Performance Evaluation of Network I/O Control in VMware vSphere® 6

Performance Study

TECHNICAL WHITE PAPER

vmware®

Table of Contents

Executive Summary

The Network I/O Control (NetIOC) feature in VMware vSphere® has been enhanced in vSphere 6.0 to support a number of exciting new features such as bandwidth reservations. After tests demonstrate the performance utility of these new features, the performance impact of the new NetIOC algorithm is explored. Later tests show that NetIOC offers a powerful way to achieve network resource isolation at minimal cost, in terms of latency and CPU utilization.

Introduction

Today's servers are increasingly multi-socket and multi-core, carry huge amounts of RAM, and can drive huge I/O bandwidth on network adapters. The current trend in datacenters is to consolidate large numbers of legacy applications running on legacy machines on a single server with the help of virtualization technology, thereby driving down the capital expenditure (CAPEX) and operational expenditure (OPEX) of managing a datacenter. A key requirement of such consolidation is to achieve isolation among competing virtual machines (VMs) running on the same physical host, so that the performance of one VM does not affect the performance of the other.

Network I/O Control (NetIOC) was first introduced in vSphere 4.1 [1] [2]. This feature ensures you have control over the desired network performance when multiple traffic sources contend for the same physical network resource.

In vSphere 6.0, VMware has further built on NetIOC features to deliver more predictable bandwidth. The goal of introducing these changes has been to allow tighter control on the network resources available to different classes of traffic, irrespective of the traffic originating from other classes of traffic on the host. Here are the key enhancements that NetIOC provides in vSphere 6.0:

- Bandwidth reservations for classes of traffic: You can specify the minimum bandwidth that must be reserved for a class of traffic. This guarantees that the bandwidth to the same class of traffic never falls below the specified threshold. As an example, if VM traffic is dedicated 5 Gbps of bandwidth, then the combined VM traffic is always guaranteed 5 Gbps of bandwidth, irrespective of traffic from other classes of service such as VMware vSphere ® vMotion®, NFS, VMware vSphere® Fault Tolerance (FT), and VMware Virtual SAN™.
- Bandwidth reservations for VMs: NetIOC also allows the ability to provide bandwidth reservations to each VM virtual adapter (vNIC), thus providing the ability to provide dedicated bandwidth reservations at a per VM granularity. NetIOC also allows you to create abstract network resource pools that can be attributed to a port group of a distributed virtual switch (DVS). Bandwidth reserved for a resource pool is available only to VM vNICs that are part of the port group associated with the resource pool.
- Load balancing: This feature allows VMware vSphere® Distributed Resource Scheduling™ (DRS) to migrate VMs within a cluster of vSphere hosts to accommodate bandwidth reservations assigned to VM ports. This powerful feature allows you to assign bandwidth reservations to VMs without worrying about hitting the reservation limit in a single host of the cluster.

The above features are in addition to NetIOC features already available in vSphere 5, such as:

- Resource isolation through resource pools
- Distributing bandwidth with fair shares
- Bandwidth limits
- Load-based teaming policies

The ability to assign bandwidth reservations, along with bandwidth limits and shares, provides you with immense flexibility to control and isolate network resources. A bandwidth reservation guarantees that the network port (the term network port is used in this paper to describe a VM vNIC, or a vSphere kernel NIC) is guaranteed a specified amount of transmit bandwidth under all circumstances. This is a much more powerful feature compared to the fair shares and bandwidth limit features available in previous versions of vSphere. While you could control

the relative priorities of different VMs by assigning different shares, the proportion of bandwidth assigned could have fallen to less than the desired expectation if there were a lot of competition between different traffic flows. Bandwidth reservation enforces a minimum guarantee and thereby provides a much easier way of consolidating VMs, guaranteeing them bandwidth, and not worrying about the effect of virtualization on application performance.

Note: In the context of a VM, the concepts of fair shares, bandwidth reservations, and bandwidth limits that are discussed in this paper apply only to *VM outgoing traffic*. This is important to mention because *VM incoming traffic* is not under NetIOC control because the origin of the traffic could be outside the cluster.

NetIOC Features

The complete set of features available in NetIOC for vSphere 6.0 is as follows:

- Bandwidth shares, reservation, and limits for the following classes of traffic:
  - Management traffic
  - VM traffic
  - NFS traffic
  - Virtual SAN traffic
  - iSCSI traffic
  - vMotion traffic
  - vSphere replication (VR) traffic
  - Fault tolerance (FT) traffic
  - vSphere data protection backup traffic

  Bandwidth reservations can be used to isolate network resources for a class of traffic. As an example, consider a Virtual SAN cluster where the same device may be used for all of the above classes of traffic. In this case, a chunk of bandwidth may be reserved for Virtual SAN traffic to ensure steady Virtual SAN performance, independent of what other factors are driving traffic in the Virtual SAN cluster.
- Bandwidth shares, reservation, and limit per VM vNIC. This feature allows network resource isolation for VM vNICs.
- Load balancing of VMs by DRS during VM power on. This DRS feature will take into account network bandwidth reservations in addition to CPU and memory reservations while recommending which host a VM should be powered on. However, if DRS is not enabled in the cluster, you will not be able to power on those VMs whose bandwidth reservations cannot be met due to bandwidth reservations on competing VMs on the same host.
- Load balancing of powered on VMs when bandwidth reservations are violated. If the network reservation of a powered on VM is reconfigured, this DRS feature migrates the VM to a different host where the bandwidth reservation can now be met.

