because the AMD PCnet card that VMware Cloud on AWS is emulating is a 10Mb/s device. This is true even if the physical card on the server is 100Mb/s, 1Gb/s, or faster. However, VMware Cloud on AWS is not limited to 10Mb/s in this situation and transfers network packets as fast as the resources on the physical server machine allow.

For more information on virtual network adapters, see VMware KB article 1001805 and the Virtual Machine Network Configuration section of *Virtual Machine Administration*.

Selecting Virtual Network Adapters

This section describes how to select virtual network adapters for the best performance.

- For the best performance, use the VMXNET3 paravirtualized network adapter for operating systems in which it is supported. This requires that the virtual machine use virtual hardware version 7 or later and, in some cases, requires that VMware Tools be installed in the guest operating system.

  Note Because Microsoft Windows doesn’t include a VMXNET3 driver, you’ll need to either:

  - Create a custom Windows image that includes the VMXNET3 driver (this is the preferred option).
  - Use a different virtual network adapter (typically E1000E or E1000) during operating system installation then, after the OS is installed, install VMware Tools (which includes a VMXNET3 driver), then add a VMXNET3 virtual network adapter.

- For guest operating systems in which VMXNET3 is not supported, we recommend using the E1000E virtual network adapter.
- If E1000E is not an option, use the flexible device type. The “flexible NIC” automatically converts each Vlance network device to a VMXNET device (a process also called “NIC Morphing”) if the VMware Tools suite is installed in the guest operating system and the operating system supports VMXNET.

Virtual Network Adapter Features and Configuration

This section describes various virtual network adapter features and how to configure the adapters for the best performance.

- When networking two virtual machines on the same host, try to connect them to the same vSwitch. When connected this way, their network speeds are not limited by the wire speed of any physical network card. Instead, they transfer network packets as fast as the host resources allow.
- Jumbo frames are recommended as a way to increase network throughout and reduce CPU load; they do this by allowing data to be transmitted using larger, and therefore fewer, packets.

  Jumbo frames are supported on the E1000, E1000E, VMXNET2, and VMXNET3 devices. They are activated by default on the underlying network for all same-data-center traffic, Connected VPC traffic, and VMware Transit Connect™ traffic. For paths using Direct Connect, jumbo frames must be explicitly activated on the Direct Connect private virtual interface by following the steps in Specify the Direct Connect MTU. The maximum supported MTU sizes are listed on the VMware Configuration Maximums site (Select Product: **VMware Cloud on AWS**, select your version, expand **Networking and Security**, select **General Networking**, click **VIEW LIMITS**).
- TCP Segmentation Offload (TSO) in VMware Cloud on AWS is activated by default in the VMkernel but is supported in virtual machines only when they are using an E1000, E1000E, VMXNET2, or VMXNET3 device. TSO can improve performance even if the underlying hardware does not support TSO.
- Similarly, Large Receive Offload (LRO) in VMware Cloud on AWS is activated by default in the VMkernel, but is supported in virtual machines only when they are using the VMXNET2 or VMXNET3 device.

  Note LRO support varies by operating system. Many Linux variants support LRO. In Windows virtual machines LRO is supported when the following prerequisites are met:

  - The virtual machine uses virtual hardware version 11 or later.
  - The virtual machine uses the VMXNET3 device.
  - The virtual machine is running Windows Server 2012 or later or Windows 8.0 or later.
  - The guest operating system uses VMXNET3 vNIC driver version 1.6.6.0 or later.
  - Receive Segment Coalescing (RSC) is globally activated in the guest operating system.

  For more information about configuring LRO, see the Large Receive Offload section of *vSphere Networking*.

- In some cases, low receive throughput in a virtual machine can be due to insufficient receive buffers (also described as an overflowing receive ring) in the receiver network device. If the receive buffers in the guest operating system's network driver are insufficient, packets will be dropped in the VMkernel, degrading network throughput. A possible workaround is to increase the number of receive buffers, though this might increase the host physical CPU workload.

  For VMXNET, the default number of receive and transmit buffers is 100 each, with the maximum possible being 128. For VMXNET2, the default number of receive and transmit buffers are 150 and 256, respectively, with the maximum possible receive buffers being 512. See VMware KB article 1010071 for information about altering these defaults.

