PERFORMANCE STUDY – MARCH 2017

PERFORMANCE IMPLICATIONS OF STORAGE I/O CONTROL–ENABLED SSD DATASTORES

VMware vSphere 6.5

vmware®

Table of Contents

Executive Summary

The Storage I/O Control (SIOC) feature in VMware vSphere® 6.5 is designed to differentiate performance between high priority virtual machines and low priority ones. Additionally, SIOC allows administrators to allocate absolute IOPS to virtual devices to meet performance requirements.

SIOC offers dynamic control of I/O devices per user-specified policy. VMware introduced this feature in vSphere 4.1 and extended it to network attached storage (NAS) in vSphere 5.1, but at that time, SIOC supported only share-based policies. In vSphere 6.5, SIOC also supports operation-based policies, and now administrators can specify an absolute IOPS number to VMs by assigning the maximum IOPS as a limit and the minimum IOPS as a reservation.

IT professionals can also use SIOC in datacenters designed with solid-state drives (SSDs) in their hierarchies. Even though SSDs can create a situation of very low I/O latency and make it difficult to detect I/O congestion, SIOC can be triggered by either the absolute latency value or latency increase percentage compared to normal situations, and, hence, it can detect storage contention effectively. We conduct all of the experiments in this paper on an SSD datastore.

Experiments in the VMware storage performance lab show:

- SIOC schedules IOPS based on user-provided policies. Users can separately specify the share (the proportion of IOPS allocated to the VM), the limit (the upper bound of VM IOPS), and reservation (the lower bound of VM IOPS). SIOC is triggered automatically when there is a contention, and it tries to guarantee the system's performance based on the user's policy.
- Users can combine the three performance features (share, limit, and reservation) together to give a detailed performance specification (policy) for each VM. We have tested the sample policies, provided in a vSphere cloud VM, for high priority, medium priority, and low priority VMs in our experiments to prove the efficiency of policy-based management.
- In real datacenters, over a long period, storage usage is not stable. It is very normal to see sudden spikes in I/O requests amid a low-demand period. We mimick the datacenter environment by introducing different types of VMs like logging, production, and others. We assign each of the VMs a different priority level, and we test the system performance over a long period to make sure that the overall system performance is guaranteed.
- Uneven configuration is also very common among hosts. It is entirely possible that some hosts carry a heavier workload compared to others. We want to make sure that heavy computation demands on a host do not impact the I/O performance of the VMs located on that host. So, we create an uneven setting, using a shared datastore, and prove that the policy is still met, regardless of the number of VMs on each host.

Introduction

VMware virtualization technology gives IT professionals an efficient way to manage and allocate datacenter resources, including shared storage devices accessed through LUNs or NFS mounts. Shared storage reduces datacenter costs because the disk space needs of multiple VMs can fully utilize the same storage array and the active VMs can achieve higher IOPS than they would if they each had their own dedicated disk. This is because a storage array can typically manage many more I/Os than a smaller, dedicated disk, and administrators can over-commit VMs to fully utilize the storage.

However, there are two challenges that affect the expected IOPS on VMs:

1. Storage resource contention happens occasionally. All active VMs suffer from contention when they are competing for a limited resource at the same time.
2. With the introduction of fast storage devices like solid-state disks (SSDs), users expect to allocate the extra IOPS to high priority VMs. This requires datacenter management to treat virtual disks, which they allocate to different VMs, uniquely and to specify an exact performance expectation for each virtual device.

SIOC solves these problems. With SIOC, administrators can guarantee the desired level of IOPS for each VM. It provides fine-grained I/O control to datastores using shared storage. It automatically detects contention for storage resources and allocates resources based on user-specified policies when contention occurs. In addition to share-based policy, now vSphere users can provide operation-based policies. With policies, users can specify the requirement for storage IOPS using absolute numbers, in addition to using proportional values. This allows datacenter administrators to better allocate IOPS by specifying the minimum and maximum IOPS a VM can take. SIOC is available on all kinds of storage systems, including NFS and VMFS built on SSD and HDD disk arrays.