Note: NetIOC is only available on a distributed virtual switch (DVS). The features of load balancing require that you enable DRS.

NetIOC Bandwidth Reservations

This section shows performance tests of the new features of NetIOC in vSphere 6.0: bandwidth reservations and VM load balancing.

Cluster Configuration

Performance engineers set up the following cluster to evaluate NetIOC performance. The cluster comprises two

Intel dual-socket Westmere-EX-based servers, which are known as the systems under test (SUTs). These two servers are connected with an Intel 10 GbE card to an Arista 10 GbE switch. The detailed configuration of the machines is described in Table 1. There is also a group of clients connected to the Arista switch to simulate VM network traffic. DRS automation is turned on for the cluster and a "Fully Automated" policy is chosen. DVS is configured across the cluster and the Intel 10 GbE adapters on both hosts are connected as uplinks. VMs are deployed on this cluster; each VM is configured to have network connectivity with the vmxnet3 guest driver to the DVS. A diagrammatic layout of the cluster is available in Figure 1.

	vSphere Host
Make	HP Proliant DL380 G7 server
CPU	2-socket, 24 core, HT enabled, Intel® Xeon® CPU X5660 @ 2.80GHz
RAM	256 GB
NICs	1 Intel 10 Gbps Ethernet adapter
Host OS	vSphere 6.0
VM OS	RHEL 6.3 Server
VM configuration	1 vCPU VM with 2 GB memory

Table 1. Hardware setup and virtual machine details

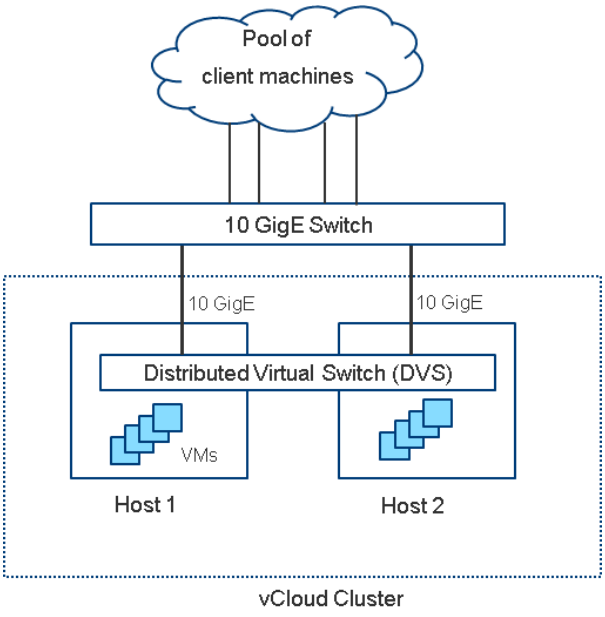

Figure 1. Test-bed layout

Bandwidth Reservations

This subsection evaluates how NetIOC bandwidth reservations work.

Test Methodology

The test is carried out in the following manner:

1. On host 1, 3 VMs are deployed and powered on.
2. A network resource pool is created that encompasses the vNICs of the 3 VMs, and this network resource pool is assigned a reserved bandwidth of 15 Gbps. 15 Gbps is the maximum reservation that can be assigned, since NetIOC imposes a limit of 75% of the total uplink capacity (2 x 10GbE). The 75% limit is imposed to avoid any class of traffic getting completely starved of network resources.
3. Each VM is assigned a "Normal" bandwidth share of 50 units, and no bandwidth reservation or limit is set up. Note that, by default, DRS would distribute the VMs across the two nodes of the cluster; therefore, DRS recommendations were not followed.
4. Netperf 2.6.0 is used to drive network traffic from the VMs. The network traffic is configured with a transmit workload such that each VM can achieve line rate throughput of 10 Gbps. However, since the three VMs are currently placed on a single host, they share the bandwidth.
5. Bandwidth reservations are set as follows (no other traffic is present in the cluster):
    a. A 7.5 Gbps reservation is assigned to VM1. Initially each VM is allocated bandwidth according to its share value. Since each VM has the same share, each is allocated an equal share of bandwidth. It is observed that the three VMs are able to send approximately 3.1 Gbps each. Please remember that the cumulative bandwidth is approximately 9.4 Gbps; lower than the network adapter line rate of 10 Gbps due to the overhead of TCP, IP, and Ethernet headers. Once VM1 is allocated a bandwidth reservation of 7.5 Gbps, it receives its minimum share. The remaining bandwidth is divided according to the VM shares. Since each VM has an equal share, the remaining bandwidth (9.4 – 7.5 = 1.9 Gbps) is divided by three, and allocated to the VMs. As a result, VM1 achieves (7.5 + 0.6 = 8.1) Gbps of bandwidth; while VMs 2 and 3 achieve 0.6 Gbps of bandwidth. This is shown in Figure 2.
    b. A 4.5 Gbps reservation is assigned to VM2. At this point, a fault is generated since a single host can no longer accommodate the bandwidth reservation of both VM1 and VM2. Once the fault is detected, DRS attempts to migrate VM2 to a host where the new reservation can be made. Note that DRS migrates the VM due to other faults too, such as uneven CPU or memory usage. However, these faults are not triggered in this test since the CPU and memory resource utilization are low. This migration may not happen immediately, but will happen within five minutes of the fault. Figure 2 shows that VM2 is moved to host 2. Once the migration happens, VM2 gets line rate bandwidth of 9.4 Gbps since it is not competing with any other VM for bandwidth on host 2. However, VM1 and VM3 are still competing for bandwidth on host 1. As a result, at this point in time, VM1 achieves 8.4 Gbps of bandwidth, and VM2 achieves 0.9 Gbps.
    c. A 3 Gbps reservation is assigned to VM3. This step assigns the remaining 3 Gbps of available VM reservation in the network resource pool to VM3. A fault is generated again because host 1 cannot accommodate a reservation of both VMs. DRS now migrates VM3 to host 2. At this point VM1 can achieve line rate bandwidth because there is no competition on host 1. VM2 and VM3 are allocated 5.4 Gbps and 3.9 Gbps respectively. Thus, reservations of all three VMs were met.