  For E1000, E1000E, and VMXNET3, the default number of receive and transmit buffers are controlled by the guest driver, with the maximum possible for both being 4096. In Linux, these values can be changed from within the guest by using ethtool. In Windows, the values can be changed from within the guest in the device properties window. For additional information see VMware KB article 1010071.

- Multiple receive and transmit queues (often referred to as receive-side scaling (RSS) or scalable I/O) allow network packets from a single NIC to be scheduled in parallel on multiple CPUs. Without multiple queues, network interrupts can be handled on only one CPU at a time. Multiple queues help throughput in cases where a single CPU would otherwise be saturated with network packet processing and become a bottleneck. To prevent out-of-order packet delivery, the driver schedules all of a flow's packets to the same CPU.

  The E1000E and VMXNET3 devices support multiple queues for many guest operating systems that natively support the feature, which include Windows Server 2003 SP2 and later, Windows 7 and later, and some Linux distributions.

  The VMXNET3 drivers included with the vSphere 5.0 and later versions of VMware Tools default to having multiple receive queues activated, as does the VMXNET3 driver included with Linux kernel 2.6.37 and later. For more information, see VMware KB article 2020567.

  When multiple receive queues are activated, the feature by default configures 1, 2, 4, or 8 receive queues in a virtual machine, choosing the largest of these values less than or equal to the number of vCPUs in that virtual machine.

  In both Windows and Linux the number the number of transmit queues, by default, is the same as the number of receive queues.

  In order to obtain the maximum performance with your specific workloads and resource availability you can try out different values for the number of receive queues (which must be set to a power of 2 and can be a maximum of 8 or the number of vCPUs, whichever is lower). This setting is changed on the advanced driver configuration tab within the guest operating system.

Virtual Infrastructure Management

3

This chapter provides guidance regarding infrastructure management best practices. Most of the suggestions included in this section can be implemented using a vSphere Client connected to a VMware vCenter Server.

Further detail about many of these topics, as well as background information, can be found in the white paper VMware vCenter Server Performance and Best Practices. (Though written for vSphere 6.0, most of the information in this paper is still relevant.)

This chapter includes the following topics:

- General Resource Management
- VMware vSphere Management
- VMware vSphere vMotion
- VMware vSphere Distributed Resource Scheduler (DRS)
- VMware vSphere High Availability
- VMware Virtual SAN (vSAN)

General Resource Management

VMware Cloud on AWS provides several mechanisms to configure and adjust the allocation of CPU and memory resources for virtual machines running within it. Resource management configurations can have a significant impact on virtual machine performance.

This section lists resource management practices and configurations recommended by VMware for optimal performance.

- Use resource settings (that is, Reservation, Shares, and Limits) only if needed in your environment.
- If you expect frequent changes to the total available resources, use Shares, not Reservation, to allocate resources fairly across virtual machines. If you use Shares and you subsequently upgrade the hardware, each virtual machine stays at the same relative priority (keeps the same number of shares) even though each share represents a larger amount of memory or CPU.

- Use Reservation to specify the minimum acceptable amount of CPU or memory, not the amount you would like to have available. After all resource reservations have been met, VMware Cloud on AWS allocates the remaining resources based on the number of shares and the resource limits configured for your virtual machine.

  As indicated above, reservations can be used to specify the minimum CPU and memory reserved for each virtual machine. In contrast to shares, the amount of concrete resources represented by a reservation does not change when you change the environment, for example by adding or removing virtual machines. Don't set Reservation too high. A reservation that's too high can limit the number of virtual machines you can power on in a resource pool, cluster, or host.

  When specifying the reservations for virtual machines, always leave some headroom for memory virtualization overhead (as described in VMware Cloud on AWS Memory Considerations) and migration overhead. Reservations that fully commit the capacity of the cluster or of individual hosts in the cluster can prevent DRS from migrating virtual machines between hosts. As you approach fully reserving all capacity in the system, it also becomes increasingly difficult to make changes to reservations and to the resource pool hierarchy without violating admission control.

- Use resource pools for delegated resource management. To fully isolate a resource pool, make the resource pool type Fixed and use Reservation and Limit.

- Group virtual machines for a multi-tier service into a resource pool. This allows resources to be assigned for the service as a whole.

VMware vSphere Management

VMware vSphere environments are typically managed with the VMware vSphere® Client or the VMware vSphere Web Services SDK.

Large numbers of connected vSphere Clients and vSphere Web Services SDK Clients can affect the performance of a vCenter Server. Exceeding the supported maximums, even if it seems to work, is even more likely to affect vCenter Server performance.