This paper shows the advantages of using SIOC to manage shared storage using SSD. We demonstrate that high priority VMs achieve more IOPS, and administrators can set the performance of VMs using explicit numbers, in addition to using proportional values that were available in vSphere 5.0 and 4.1.

Terminology

We use the following terms in this paper:

- **Share** – the proportion of IOPS assigned to a virtual disk. The proportion is calculated using $share / \sum_{VMDK}^{Datastore} Share$. The denominator is the sum of the shares of the virtual disks in the same datastore.
- **Limit** – the upper limit of IOPS, represented in I/O per second (IOPS). For instance, if we set the limit to 5000, the maximum IOPS allowed of that device is 5000 IOPS, even if we have an abundance of IOPS.
- **Reservation** - the lower limit of IOPS, represented in I/O per second (IOPS). For instance, if we set reservation to 5000, vSphere will try to guarantee the IOPS. But, if the hardware does not permit such a setting, the reservation will not come into effect.
- **Policy** – a combination of share, limit, and reservation. We have the following default policies:

	High Priority	Medium Priority	Low Priority
Share	2000	1000	500
Limit	100,000	10,000	1000
Reservation	100	50	10

Table 1. Default policy specification

- **Congestion threshold** – the threshold for the system to detect congestion. There are two types of congestion threshold:
  - **Fixed latency** - SIOC is triggered when the actual datastore latency exceeds the latency threshold,
  - **Percentile** - SIOC is triggered when the storage performance has fallen below the previously set percentile of the normal performance.

Experimental Environment

Sixteen VMs, which run the same workloads from the FIO micro-benchmark, are located on two hosts. For the first three tests, the VMs are equally distributed and hence each host runs eight FIO benchmark applications. For the uneven test, four VMs run on Host 1 and the remaining 12 VMs run on Host 2.

Table 2 describes the vSphere host and system configuration for the experiments, and Table 3 describes datastore properties.

Component	Details
Hypervisor	vSphere 6.5
Processors	Two Intel Xeon E5-2630 per host
Memory	64 GB per host
Guest Operating System	Ubuntu Linux
Virtual CPU / Memory	4vCPUs, 4 GB memory
Virtual Disk (OS / Data)	10 GB / 2 GB
File System	VMFS
Storage Server / Array	EMC VNX5700 Hybrid Storage Array
Test Application	FIO

Table 2. vSphere host and storage system configuration information

Datastore Name	Size	Purpose	SIOC Enabled	SIOC Triger
Data-SSD	200G	Holding all data disks	Yes	Latency > 5ms
OS-HDD	500G	Holding all OS disks	No	N / A

Table 3. vSphere datastore configuration information

Test and Measurement Tool

As mentioned above, we use FIO, a storage micro-benchmark, to generate workloads on the sixteen VMs. We set these workloads to produce storage I/O in four of the most commonly used I/O patters: sequential read, sequential write, random read, and random write. We use esxtop to report the I/Os per second (IOPS).

Setup for the Tests

Sixteen identical VMs run on two hosts, sharing one VMFS datastore. The datastore is built on the SSD disk(s) in an EMC VNX5700 hybrid storage array. The hosts are connected to the disk array via Fiber Channel through direct connect (no network switch). Both hosts have the same hardware configuration. All the VMs are thick-provisioned. The system configuration is shown in Figure 1.