       Note that bandwidth limits could be set up in conjunction with bandwidth reservations. As an example, if a bandwidth limit of 7.5 Gbps were to be set for VM1, then VM1 would get no more than 7.5 Gbps under any circumstances.

Results

As the bandwidth reservations are assigned, the network throughput of the 3 VMs is determined. The result is shown in Figure 2. The VMs exhibit a very small performance dip at the time of migration, but there is no disruption of network connectivity.

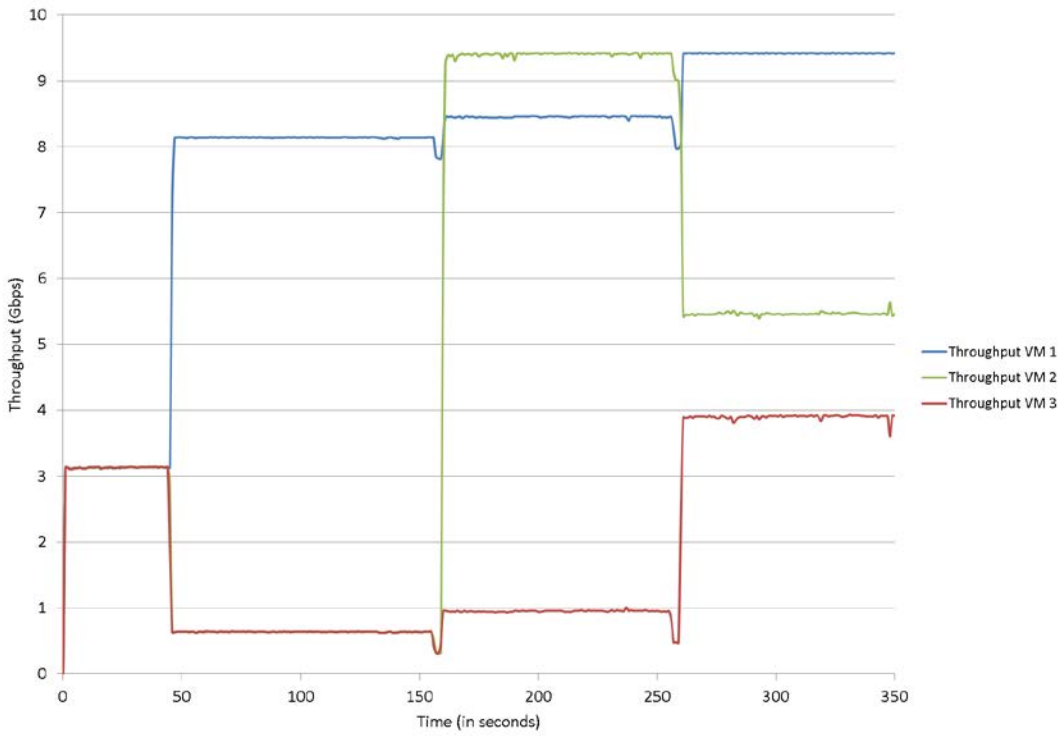

Figure 2. VM bandwidths as they are allocated NetIOC bandwidth reservations

One of the strongest features of NetIOC in vSphere 6 is automatic load balancing with DRS; steps a-c do not require you to think about the VM placement on hosts while deciding the reservation policy.

NetIOC Scheduler Performance Evaluation

This section shows the impact of the NetIOC scheduler on a vSphere host. Therefore, for the rest of this study, the DRS automatic migration policy will be disabled, and network traffic performance on a single vSphere host (Host 1) will be the focus.

NetIOC introduces an additional layer of packet scheduling at the hypervisor. A major enhancement in NetIOC in vSphere 6.0 with respect to previous NetIOC releases is that a separate scheduler queue is maintained for each virtual port. The network scheduler picks up packets from the network port queues and dispatches them for transmit over the network adapter, while making sure that bandwidth, shares, and limit settings are provided to each network port in NetIOC.