Note For supported maximums, see VMware Configuration Maximums, then select **VMware Cloud on AWS** for Product, and appropriate choices for Version and Categories.

VMware vSphere Clients

This section describes how to obtain the best performance from the VMware vSphere Client.

The vSphere Client can be used to control and configure vCenter Servers and virtual machines.

The vSphere Client back end, called the vSphere Client Server, is a Java application; in VMware Cloud on AWS this is managed by VMware. The vSphere Client front end, called simply the vSphere Client, is an HTML5-based client running in a web browser pointed to the Java-based Application Server.

- For the best vSphere Client performance, make sure the user's system has sufficient CPU resources (we recommend at least two CPU cores; the faster the better) and memory (as an example, for client systems running Windows we recommend at least 4GB of RAM).
- Large network latencies can significantly reduce vSphere Client performance. For the best performance, the network latencies between the vSphere Client running on the user's system and the vSphere Client back end should be under 30ms.

  Note When it's not possible to establish a low-latency connection, an alternate option would be to place a virtual or physical system near the vSphere Client back-end (such that the system has a low-latency connection to the back-end), and run the vSphere Client front-end on that system, remotely accessing it from the more distant location using a remote protocol such as RDP.

- Using a 64-bit browser to view the vSphere Client allows the allocation of more memory, potentially improving performance.
- When possible, use the search function instead of navigating to managed objects. The vSphere Client is designed for the use of inventory search to find managed objects (clusters, hosts, virtual machines, datastores, tags, and so on). Though you can access managed objects through the inventory tree or inventory list views, the inventory search function will typically provide better performance than navigating among these objects.

vSphere Web Services SDK Clients

The VMware vSphere Web Services SDK can be an efficient way to manage the vSphere environment.

To learn more about the VMware vSphere API and supported SDK libraries, refer to the vSphere API and SDK Documentation.

For examples of good programming practices, see code samples from the VMware Communities sample code page (http://communities.vmware.com/community/vmtn/developer/codecentral).

To use the SDK to obtain cluster performance data by aggregating information from individual hosts, see the vCenter Cluster Performance Tool fling (https://labs.vmware.com/flings/vcenter-cluster-performance-tool).

VMware vSphere vMotion

This section provides performance best practices for VMware vSphere®vMotion®.

vSphere vMotion Recommendations

Consider the recommendations in this section for the best vSphere vMotion performance. For more detail about vSphere vMotion architecture and performance, see the white paper VMware vSphere vMotion Architecture, Performance and Best Practices in VMware vSphere 5. (Though written about a previous version, much of the material still applies.)

- VMware Cloud on AWS supports multiple virtual hardware versions. If you plan to move a virtual machine from VMware Cloud on AWS to an on-premises ESXi host (in a hybrid cloud scenario, for example), make sure the machine's virtual hardware version is supported by the ESXi hosts on which you intend to use them. You can see what's supported by each version of ESXi in VMware KB article 2007240.
- VMware Cloud on AWS supports encrypted vSphere vMotion. This feature encrypts vSphere vMotion traffic when both the source and destination hosts are capable of supporting encrypted vSphere vMotion (that is, when both hosts are in VMware Cloud on AWS or are running ESXi 6.5 or later). If a virtual machine running on a host that supports encrypted vSphere vMotion is migrated to a host that does not support it, vSphere vMotion traffic will not be encrypted.

  Encrypted virtual machines are always moved with encrypted vSphere vMotion. For virtual machines that are not encrypted, vSphere vMotion encryption can be changed from **Opportunistic** (the default) to **Disabled** or **Required** (right-click the virtual machine, select **Edit Settings**, select **VM Options**, click **Encryption**, then select an option from the **Encrypted vMotion** drop-down menu).

  For more detail about encrypted vSphere vMotion architecture and performance, see the white paper VMware vSphere Encrypted vMotion Architecture, Performance, and Best Practice. (Though written about a previous version, much of the material still applies.)
- While a vSphere vMotion operation is in progress, VMware Cloud on AWS opportunistically reserves CPU resources on both the source and destination hosts in order to ensure the ability to fully utilize the network bandwidth. VMware Cloud on AWS will attempt to use the full available network bandwidth regardless of the number of vSphere vMotion operations being performed. VMware Cloud on AWS will reserve one core worth of CPU for each 10Gb/s of reported network bandwidth. On i3.metal instances, that amounts to 5.75GHz. Therefore leaving some unreserved CPU capacity in a cluster can help ensure that vSphere vMotion tasks get the resources required in order to fully utilize available network bandwidth.