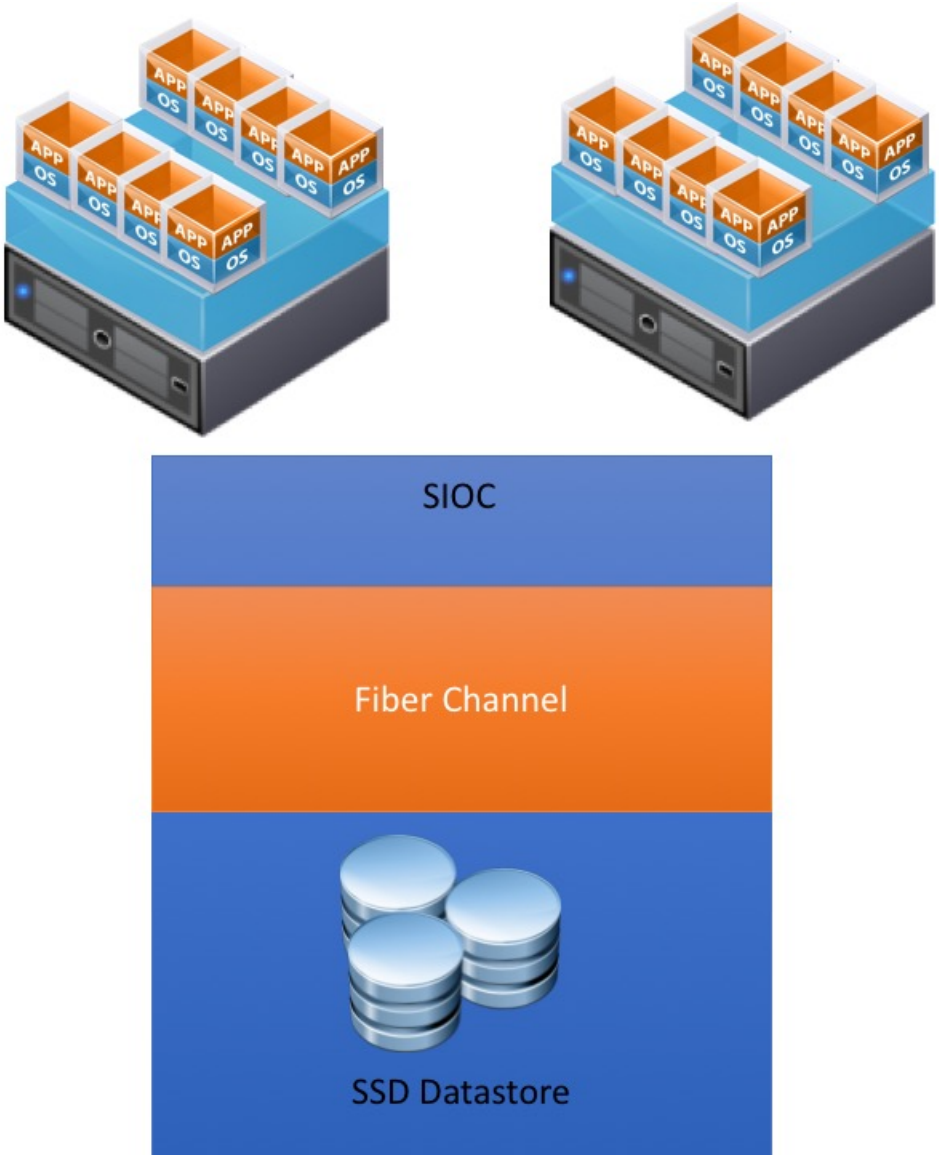


Figure 1. Test environment

Test 1: Performance Impact of Using a Single Feature in SIOC

This test case shows the impact SIOC can have on the performance of an application if we only set a single feature while the rest remains unset. The three features tested are share, limit, and reservation. In these tests, FIO is used to generate a workload to the shared datastore, which is controlled by SIOC. As described in the section, "Experimental Environment," there are sixteen identical VMs equally distributed on two hosts using the same datastore, which is built on SSDs. Each VM runs an instance of the FIO benchmark, and all of the VMs run the workload at the same time. For each of the tests, we change only the SIOC configuration on the first VM of each host, while the rest seven VMs remain at the default setting to act as a comparison group.

