Deploying Extremely Latency-Sensitive Applications in VMware vSphere 5.5

Performance Study

TECHNICAL WHITEPAPER

Table of Contents

Introduction

Performance demands of latency-sensitive applications have long been thought to be incompatible with virtualization. Such applications as distributed in-memory data management, stock trading, and high-performance computing (HPC) demand very low latency or jitter, typically of the order of up to tens of microseconds. Although virtualization brings the benefits of simplifying IT management and saving costs, the benefits come with an inherent overhead due to abstracting physical hardware and resources and sharing them.

Virtualization overhead may incur increased processing time and its variability. VMware vSphere® ensures that this overhead induced by virtualization is minimized so that it is not noticeable for a wide range of applications including most business critical applications such as database systems, Web applications, and messaging systems. vSphere also supports well applications with millisecond-level latency constraints such as VoIP streaming applications [6]. However, certain applications that are extremely latency-sensitive would still be affected by the overhead due to strict latency requirements.

In order to support virtual machines with strict latency requirements, vSphere 5.5 introduces a new per-VM feature called Latency Sensitivity. Among other optimizations, this feature allows virtual machines to exclusively own physical cores, thus avoiding overhead related to CPU scheduling and contention. Combined with a pass-through functionality, which bypasses the network virtualization layer, applications can achieve near-native performance in both response time and jitter.

Briefly, this paper presents the following:

- It explains major sources of latency increase due to virtualization, which are divided into two categories: 1) contention created by sharing resources and 2) overhead due to the extra layers of processing for virtualization.
- It presents details of the latency-sensitivity feature that improves performance in terms of both response time and jitter by eliminating the major sources of extra latency added by using virtualization.
- It presents evaluation results demonstrating that the latency-sensitivity feature combined with pass-through mechanisms considerably reduces both median response time and jitter compared to the default configuration, achieving near-native performance.
- It discusses the side effects of using the latency-sensitivity feature and presents best practices.

Latency Issues in a Virtualized Environment

Latency-sensitive applications typically demand a strict requirement for both response time of a transaction and its variability. The worst-case timing behavior is equally important to the average timing behavior. Transactions finishing too late (say, because packets are delivered too late) may even be considered as failures in certain types of applications. In the virtualized environment, however, meeting those requirements are more challenging due to 1) resource contention due to sharing and 2) the additional layer of scheduling virtual machines (VMs) and processing network packets for enabling resource sharing.

Resource Contention due to Sharing

One of the benefits that virtualization brings is reduced costs due to resource sharing (by consolidating many VMs in a single physical host). However, under heavy resource sharing, latency easily increases as the wait time for the resources (that is, ready time) increases. For example, with a large number of VMs running on the same host, the average response time of individual transactions of the workload would increase fast as they contend each other for resources such as CPU, memory, and network bandwidth.

Properly enforcing resource entitlement for VMs (hence controlling resources allocated to VMs) handles well those resource contention issues for the majority of applications. vSphere's fair share CPU scheduling allocates more CPU time to VMs with more CPU entitlement so that transactions can be quickly processed under CPU contention [2]. vSphere provides similar control knobs for enforcing memory entitlement [3]. Network I/O Control (NetIOC) partitions the physical network bandwidth based on its entitlement [8], which helps with bounding the (average) network packet processing latency. Enforcing resource entitlement in vSphere, however, is done at a relatively coarser time scale (for example, milliseconds). This may not be sufficient to support extremely latency-sensitive applications; it is possible for individual transactions or packets to have missed their latency requirements.

Overhead Due to Virtualization Layers

In a virtualized environment, the guest cannot directly access the physical resources in the host; there are extra virtual layers to traverse and the guest sees only virtualized resources (unless the VM is specifically configured via a pass-through mechanism to be given direct access to a physical resource). This is essential to allow the benefit of sharing resources while giving the illusion that virtual machines own the hardware. This also implies that there is an overhead associated with those virtualization layers.

Network I/O Virtualization

Since network devices are typically shared by multiple VMs, a VM cannot be allowed direct access to a physical network interface card (PNIC). Instead, a VM can be configured with virtual NICs (VNIC) that are supported by the virtual machine hardware [1]. In addition, in order to properly connect VMs to the physical networks or connecting VMs to each other, a switching layer is necessary. The cost of NIC virtualization and virtual switching is directly added to the response time and affects its variance due to extra processing.

In vSphere, the *virtual machine monitor* (VMM) manages the virtual machine hardware for the guest, while *VMkernel* manages physical hardware resources. (Throughput this paper, the virtualization layer combining both VMM and VMkernel is collectively referred to as *hypervisor.*) VMkernel utilizes dedicated threads to process network I/O and perform network virtualization. The communication between VMM and VMkernel (threads) involving network I/O are batched in an attempt to reduce the inefficiencies due to frequent communication. This technique, referred to as *VNIC coalescing* in this paper, can add an extra delay and variance to the response time.

In addition to the VNIC coalescing, vSphere supports Large Receive Offload (LRO) for VNICs to further reduce CPU cost by aggregating multiple incoming TCP packets into a large single packet. The side effect is similar to that of VNIC coalescing; it can impact response time and jitter in processing packets.

CPU Virtualization

Similar to other resources such as NICs, physical CPUs (PCPUs) also need to be virtualized. VMs do not exclusively own PCPUs, but they are instead configured with virtual CPUs (VCPU) supported by the virtual machine hardware. vSphere's proportional-share-based CPU scheduler enforces a necessary CPU time to VCPUs based on their entitlement [2]. This is an extra step in addition to the guest OS scheduling work. Any scheduling latency in the VMkernel's CPU scheduler and the context-switching cost from the VMM to the VMkernel to run the scheduler therefore results in an additional delay.