The implementation of the NetIOC scheduler is based on the hClock scheduler [3], please read the referenced paper for more details. The NetIOC scheduler consumes CPU while it maintains track of each network port, and decides which network port to pick for scheduling at a given point of time. Since network traffic is real-time, the scheduler needs to make its decision quickly and transmit fast enough to reach line rate on the device. Latency is another critical requirement of many applications, and the NetIOC scheduler must not add to the latency in the hypervisor. VMware designed NetIOC to facilitate efficient sharing of the network across different network ports, and deliver great throughput, latency, and system utilization at the same time.

Workloads

The following workloads are used:

1. Netperf: The TCP_STREAM and TCP_RR profiles of netperf 2.6.0 [4] are used to emulate bulk throughput and latency-sensitive traffic respectively. A message size of 16K and socket buffer size of 64K was used for the TCP_STREAM profile. The request and response size was set to 64 bytes for the TCP_RR profile.
2. Oliobench: Olio is an Apache Incubator benchmark [5] which emulates a setup where a number of users are interacting with a social networking website. A client machine simulating the users generates a request, and the Web server provides a response after querying the database. The key metric is the average response time, which represents the latency of serving a user request. The Oliobench is also a component of VMmark [6] and was chosen to showcase a realistic application using NetIOC.
3. Iometer: The Iometer [7] workload is used to generate NFS traffic to a remote mount. All NFS traffic was sequential read with 32K block size. Iometer was configured to run with 4 Iometer worker threads to drive high bandwidth NFS traffic.

NetIOC Performance Test Scenarios

The primary goal of NetIOC is to provide you with sufficient control so you can explicitly set the desired behavior of VM traffic in the event of competition for network resources. The Quality-of-Service (QoS) features that NetIOC provides can be well understood when traffic flows originating from different network ports compete with each other for the bandwidth of the physical NIC. The experiments below are designed to stress such scenarios of heavy competition.

Experiment 1: VM Throughput in the Presence of vMotion and NFS Traffic

In this experiment, 12 VMs are created, and each is assigned 1 VCPU, 2GB memory, and 1 vNIC connected to the DVS. All 12 VMs are powered on, starting with VM 1. Each VM runs the RHEL 6.3 server. All 12 vNICs are attached to a port group. VM traffic is assigned a bandwidth reservation of 4.8 Gbps. All the VMs are assigned the same share, and therefore the expectation is that each VM would be allocated a minimum of 400 Mbps of bandwidth under all circumstances.

To validate this theory, 4 netperf TCP_STREAM sessions are started from each VM. Two additional VMs are created:

- A Windows Server 2008 VM from which is run Iometer to simulate NFS traffic to a RAMDisk temp file system created on one of the clients
- An RHEL 6.3 VM with 16 GB memory, which will be migrated to Host 2, to simulate vMotion traffic.

The rationale behind doing I/O to a remote RAMDisk via NFS is to generate competing NFS network traffic over the same uplink. At the time of 15 seconds, Iometer starts NFS traffic from a Windows VM. At 90 seconds, the migration of the RHEL 6.3 VM is started. Note that no bandwidth reservations have been set for either NFS or vMotion traffic.

The throughput per VM is plotted in Figure 3. The following is observed:

1. Each of the 12 VMs gets an almost identical share of traffic.
2. Even in the presence of heavy vMotion and NFS traffic, the throughput for the bandwidth reserved vNICs does not drop below 500 Mbps, exceeding the expected 400 Mbps expected bandwidth reservation.

These results show that NetIOC is able to fairly guarantee all reservation bandwidths even when stressed with heavy traffic from different types of service.