We run four tests, each representing a type of workload. The workloads are summarized in Table 4. We report the IOPS to measure the performance impact.

	Sequential Write	Sequential Read	Random Write	Random Read
Outstanding I/O	64	64	64	64
Block Size	16K	16K	16K	16K

Table 4. Summary for workloads and I/O pattern

Share

First, we change the share setting on the first VM of each host, while maintaining the default setting on the rest seven VMs on that host. So, we have the following configuration as shown in Table 5.

At the beginning of the test, all VMs have the same default SIOC share setting, which is 1000. After 1 minute, we change the share setting of the first VM on each host to 3000 while maintaining the settings on the rest of the VMs.

VM ID	Host ID	Disk Shares	
		Default	Test Setting
1	1	1000	3000
1	2	1000	3000
{2, 3, 4, 5, 6, 7}	{1, 2}	1000	1000

Table 5. Share settings for single feature test

The results are reported in Figure 2. Let's take sequential write as an example. Prior to changing the SIOC configuration, we have an IOPS of 2500 * 8 = 20,000 IOPS per host. After changing the share, there are in total 3000 + 1000 * 7 = 10,000 shares per host on the datastore. The high priority VM should get 3000 / 10,000 = 30% of the total IOPS, which is 20,000 * 0.3 = 6000 IOPS, which is accurately met by the test result. We also run and report the results of four most commonly used I/O patterns.

Figure 2. Performance impact of setting share in SIOC (y-axis shows IOPS)

Limit

Limit is the upper bound of the IOPS allowed for a virtual disk. It is commonly used to limit of the resource consumed by low priority applications to make room for the other time sensitive applications. The test settings are specified in Table 6.

VM ID	Host ID	IOPS Limit	
		Default	Test Setting
1	1	N / A	500
1	2	N / A	500
{2, 3, 4, 5, 6, 7}	{1, 2}	N / A	N / A

Table 6. Limit settings for single feature test

At the SIOC default setting, there is no limit set, which means there is no upper bound of IOPS for the device. In the test, we have set the limit to 500 IOPS for the data disk of the first VM on each host. So, the two VMs represent the VMs of less importance compared to others sharing the datastore at the same time.

Figure 3 shows the results. As we can see, there are two effects for setting limits. The first is that the performance of the low priority VM is limited to the set value. The difference between real IOPS and set value is less than 1%. The second is that the saved IOPS is equally shared among the rest of the VMs, because it is our goal to make full use of storage resources.