The CPU scheduler overhead is more visible when VCPUs frequently halt and wake up. Such a behavior is fairly commonly seen in latency-sensitive workloads where the VCPU becomes idle after a packet is sent out because it has to wait for a subsequent request from the client. Whenever the VCPU becomes idle and enters a halted state, it is removed from the PCPU by the scheduler. Once a new packet comes in, the CPU scheduler runs to make the VCPU ready again. This process of descheduling and rescheduling of a VCPU performed at each transaction incurs a non-negligible and variable latency that leads to increased response time and jitter.

Another consequence of CPU virtualization is the sharing of the last level cache (LLC) between VMs. The impact of sharing LLC depends on workloads that VMs run; a memory-intensive VM can negatively affect the performance of other VMs on the same socket by polluting the LLC.

Other Sources of Latency Issues

Power management in both the BIOS and vSphere can negatively affect latency. It is therefore recommended to turn off all power management schemes to achieve the lowest latency. For details on configuring power management settings, refer to the *Host Power Management in VMware vSphere 5* [14].

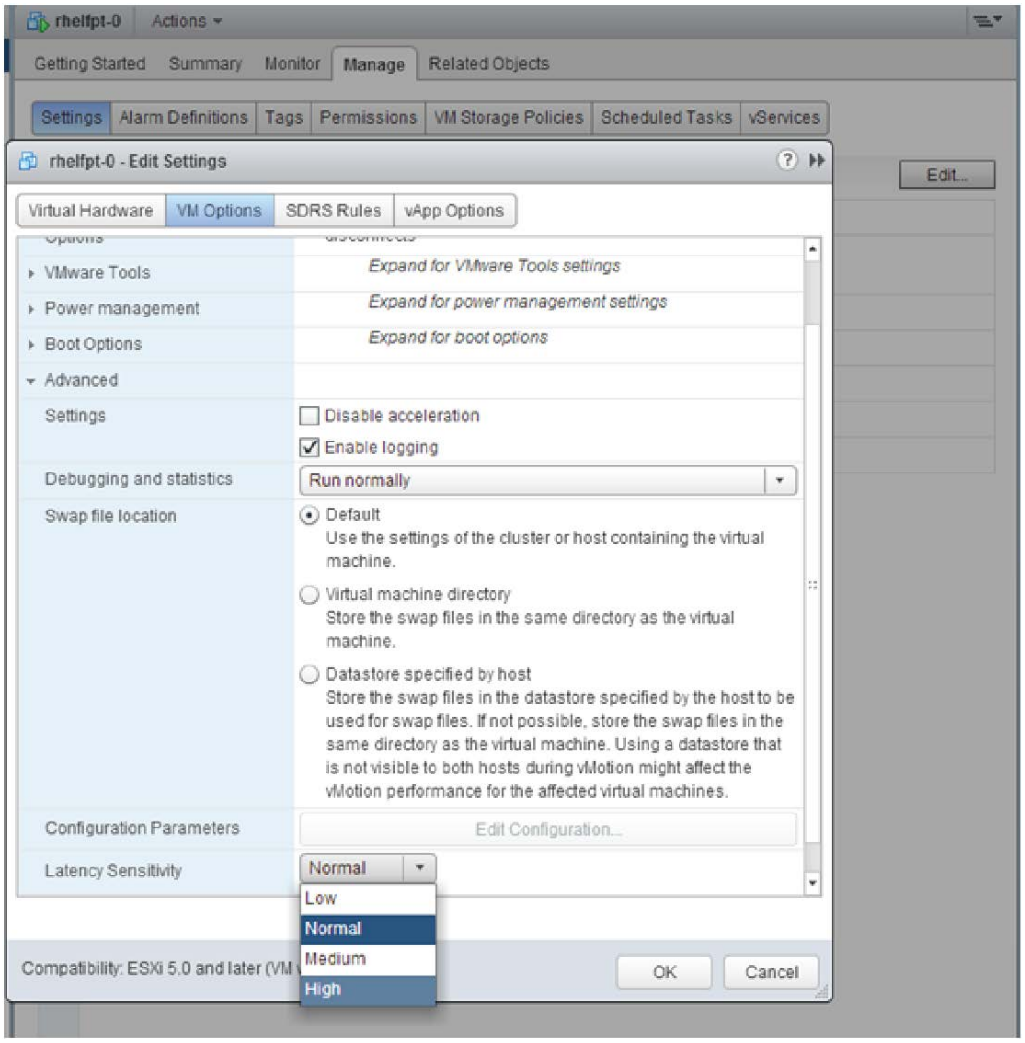
Latency can be also impacted by other shared resources such as memory and storage. Since the Latency-Sensitivity feature addresses only CPU and network related issues on the host where the VM is running, we will not discuss latency overhead from other shared resources.

Latency-Sensitivity Feature

The Latency-Sensitivity feature aims at eliminating the aforementioned major sources of extra latency imposed by virtualization to achieve low response time and jitter. Briefly, this per-VM feature achieves the goal by 1) giving exclusive access to physical resources to avoid resource contention due to sharing, 2) bypassing virtualization layers to eliminate the overhead of extra processing, and 3) tuning virtualization layers to reduce the overhead. Performance can be further improved when the latency-sensitivity feature is used together with a pass-through mechanism such as single-root I/O virtualization (SR-IOV).

Enabling the Latency-Sensitivity Feature

To enable the latency-sensitivity feature, the virtual machine's **Latency Sensitivity** should be set to be **High**. The **High** setting is designed for extremely latency-sensitive applications and all the functionalities and tunings that this feature provides are applied. The **Normal** setting does not engage the latency-sensitivity feature. The **Medium** and **Low** settings are experimental for vSphere 5.5 and hence are not covered in this white paper. Throughout this paper, when the latency-sensitivity feature is mentioned, it refers to the **High** setting unless specified otherwise.