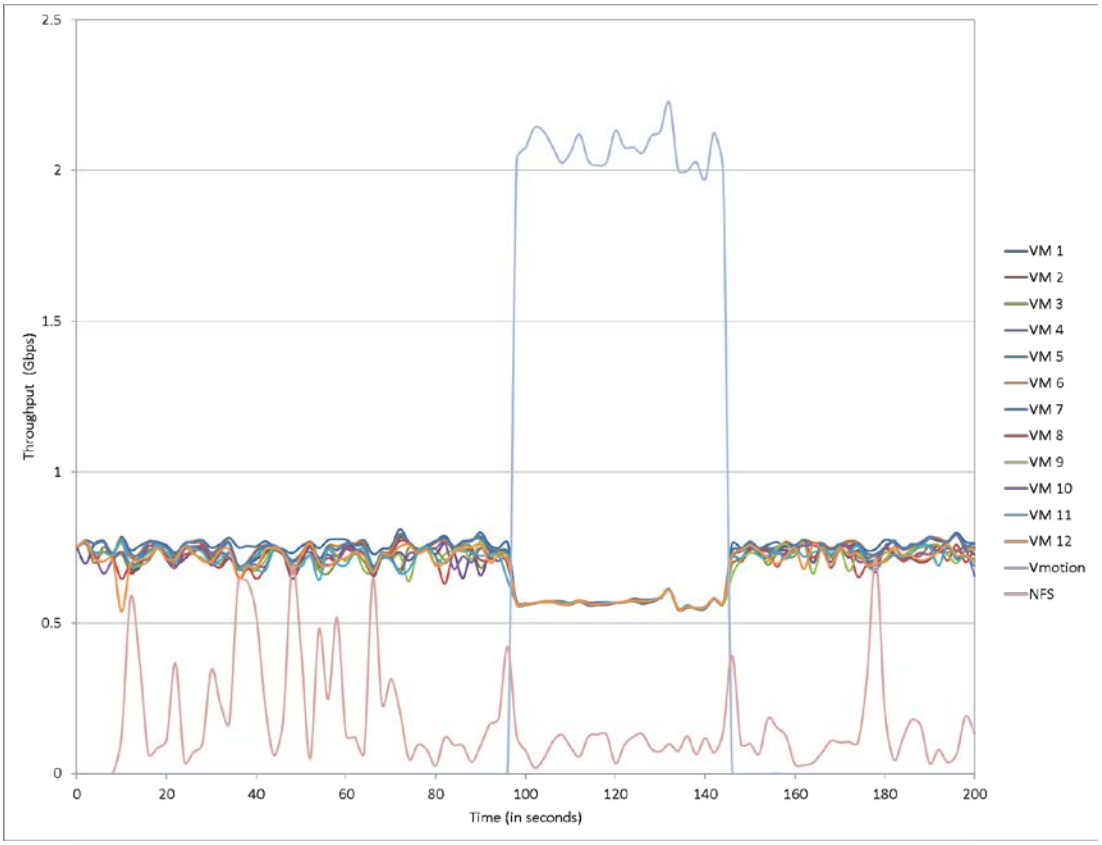

Figure 3. Throughput across VMs in presence of vMotion and NFS traffic.

Experiment 2: Response Time on Latency-Sensitive VM Running Oilo in Presence of Very Heavy Traffic from Competing VMs

Netperf generates predictable synthetic traffic, but it is not the best representation of real world traffic. The objective of this experiment is to understand how NetIOC policies can help realistic workloads and benchmarks. For this purpose, Olio is used. This is a Web 2.0 toolkit, which benchmarks the response time of a Web server. Olio is available as an Apache Incubator project, and is also one of the components of VMmark. The deployment of Olio for testing here consists of a VM which hosts the Web server and a separate VM which hosts the database. Both VMs are created with 2 vCPUs and 2 GB memory reach, and run the RHEL 6.3 OS. Both VMs are located on the same host (Host 1); therefore, the traffic between the Web server and database will be inter-VM traffic. The client machines generate requests to simulate 150 and 500 users, with a ramp-up time of 60 seconds, and a steady run time of 300 seconds. The traffic between the client machines and Web server is over the 10 GbE NIC. Since this is shared with other VMs, a bandwidth reservation of 500 Mbps is assigned to this Web server VM to achieve resource isolation. The goal is to measure the average response time for each request, and understand how competition from traffic originating from other VMs can affect the response time.

To simulate the competing traffic, 1 netperf session is started with the TCP_STREAM profile from each of the 12 VMs created in Experiment 1. However, the bandwidth reservations that were set in Experiment 1 have been removed. This generates sufficient traffic to cause contention on the bandwidth of the 10 GbE NIC.

Figure 4 compares the response time reported by Olio for the following cases:

1. Without any competing traffic
2. With the default network scheduler

3. With the NetIOC scheduler, but no bandwidth reservations assigned
4. With the NetIOC scheduler, and bandwidth reservation enforced for the vNIC of the database VM

No bandwidth reservation was set for any of the competing 12 VMs. Simply invoking the NetIOC scheduler without bandwidth reservations gives some benefit because NetIOC ensures a fairer distribution of network resources across all VMs. Once bandwidth reservations are enforced with the NetIOC scheduler, it is observed that the increase in response time in the presence of competing traffic is limited to 15%. This exercise further demonstrates the benefit of prioritizing traffic classes with the NetIOC scheduler.