VMware vSphere Distributed Resource Scheduler (DRS)

This section lists VMware vSphere® Distributed Resource Scheduler™ (DRS) practices and configurations recommended by VMware for optimal performance.

DRS in General

- If appropriate for your workloads, consider selecting one of the higher-performing elastic DRS policies, **Optimize for Best Performance** or **Optimize for Rapid Scale-Out**, as described in About Elastic DRS.
- DRS actions are governed by compute policies, as described in Using Policies and Profiles. Note, however, that compute policies can be violated to allow maintenance operations.

DRS Cluster Sizing and Resource Settings

This section describes the DRS cluster sizing and resource settings.

- Carefully select the resource settings (that is, reservations, shares, and limits) for your virtual machines.
  - Setting reservations too high can leave few unreserved resources in the cluster, thus limiting the options DRS has to recommend vSphere vMotion migrations.
  - Setting limits too low could keep virtual machines from using extra resources available in the cluster to improve their performance.

  Use reservations to guarantee the minimum requirement a virtual machine needs, rather than what you might like it to get. Note that shares take effect only when there is resource contention. Note also that additional resources reserved for virtual machine memory overhead need to be accounted for when sizing resources in the cluster.

  If the overall cluster capacity might not meet the needs of all virtual machines during peak hours, you can assign relatively higher shares to virtual machines or resource pools hosting mission-critical applications to reduce the performance interference from less-critical virtual machines.
- If you will be using vSphere vMotion, it’s a good practice to leave some unused (and unreserved) CPU capacity in your cluster. As described in vSphere vMotion Recommendations, when a vSphere vMotion operation is started, VMware Cloud on AWS reserves some CPU resources for that operation.

VMware vSphere High Availability

VMware vSphere® High Availability (HA) minimizes virtual machine downtime by monitoring hosts, datastores, virtual machines, or applications within virtual machines, then, in the event a failure is detected, restarting virtual machines on alternate hosts or resetting them on the same host.

- All active hosts in a cluster (those not in standby mode, maintenance mode, or disconnected) participate in an election to choose the primary host for the cluster; all other hosts become secondary hosts. The primary has a number of responsibilities, including monitoring the state of the hosts in the cluster, protecting the powered-on virtual machines, initiating failover, and reporting cluster health state to vCenter Server. The primary is elected using an algorithm that takes into account various properties of the hosts, such as the number of datastores to which they are connected. Serving in the role of primary will have little or no effect on a host’s performance.

- When the primary host can’t communicate with a secondary host over the management network, the primary uses datastore heartbeating to determine the state of that secondary.
- In the event of a failure, vSphere HA queries DRS running on vCenter for its recommendation about which hosts the virtual machines should be restarted on. If DRS doesn't supply a recommendation, HA falls back to a simpler algorithm.

For further details about vSphere HA, see vSphere Availability.

VMware Virtual SAN (vSAN)

VMware Cloud on AWS uses VMware vSAN storage, allowing storage resources attached directly to hosts to be used for distributed storage and accessed by multiple other hosts. Follow the recommendations in this section for the best performance.

For additional information about vSAN performance, see the white paper vSAN 6.6 Performance Improvements (though written for vSAN 6.6, much of the content is still relevant). Note also that VMware Cloud on AWS uses all-flash vSAN storage, so recommendations about hybrid vSAN wouldn’t apply.

- For the best performance virtual machines should be distributed relatively evenly across vSAN hosts. VMware DRS can help to achieve this (see VMware vSphere Distributed Resource Scheduler (DRS)).
- RAID-5 and RAID-6 (also known collectively as erasure coding) can provide data protection, but with a performance penalty. RAID-1 (mirroring) offers the best performance, but at the cost of significantly lower space efficiency.

  Erasure coded storage performs better on i3en.metal than on i3.metal. This is because while the former performs compression, the latter performs both compression and deduplication.

  For further information about vSAN performance with erasure coding, see the white paper vSAN 6.6 Performance Improvements (though written for vSAN 6.6, much of the content is still relevant).

For more details about vSAN configuration options and other vSAN performance tips, see the white paper VMware vSAN Network Design.