VMware Tools — *Expand for VMware Tools settings*
Power management — *Expand for power management settings*
Boot Options — *Expand for boot options*
Advanced
Settings — ☐ Disable acceleration; ☑ Enable logging
Debugging and statistics — Run normally
Swap file location — ● Default: Use the settings of the cluster or host containing the virtual machine. ○ Virtual machine directory: Store the swap files in the same directory as the virtual machine. ○ Datastore specified by host: Store the swap files in the datastore specified by the host to be used for swap files. If not possible, store the swap files in the same directory as the virtual machine. Using a datastore that is not visible to both hosts during vMotion might affect the vMotion performance for the affected virtual machines.
Configuration Parameters — Edit Configuration...
Latency Sensitivity — Normal (Low / Normal / Medium / High)
Compatibility: ESXi 5.0 and later (VM

Figure 1. Enabling the latency-sensitivity feature for a given VM.

The latency-sensitivity feature is applied per VM, and thus a vSphere host can run a mix of normal VMs and VMs with this feature enabled. To enable the latency sensitivity for a given VM from the UI, access the **Advanced Settings** from the **VM Options** tab in the VM's **Edit Settings** pop-up window and select **High** for the **Latency Sensitivity** option as shown in Figure 1.

Enabling the latency-sensitivity feature requires memory reservation. Also, we recommend over-provisioning CPU to reduce contention; the number of VCPUs in the host should be less than the number of PCPUs to leave one or more PCPUs for VMkernel threads for I/O processing and system management. Since CPU contention from other VMs can change over time (for example, due to configuration changes), we strongly recommended maximizing CPU reservations of latency-sensitive VMs for best performance. Details of the feature are explained next.

What the Latency-Sensitivity Feature Does

Gives Exclusive Access to Physical Resources

- With the latency-sensitivity feature enabled, the CPU scheduler determines whether exclusive access to PCPUs can be given or not considering various factors including whether PCPUs are over-committed or not. Reserving 100% of VCPU time guarantees exclusive PCPU access to the VM. With exclusive PCPU access given, each VCPU entirely owns a specific PCPU; no other VCPUs and threads (including VMkernel I/O threads) are allowed to run on it. This achieves nearly zero ready time and no interruption from other VMkernel threads, improving response time and jitter under CPU contention. Note that even with exclusive access to PCPUs, the LLC is still shared with other VMs and threads residing on the same socket.
- The latency-sensitivity feature requires the user to reserve the VM's memory to ensure that the memory size requested by the VM is always available. Without memory reservation, vSphere may reclaim memory from the VM, when the host free memory gets scarce. Some memory reclamation techniques such as *ballooning* and *hypervisor swapping* may significantly downgrade VM performance, when the VM accesses the memory region that has been swapped out to the disk. Memory reservation prevents such performance degradation from happening. More details on memory management in vSphere can be found in VMware's memory resource management white paper [3].

Bypasses Virtualization Layers

- Once exclusive access to PCPUs is obtained, the feature allows the VCPUs to bypass the VMkernel's CPU scheduling layer and directly halt in the VMM, since there are no other contexts that need to be scheduled. That way, the cost of running the CPU scheduler code and the cost of switching between the VMkernel and VMM are avoided, leading to much faster VCPU halt/wake-up operations. VCPUs still experience switches between the direct guest code execution and the VMM, but this operation is relatively cheap with the hardware-assisted visualization technologies provided by recent CPU architectures [10][11].

Tunes Virtualization Layers

- When the VMXNET3 paravirtualized device is used for VNICs in the VM, VNIC interrupt coalescing and LRO support for the VNICs are automatically disabled to reduce response time and its jitter. Although such tunings can help improve performance, they may have a negative side effect in certain scenarios, which is discussed later in this paper. If hardware supports a pass-through mechanism such as single-root I/O virtualization (SR-IOV) and the VM doesn't need a certain virtualization features such as vMotion, NetIOC, and fault tolerance, we recommend the use of a pass-through mechanism for the latency-sensitive feature. Tuning techniques to improve latency for normal VMs (without the latency-sensitivity feature) can be found in the best practices white paper [4].

Using a Pass-Through Mechanism with the Latency-Sensitivity Feature

Although, the latency-sensitivity feature tunes the VMXNET3 device to reduce response time and jitter, packets still have to go through the whole network virtualization layer. Avoiding the network virtualization layer requires the use of a pass-through mechanism. Single-root I/O virtualization (SR-IOV) is a standard that enables one PCI Express (PCIe) adapter to be presented as multiple, separate logical devices to VMs [7]. On supported hardware, one or more VMs can be configured to directly access one or more of the logical devices directly. vSphere also supports VM DirectPath I/O, which allows VMs to access a wider range of network devices, but does not allow multiple VMs to share the same device. Please note that the use of a pass-through mechanism is incompatible with many of the benefits of virtualization such as vMotion (except DirectPath I/O with vMotion on the Cisco UCS platform), NetIOC, network virtualization, and fault tolerance. For configuring pass-through mechanisms including SR-IOV, refer to the *vSphere Networking Guide* for vSphere 5.1 or later [13].

Evaluation with Micro-Benchmarks

This section evaluates the new Latency-Sensitive feature of vSphere 5.5. Latency performance of VMs without latency-sensitivity feature can be found in the network I/O latency white paper [5].

Testbed

The testbed (shown in Figure 2) consists of a vSphere host running VMs and a client machine that drives workload generation. Both machines are configured with dual-socket, 6-core 2.5GHz Intel Xeon E5-2640 Sandy Bridge processors and 64GB of RAM. They are equipped with a 10GbE Intel 82599 NIC. Hyper-threading is not used.