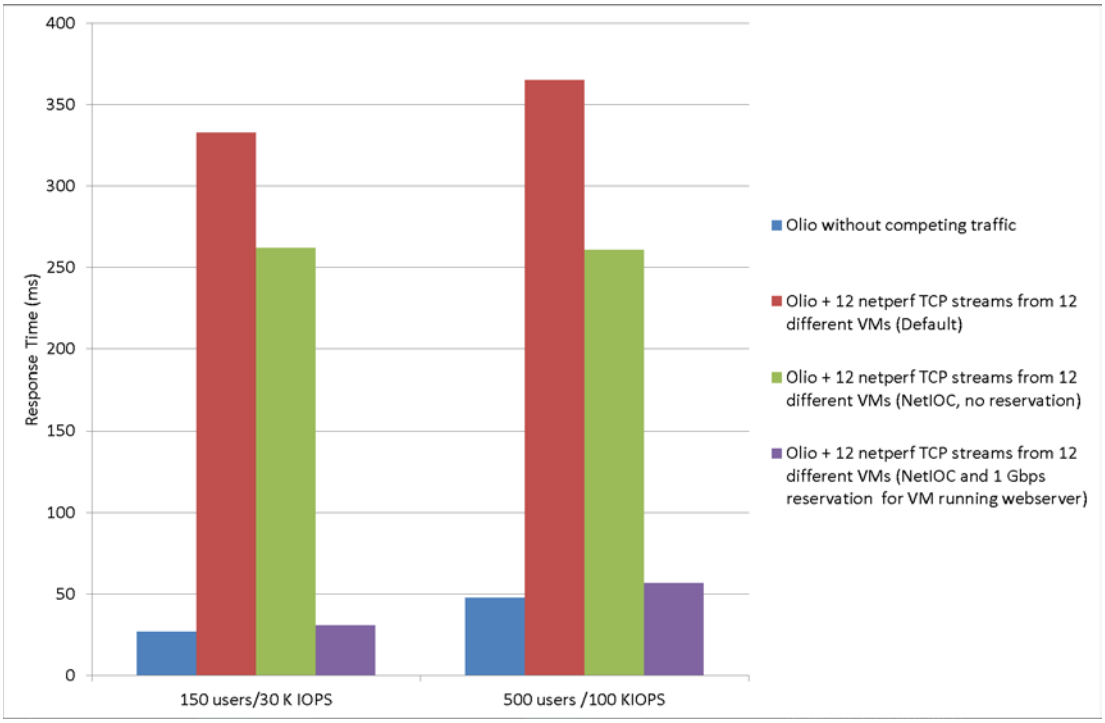

Figure 4. Comparison of response times seen by the Olio server

Experiment 3: Comparison of CPU Utilization Between NetIOC and the Default Scheduler

A key concern with every scheduling algorithm is the overhead of the scheduling computation. VMware's implementation of the NetIOC scheduling algorithm in vSphere 6 has several innovations designed to minimize the scheduling overhead. Some of these are described in the referenced paper [3].

Figure 5 compares the aggregate system CPU utilization between the default and NetIOC scheduler as the number of VMs is increased from 1 to 100 VMs, with each VM running netperf with the TCP_STREAM profile.

In the case of the NetIOC scheduler, a reservation of 7.5 Gbps was assigned for VM traffic. It is observed that with less than 25 VMs, the CPU utilization of the NetIOC scheduler is within 4-5% compared to the default scheduler. At a larger number of VMs, the overhead is a bit higher: at 100 VMs it is about 12%. This difference is actually not because of the complexity of the scheduler, but because of the fairness in resource distribution that NetIOC provides. With the default scheduler, VM traffic is very bursty. NetIOC does a very good job of interleaving packets from each VM so that the throughput and latency from each VM is predictable and there is less burstiness.

However, this has the consequence of a higher number of guest interrupts, because interrupts were no longer coalesced for a single burst of traffic, which happened for the default network scheduler. The majority of the increased CPU utilization in NetIOC is attributed to the higher guest interrupt servicing.

If the higher CPU utilization of NetIOC is an issue, you can decrease the guest VM interrupt rate by adjusting the interrupt coalescing parameter for the VM. To do so from the vCloud interface:

1. Go to VM Settings → VM Options → Configuration Parameters → Edit Configuration.
2. Add an entry for ethernetX.coalescingParams (where X is the interface number, for example, ethernet0). The default interrupt rate is 4000 per second.

Caution: Reducing the interrupt coalescing rate may lead to higher latency for small packets

Figure 6 compares the average latency seen in a netperf TCP_RR profile test across each of 12 VMs between the default and NetIOC schedulers. It is observed that NetIOC leads to a latency overhead of 4-5 us, which should be in the tolerance range for almost all applications.