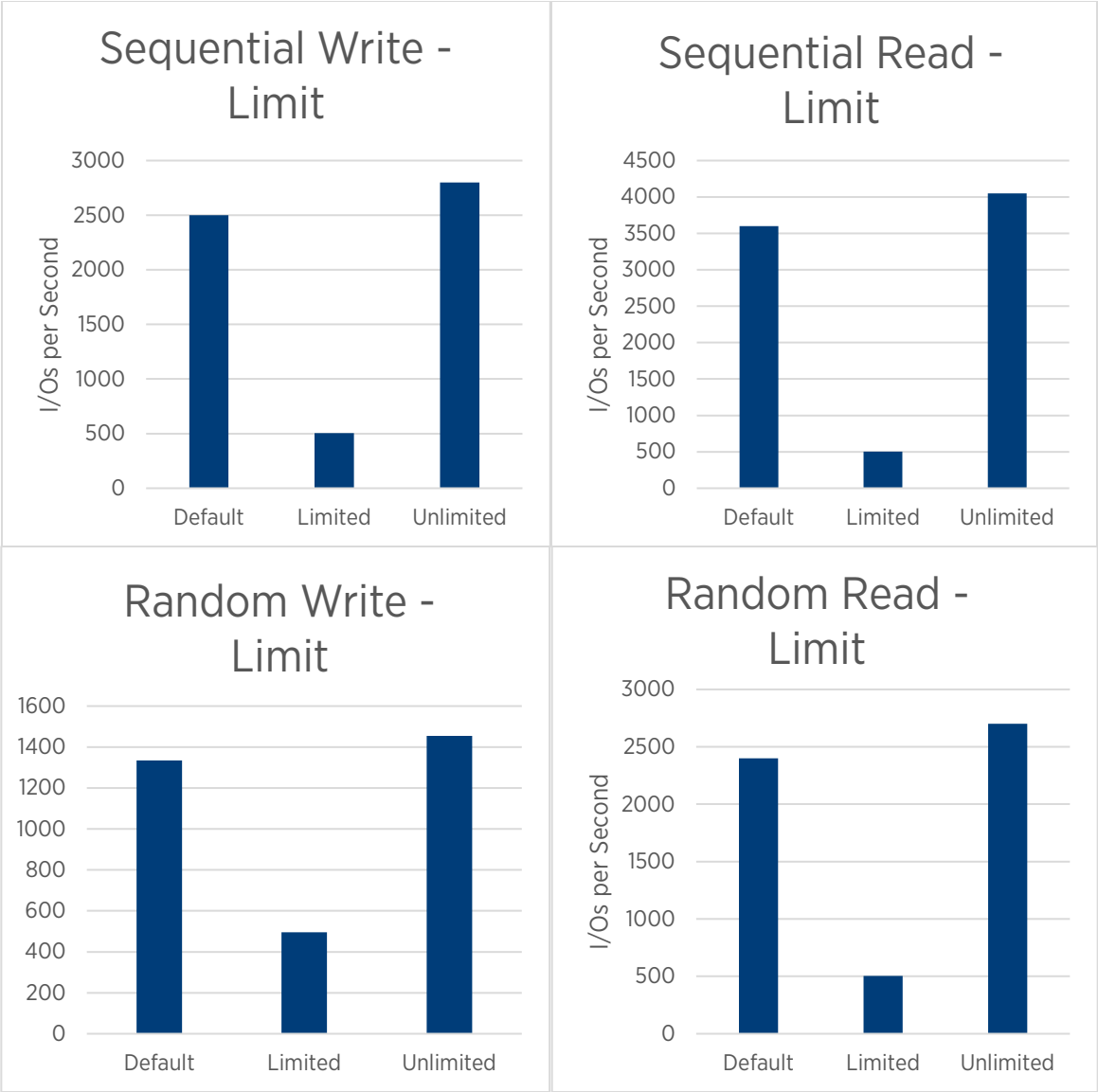


Figure 3. Performance impact of setting Limit in SIOC

Reservation

The opposite of limit, reservation is set as a lower limit of IOPS. It is used to guarantee the performance of high priority VMs. vSphere tries to guarantee the reservation setting, even if there is not enough resource. This means, if the reservation is set too high to meet, the system cannot guarantee the lower bound as a reservation, but it will try to do so by scheduling more I/Os to the favored VMDK.

In this part of the test, we set the reservation to be 100 IOPS more than the IOPS we get from high share disk in the tests for shares above. This is to make sure that our high reservation virtual disk achieves high IOPS within the provable area. The detailed settings are specified in the table below, where SW represents sequential write, SR represents sequential read, RW represents random write, and RR represents random read.

VM ID	Host ID	IOPS Reservation				
		Default	SW	SR	RW	RR
1	1	N / A	6843	9515	4130	7770
1	2	N / A	6843	9515	4130	7770
{2, 3, 4, 5, 6, 7}	{1, 2}	N / A	N / A			

Table 7. Reservation settings for single feature test

The test results are shown in Figure 4. It shows that all the reservation specifications are met. In all four I/O patterns, the less important VMs contribute IOPS to guarantee the performance of high priority VMs, while the total maximum IOPS remains unchanged.