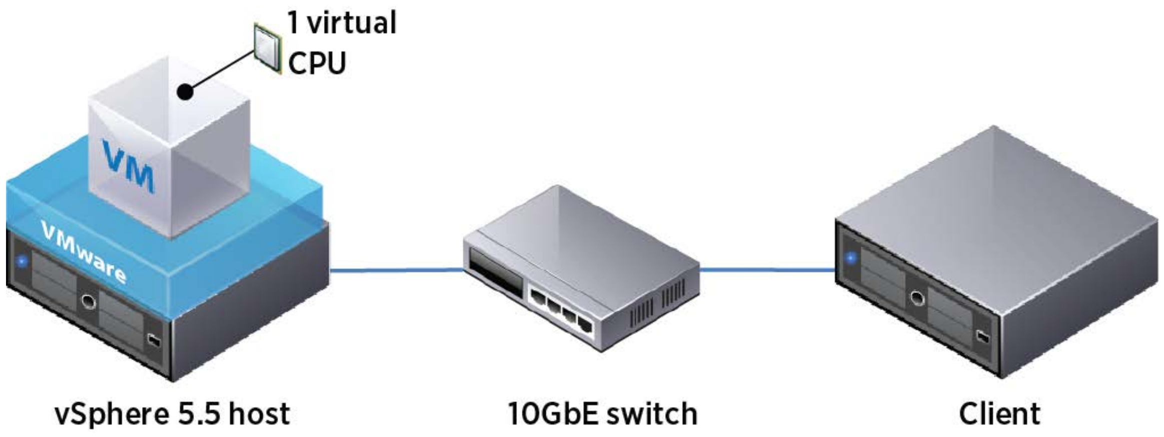


Figure 2. Testbed setup

The client machine is configured with native RHEL6 Linux that generates latency-sensitive workloads. The vSphere server hosts VMs that run the same version of RHEL6 Linux. A 10 gigabit Ethernet (GbE) switch is used to interconnect the two machines. All power management features are disabled in the BIOS and Red Hat Linux (by using the kernel boot option of intel_idle.max_cstate=0). The detailed hardware setup is described in Table 1. Inside the VMs, we set “nohz=off” in the kernel. PNIC interrupt coalescing/throttling is disabled in both machines. Firewalls are off on all machines and VMs.

	Server & Client
Make	Dell PowerEdge R720
CPU	2X Intel Xeon E5-2640 Sandy Bridge @ 2.5GHz, 6 cores per socket
RAM	64GB
NICs	1 Intel 82599EB 10Gbps adapter with Interrupt Throttling disabled

Table 1. Hardware specification

Workloads

Two different request-response type workloads are used to evaluate the latency-sensitivity feature: 1) Ping, 2) Netperf_RR. In this type of workload, the client sends a request to the server, which replies back with a response after processing the request. This kind of workload is very commonly seen: Ping, HTTP, and most RPC workloads all fall into this category. The key difference between Ping and Netperf_RR is that ping requests are processed in the kernel, whereas netperf requests are processed in user-space. This paper primarily focuses on configurations in a lightly loaded system where only one transaction of request-response is allowed to run per machine (or VM for a VM configuration), unless specified otherwise. Any overhead induced due to additional effort performed by the virtualization layer then will be directly shown as an extra latency. This is one of the main reasons that request-response workloads can be categorized as latency-sensitive.

- **Ping** – Default ping parameters are used except that the interval is set to .001 seconds (1000 ping requests are sent out for every second from the client machine) and a 256 byte of message size is used.
- **Netperf_RR** – The Netperf micro benchmark [5] is one of the most popular network performance benchmarks. The TCP_RR type is used to generate a request-response workload in TCP. A 256-byte message size is used.

Metrics

- **Median response time** – This metric is used to compare the average behavior between various configurations. Median is used instead of mean, since large outliers may distort the latter.
- **90/99/99.9/99.99 percentile response time** – In extremely latency-sensitive applications, the worst case timing behavior (jitter in response time) is equally or more important than the average behavior. For example, response time may need to be bounded to some threshold (or deadline) or every (network) transaction needs to be finished at a certain fixed interval. Exceeding those requirements can be considered as failure. In order to observe the trend in the tail of the response time distribution, 90, 99, 99.9, and 99.99 percentile values are used to measure jitter of the workload.

Configurations

- **VM-Default** – This is the default configuration without any changes to VMs or the vSphere host except those explained in the “Testbed” subsection above.
- **VM-SRIOV** – In this configuration, SR-IOV is enabled for the VMs.
- **VM-Latency-SRIOV** – In this configuration, the latency-sensitivity feature is enabled and SR-IOV is enabled for VMs.
- **Native** – In this configuration, both the client and server machines run native RHEL6 Linux.

The evaluation results with the Ping workload are presented first, followed by the results with the Netperf workload. Only one session of the workload is generated per VM (or per machine for the Native configuration).

With the three VM configurations (the VM-Default, VM-SRIOV, VM-Latency-SRIOV), we vary the number of VMs (1 and 4 VMs). VMs are configured with 2 VCPUs and 2GB RAM. The Native configuration is only compared with the 1 VM setup, since there is no straightforward way to match the 4VM setup. The server machine for the Native configuration is configured to have only 2 CPUs online for fair comparison with the 1 VM configurations.

In both 1 VM and 4 VM setups, the number of VCPUs is less than that of PCPUs so that the vSphere host is able to give all VMs exclusive access to PCPUs when the latency-sensitivity feature is enabled. The entire guest memory is reserved for the VM-SRIOV and VM-Latency-SRIOV configurations to properly enforce the pass-through mechanism and the latency-sensitivity feature.

Ping Evaluation

Figure 3 shows the comparison of the three different VM configurations on the 1 VM setup and the Native configuration using the five different metrics (the median, 90/99/99.9/99.99 percentiles). Note that the Y-axis is depicted in a logarithmic scale.