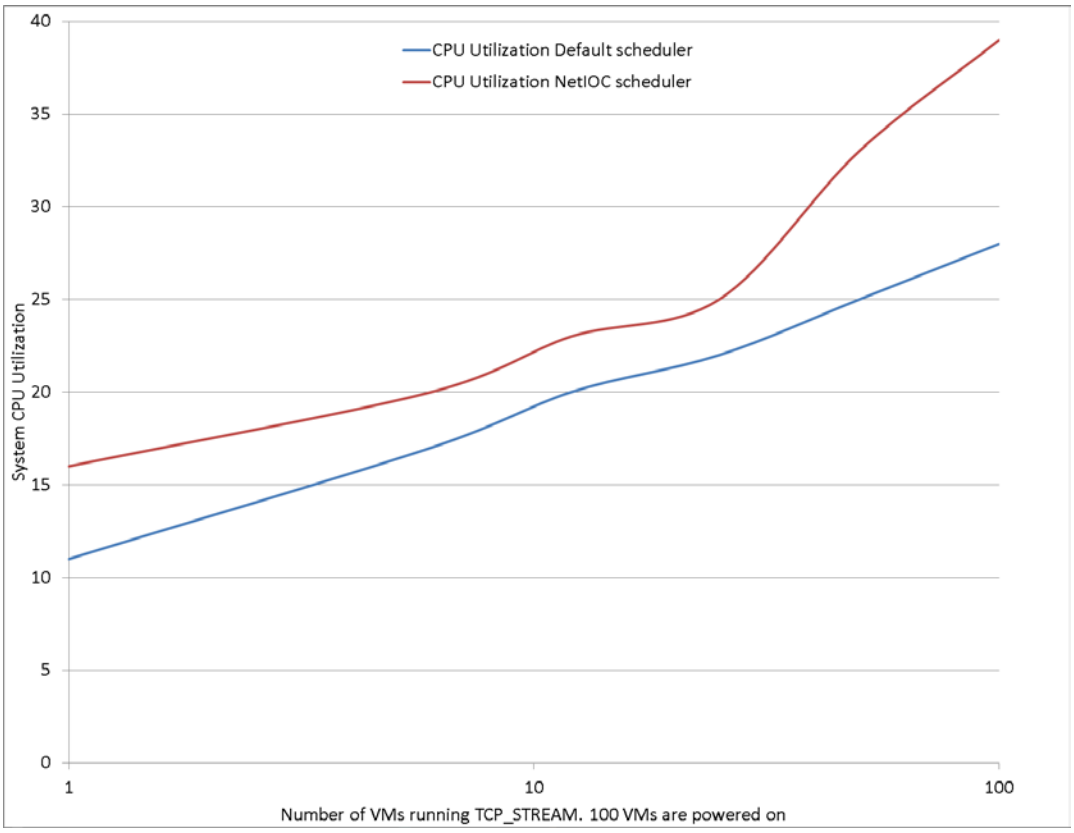

Figure 5. Comparison of aggregate CPU Utilization between NetIOC and default schedulers.

There are two main reasons behind this increase in latency:

1. NetIOC's scheduling algorithm adds an additional scheduling layer which slices the NIC bandwidth to meet the NetIOC reservation, limit, and share policies. Packets may queue up at the network port when the NIC transmit queue is being programmed by NetIOC. This might lead to a slightly higher wait time for a packet.
2. The current implementation of NetIOC is limited to using one transmit DMA channel (transmit queue) of the NIC. This is necessary to impose the constraints of bandwidth reservations. On the other hand, the default scheduler implements dynamic load balancing between network queues, and may use multiple network transmit queues depending on the traffic workload. Using multiple transmit queues prevents packets of different network connections contending for DMA transaction to a common queue. If the relatively small latency overhead is a concern for your application, please refer to the following section on how to handle latency sensitive traffic with NetIOC.

For the same reason, NetIOC is not recommended in situations where very high packet rate (greater than 1 million packets per second) is expected.

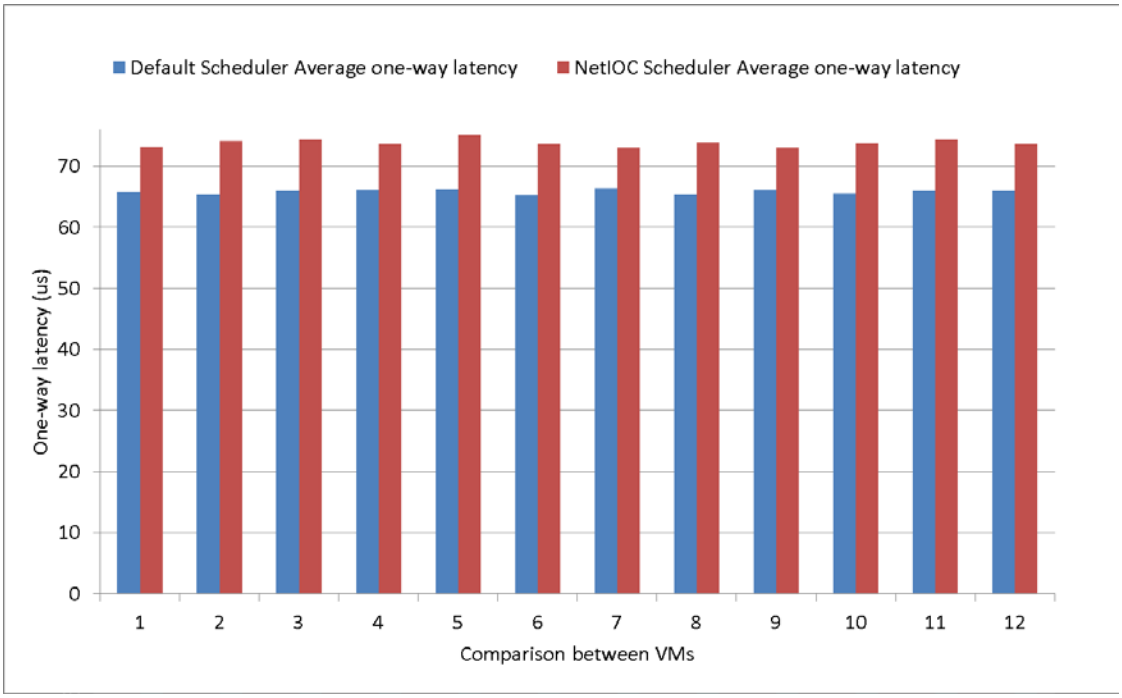

Figure 6. Comparison of one-way latency between default and NetIOC schedulers

Best Practices while Using NetIOC

NetIOC is a very powerful feature, and it is best to understand the various options available in NetIOC and their impact. This section describes some best practices for you to configure NetIOC.