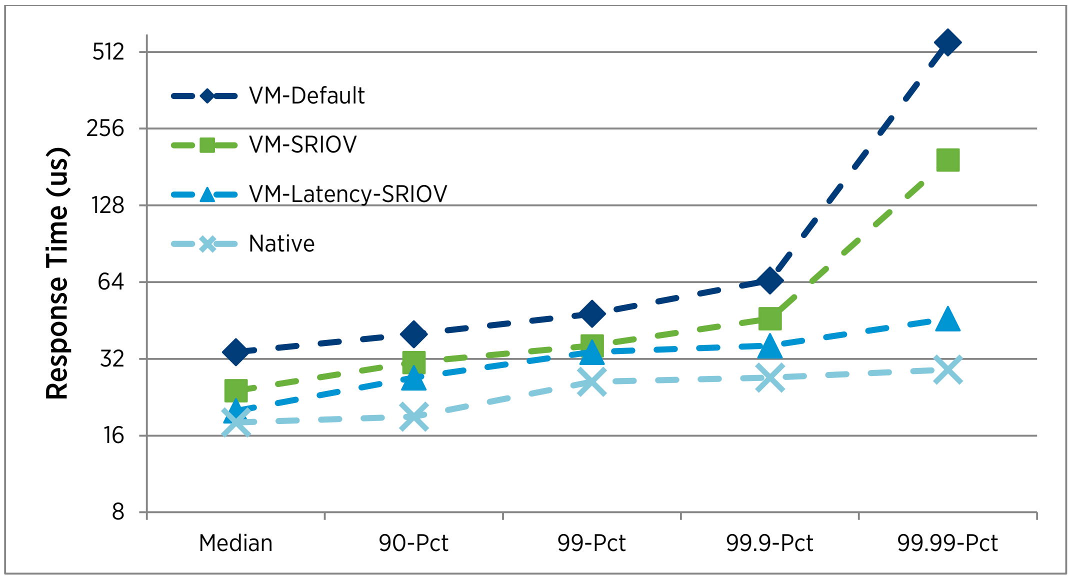


Figure 3. 1-VM Ping response time comparison

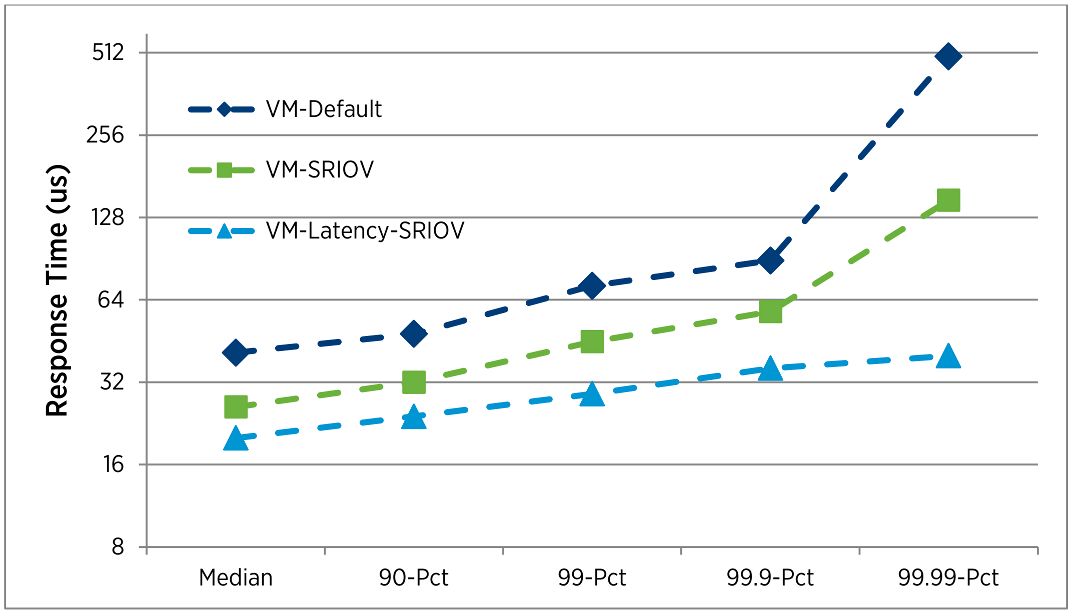


Figure 4. 4-VM Ping response time comparison

As expected, the Native configuration is the one that gives the best latency results for both the average timing behavior (represented by median values) and the worst case behavior (represented by 99/99.9/99.99 percentile values). The VM-Latency-SRIOV gives the second best performance, which is achieved by avoiding the network I/O virtualization layer and the CPU scheduling overhead (hence the cost of VCPU halt/wake-up operations). The SRIOV comes in the third place by avoiding the network virtualization layer, followed by the default.

For example, the median of the VM-Latency-SRIOV (20us) achieves a near-native performance (18us), which is considerably lower than that of the VM-SRIOV (with 24us) and the VM-Default (with 34us). The VM-Latency-SRIOV especially does a great job to keep the 99.99 percentile just at 46us, while the VM-default and VM-SRIOV yield a much higher value: 557us and 193us respectively.

Figure 4 shows the results with the 4VM setup. The Native configuration is not compared, since there is no straightforward way to match the 4VM setup. The Y-axis is depicted in a logarithmic scale.

The trend is very similar to the 1 VM setup: the VM Latency-SRIOV performs best in all metrics (median, 90/99/99.9/99.99 percentiles), followed by the VM-SRIOV and the VM-default configurations. The values are generally higher in all configurations compared to the 1 VM results except the median. This is because more packets need to be processed at the PNIC and they compete among each other for the processing. Also, the last level cache still needs to be shared by VMs that may add more jitter to response time. For the VM-default and VM-SRIOV configurations, running more VMs requires more work by the virtualization layer (network virtualization and CPU scheduling).

The results shown above clearly demonstrate that the latency-sensitivity feature combined with SR-IOV drives near-native performance in terms of average timing behavior and considerably improves jitter even with multiple VMs deployed on the same host.

Netperf_RR Evaluation

This section presents the results with the Netperf_RR workload. The same tests as the Ping workload are repeated; given the three VM configurations, the number of VMs is varied from 1 and 4. Each VM runs one Netperf_RR session and all VMs are given exclusive access to PCPUs automatically by the hypervisor. The Native configuration is only compared to the VM configurations on the 1 VM setup.

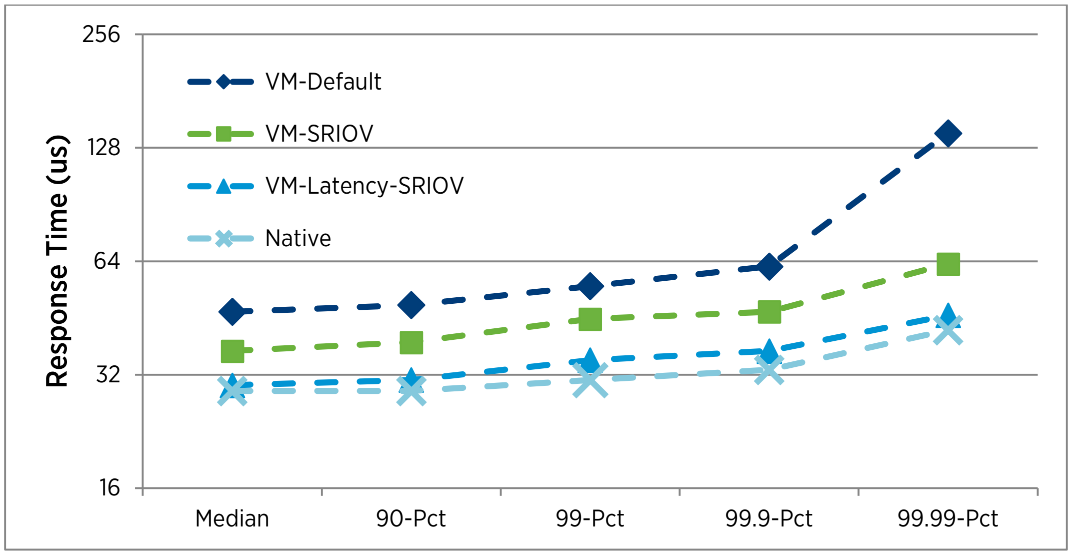


Figure 5. 1-VM Netperf_RR response time comparison

The results show exactly the same trend with Ping as depicted in Figure 5 (1 VM results) and Figure 6 (4 VM results). The Y-axis is depicted in a logarithmic scale.

Following the Native configuration, the VM-Latency-SRIOV is the one that gives the best latency results among the three VM configurations by avoiding the network I/O virtualization and VCPU halt/wake-up latency both for 1VM and 4VM setups. The VM-SRIOV comes next, followed by the VM-Default configuration.

Compared to the VM-Default, the VM-Latency-SRIOV considerably reduces the median response time from 47us to 30us in the 1 VM setup and from 56us to 33us in the 4-VM setup. Improvement is even bigger in jitter; for example, the VM-Latency-SRIOV reduces the 99.99 percentile by 87us compared to the VM-Default on the 4 VM setup, from 163us to 76us. This again clearly proves the efficacy of the latency-sensitivity feature.

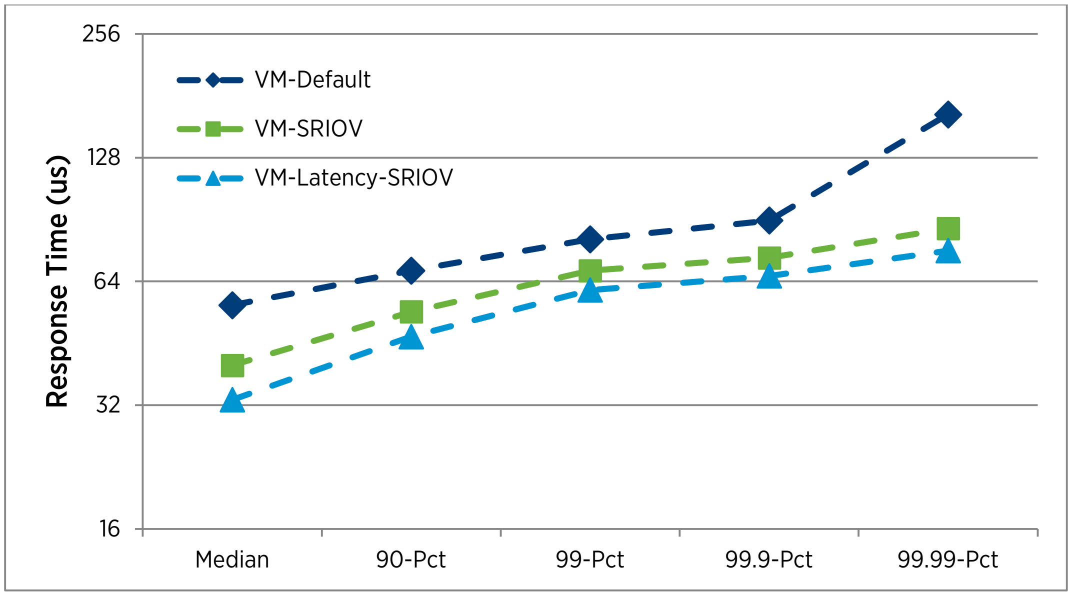


Figure 6. 4-VM Netperf_RR response time comparison

Achieving 10us Latency

Extremely latency-sensitive applications such as high frequency trading sometimes utilize specialized network devices (PNICs) to further reduce response time and jitter. Such specialized network hardware often employs proprietary techniques that bypass the OS's TCP/IP networking stack to achieve lower latency. This section evaluates latency using a Solarflare PNIC, SFC9020, with a TCP Onload feature [12]. The same testbed machines shown in Table 1 are used.

Configurations

- **Native-Intel** – In this configuration, both client and server machines run RHEL6 Linux. Intel 82599EB 10Gbps adapters are used to connect the two machines.
- **Native-SFC** – This configuration is exactly the same as the Native-Intel above except that Solarflare NICs are used instead of the Intel NIC. OpenOnload stack from Solarflare is used to fully utilize the benefits of Solarflare NICs.
- **VM-Latency-SFC** – In this configuration, the client machine runs RHEL6 Linux and the server machine runs a 2-VCPU VM on vSphere 5.5. Both machines are connected using SolarFlare NICs. The latency-sensitivity feature is enabled for the VM and the VM DirectPath I/O feature is used to allow for the VM to directly access the Solarflare NIC. Like the Native-SFC, the OpenOnload stack from Solarflare is used to fully utilize the benefits of Solarflare NICs.