Setting Bandwidth Reservations

It is often challenging to come up with the correct bandwidth reservation policy. One of the great features of NetIOC is that if the traffic demand falls below the reserved bandwidth, any unutilized bandwidth is redistributed to other network ports that might require bandwidth. Thus, allocating bandwidth reservations in excess of the demand is fine with the following caveats. First, excess reserved bandwidth reduces the remaining bandwidth available for reservations. Moreover, the instantaneous throughput on a device can often drop below the line rate because of other bottlenecks in the network such as some congestion or receive side flow control. In such a case, where NetIOC is unable to meet the all bandwidth reservations, it allocates bandwidth proportional to allocated reservations.

Handling Latency Sensitive Traffic

While NetIOC does a great job in limiting the increase in latency of a network port due to competing traffic from other network ports, there is still some increase due to sharing of resources. Experiment 2 demonstrated that the latency increase for a Web server is not more than 15% once NetIOC bandwidth reservations are in place. This latency increase should be in the tolerance zone for most applications. However, in certain situations, an application can be extremely latency sensitive. In such cases, you can limit the impact to latency from competing VMs by reducing the maximum time for which the uplink device can delay transmit completions. By default, the value of this parameter is set to 33 us. The parameter is Net.NetSchedCoalesceTxUsecs in the Advanced

System Settings tab for the host. Note that changing this parameter value adjusts the corresponding physical NIC interrupt throttle rate. Lowering Net.NetSchedCoalesceTxUsecs leads to a higher interrupt throttle rate, and thereby higher CPU utilization. Lowering Net.NetSchedCoalesceTxUsecs below a certain value will also limit the NIC from reaching line rate throughput. Please remember to keep Net.NetSchedCoalesceTxUsecs consistent across all hosts in the cluster.

Setting Shares and Bandwidth Limits

NetIOC in vSphere inherits the features of shares and bandwidth limits from previous versions of vSphere, and these features can be set in tandem. Shares are useful when a relative proportion of bandwidth must be assigned between two network ports. For example, if you want two network ports to achieve equal share in bandwidth, then you should allocate them an equal number of shares. It is important to remember that with NetIOC, the algorithm considers shares only after all bandwidth reservations are satisfied.

Likewise, you can set bandwidth limits if you want a class of traffic to never be allocated more than the specified amount of bandwidth. It is important to remember that the bandwidth will be capped for the network port even if the bandwidth is available on the NIC. Therefore, make sure your intention is clear before you place a limit.

Achieving the Full Reserved Bandwidth

In some cases, you might be unable to achieve the bandwidth reservation set through NetIOC. As an example, traffic driven by a VM may fall below the NetIOC bandwidth reservation. Sometimes, the observed throughput might be lower than expected because of the way TCP semantics interact with NetIOC scheduling. One best practice is to increase the guest OS socket buffer size and the maximum TCP congestion window, so that the TCP window is large enough to drive high throughput.

Conclusion

This white paper discusses several unique features of the NetIOC scheduler in vSphere 6. It demonstrates how to guarantee network resources to a VM, thereby guaranteeing the desired network throughput. This paper also demonstrates how DRS is able to load balance VMs across hosts to meet the bandwidth reservation of each VM. The overhead of NetIOC in terms of CPU Utilization is quite reasonable. Testing shows that NetIOC will further help in the consolidation of VMs on a physical machine in a datacenter, without sacrificing the performance of any application. Finally, some best practices are suggested to keep in mind while employing NetIOC.

References

[1] VMware, Inc. (2011) VMware vSphere 4.1 Networking Performance. http://www.vmware.com/files/pdf/techpaper/Performance-Networking-vSphere4-1-WP.pdf

[2] VMware, Inc. (2011) VMware Network I/O Control: Architecture, Performance and Best Practices. http://www.vmware.com/files/pdf/techpaper/VMW_Netioc_BestPractices.pdf

[3] J. Billaud and A. Gulati, "hClock: Hierarchical QoS for Packet Scheduling in a Hypervisor," in *8th ACM of Eurosys*, Prague, Czech Republic, 2013.

[4] VMware, Inc. Netperf 2.6.0. http://www.netperf.com

[5] Olio Web 2.0 toolkit. http://incubator.apache.org/olio

[6] VMware, Inc. (2015) VMmark 2.x. http://www.vmware.com/products/vmmark/overview

[7] Iometer Project. http://www.iometer.org

About the Author

Amitabha Banerjee is a staff engineer in the I/O Performance Engineering group at VMware. His work strives to improve the performance of networking and storage products of VMware.

Acknowledgements

The author would like to thank Shilpi Agarwal, Boon Ang, Julie Brodeur, Jin Heo, and Guolin Yang for their contributions to NetIOC performance improvement and to this whitepaper.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com
Copyright © 2015 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at http://www.vmware.com/go/patents. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. Date: 3-Mar-15 Comments on this document: http://communities.vmware.com/docs/DOC-28917