The server machine of the two native configurations (Native and Native-SFC) is set to have only 2 CPUs online to match the number of VCPUs of the VM-Latency-SFC configuration.

Workloads

Since the benefit of using Solarflare NICs is better differentiated with the Netperf_RR workload than the Ping workload, only the Netperf_RR workload is used.

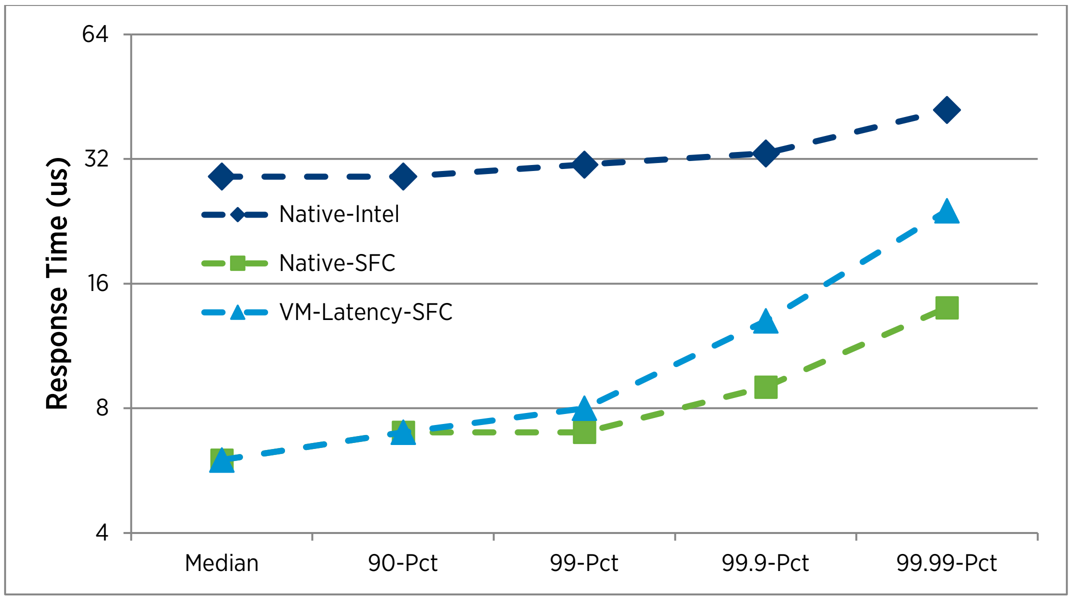


Figure 7. Netperf_RR response time comparison of three configurations: Native, Native-SFC, VM-Latency-SFC

Results

Figure 7 shows the results comparing the three configurations (Native-Intel, Native-SFC, VM-Latency-SFC). VM-Latency-SFC shows very close performance to Native-SFC. The median and 90th percentile values (6 and 7us respectively) show the same performance to Native. The 99th percentile shows only 1us in difference. Not surprisingly, Native-Intel is placed in the last place since it does not use a specialized NIC.

The results above clearly demonstrate that vSphere 5.5 is able to drive near-native performance even with specialized NICs with the latency-sensitivity feature enabled, achieving a very low latency in terms of both response time and jitter.

Side Effects of Latency-Sensitivity Feature

Impact of CPU Exclusive Access with the Latency-Sensitivity Feature

The evaluation so far is conducted in scenarios where there is not much CPU contention for the VMs, if any. In a real deployment scenario, it is common to run many VMs in a single host to efficiently utilize resources. Exclusive PCPU access enabled by the latency-sensitivity feature allows the VM to achieve near-zero VCPU ready time, considerably improving performance. However, such a benefit comes at an expense:

- The PCPUs exclusively given to a VM with latency-sensitivity enabled may not be used by other VMs even when the PCPUs are idle. This may lower overall CPU utilization and hence system throughput, negatively impacting the performance of other VMs without latency-sensitivity enabled.
- If a latency-sensitive VM uses VNICs, VMkernel I/O threads are used for network processing. Since VMkernel I/O threads are not allowed to run on the PCPUs exclusively owned by the VM, they become to share the remaining PCPUs with others. If those VMkernel I/O threads do not get enough CPU due to contention, the VM can experience a negative performance impact, even when all VCPUs are given exclusive access to PCPUs.

Table 2 and Table 3 show the impact of CPU exclusive access with the latency-sensitivity feature enabled. In this test, there are two VMs running a latency-sensitive workload (Netperf_RR with a 1024 message size) in the host. In the same host, there are four more VMs running a throughput-oriented workload that consumes high CPU time without any network I/Os. VMXNET3 devices are used for all VMs because we are only interested in examining the effect of enforcing exclusive PCPU access.

Response times of VMs with latency-sensitive workload	Throughput of VMs with throughput workload
106us	209K trans/sec

Table 2. Run 1: No VMs are configured with the latency–sensitivity feature enabled.

Response times of VMs with latency-sensitive workload	Throughput of VMs with throughput workload
73us (31% improvement)	90K trans/sec (57% regression)

Table 3. Run 2: The latency-sensitivity feature is enabled only for the VMs with the latency-sensitive workload.

Once exclusive PCPU access is given to the latency-sensitive VMs (by enabling the latency-sensitive feature for the VMs and making sure these VMs have enough CPU allocation), they show much better response time (73us) than without it (106us), clearly demonstrating the benefit of exclusively owning PCPUs. However, once the two latency-sensitive VMs exclusively occupy PCPUs, throughput-intensive VMs experience a much higher ready time and, as a result, throughput drops from 209K transactions per second to 90K transactions per second.

Impact of Using VMXNET3 Device with Latency-Sensitivity Feature

When VMXNET3 devices are used (instead of a pass-through mechanism such as SR-IOV), the latency-sensitivity feature automatically disables the VNIC interrupt coalescing and LRO. While doing so helps to improve latency, it may in turn increase CPU cost both for the guest and the hypervisor as the packet rate becomes higher. VMs have to process more interrupts and the hypervisor has to wake up the VMs more frequently. This may negatively impact throughput due to an increased per-packet processing cost.

In summary, the latency-sensitivity feature disables the VNIC interrupt coalescing and LRO, which might have the following negative performance implication:

- Automatically disabling LRO and VNIC coalescing may increase per-packet processing cost, resulting in reduced throughput in certain scenarios such as workloads with a high packet rate.

Best Practices

Based on the results and discussions so far, we recommend the following best practices for virtualizing extremely latency-sensitive application using the latency-sensitivity feature.

- Set **Latency Sensitivity** to **High**. This requires 100% memory reservation.
- Consider reserving 100% of the CPU. This guarantees exclusive PCPU access, which in turn helps to reduce VCPU halt/wake-up cost.
- Consider overprovisioning PCPUs. This reduces the impact of sharing the LLC (last level cache) and also helps to improve the performance of latency-sensitive VMs that use VNICs for network I/O.
- Consider using a pass-through mechanism such as SR-IOV to bypass the network virtualization layer, if the hardware supports one and virtualization features such as vMotion and FaultTolerance are not needed.
- Consider using a separate PNIC for latency sensitive VMs to avoid network contention.
- Consider using NetIOC, if a pass-through mechanism is not used and there is contention for network bandwidth.
- Consider disabling all power management in both the BIOS and vSphere.

Conclusion

vSphere 5.5 provides a new feature called latency-sensitivity to better support extremely-latency sensitive applications. This paper explains the latency-sensitivity feature and evaluates its performance. With this feature enabled, it is demonstrated that both the average and worse case timing behaviors considerably improve, achieving much lower response time and jitter compared to the default configuration. Such a performance improvement is accomplished by removing the major sources of overhead:1) giving exclusive access to physical resources to avoid resource contention due to sharing, 2) bypassing virtualization layers to eliminate the overhead of extra processing, and 3) tuning virtualization layers to reduce the overhead.

Evaluation results show that the latency-sensitivity feature combined with pass-through mechanisms such as SR-IOV helps to achieve a near-native performance in terms of both response time and jitter. For example, ping (median) response time reduces from 34us to 20us, showing only 2us difference from a native setup (18us). Further, when used with a specialized NIC, 90th percentile response time of 6us is achieved with a Netperf TCP_RR workload, showing the same performance compared to a native setup.

References

[1] *VMware Virtual Networking Concepts.* VMware, Inc., 2007.
http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf

[2] *The CPU Scheduler in VMware vSphere 5.1.* VMware, Inc., 2013.
http://www.vmware.com/files/pdf/techpaper/VMware-vSphere-CPU-Sched-Perf.pdf

[3] *Understanding Memory Resource Management in VMware vSphere 5.0.* VMware, Inc., 2011.
http://www.vmware.com/files/pdf/mem_mgmt_perf_vsphere5.pdf

[4] *Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VM.* VMware, Inc., 2013.
http://www.vmware.com/files/pdf/techpaper/VMW-Tuning-Latency-Sensitive-Workloads.pdf

[5] *Network I/O Latency on VMware vSphere 5.* VMware, Inc., 2011.
http://www.vmware.com/files/pdf/techpaper/network-io-latency-perf-vsphere5.pdf

[6] *Voice over IP (VoIP) Performance Evaluation on VMware vSphere 5.* VMware, Inc., 2012.
http://www.vmware.com/files/pdf/techpaper/voip-perf-vsphere5.pdf

[7] *What's New in VMware vSphere 5.1 - Performance.* VMware, Inc., 2012.
http://www.vmware.com/files/pdf/techpaper/Whats-New-VMware-vSphere-51-Performance-Technical-Whitepaper.pdf

[8] *VMware Network I/O Control: Architecture, Performance and Best Practices.* VMware, Inc., 2010.
http://www.vmware.com/files/pdf/techpaper/VMW_Netioc_BestPractices.pdf

[9] *Performance and Use Cases of VMware DirectPath I/O for Networking.* VMware, Inc., 2010.
http://blogs.vmware.com/performance/2010/12/performance-and-use-cases-of-vmware-directpath-io-for-networking.html

[10] *Hardware-Assisted Virtualization.* Intel.
http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/hardware-assist-virtualization-technology.html

[11] *AMD Virtualization.* AMD.
http://sites.amd.com/us/business/it-solutions/virtualization/Pages/virtualization.aspx

[12] *Open Onload.* Solarflare.
http://www.openonload.org

[13] *vSphere Networking Guide for vSphere 5.1 update 1.* VMware, Inc., 2012.
http://pubs.vmware.com/vsphere-51/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-511-networking-guide.pdf

[14] *Host Power Management in VMware vSphere 5.* VMware, Inc., 2010.
http://pubs.vmware.com/vsphere-51/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-511-networking-guide.pdf

About the Author

Jin Heo is a Senior Member of Technical Staff in the Performance Engineering group at VMware. His work focuses on improving network performance of VMware's virtualization products. He has a PhD in Computer Science from the University of Illinois at Urbana-Champaign.

Lenin Singaravelu is a Staff Engineer in the Performance Engineering group at VMware. His work focuses on improving network performance of VMware's virtualization products. He has a PhD in Computer Science from Georgia Institute of Technology.

Acknowledgements

The authors would like to thank Ayyappan Veeraiyan, Bhavesh Davda, Fei Guo, Garrett Smith, Guolin Yang, Haoqiang Zheng, Julie Brodeur, Lance Berc, and Shilpi Agarwal for reviews and contributions to the paper.