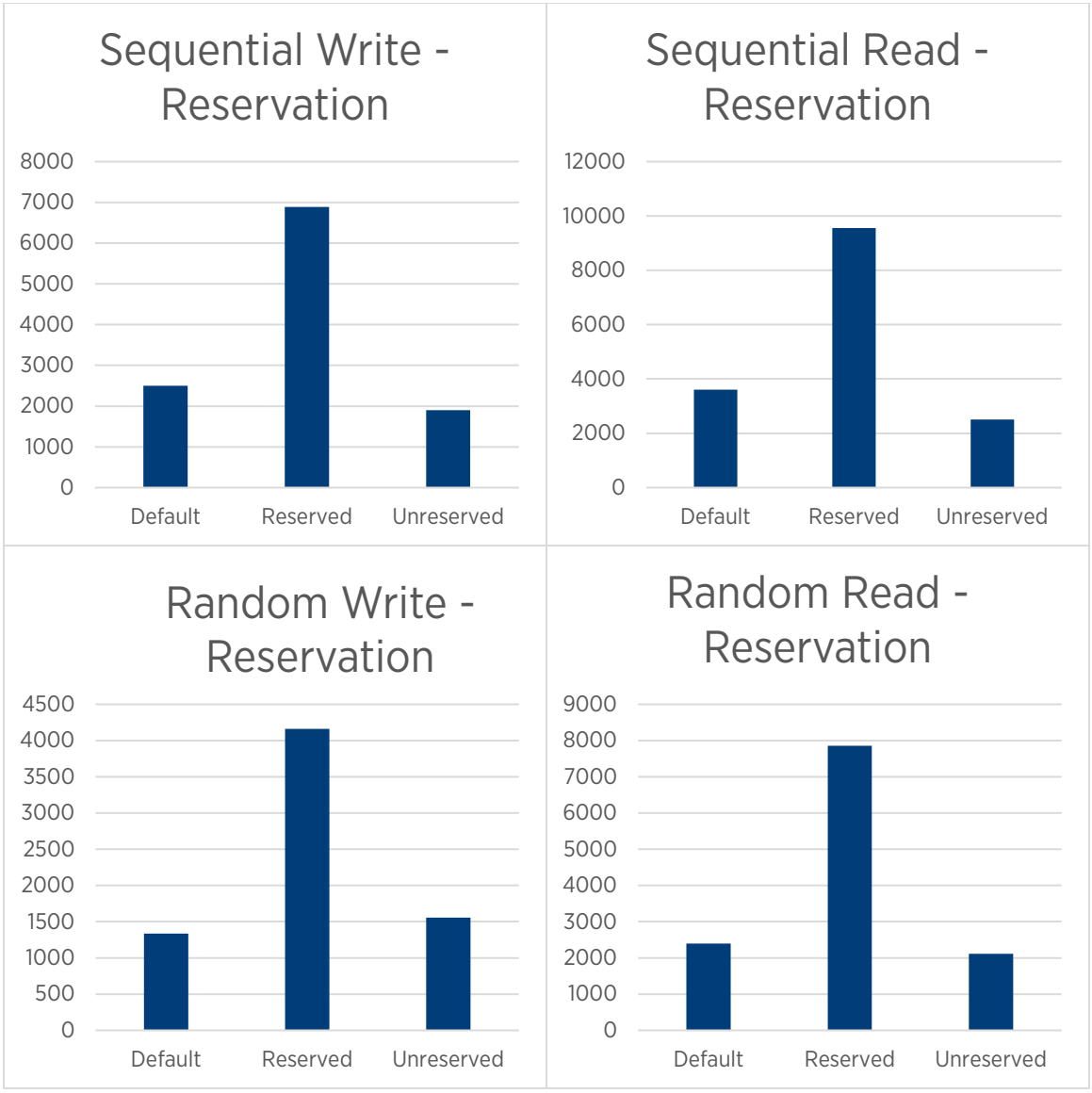


Figure 4. Performance impact of setting Limit in SIOC

Test 2: Performance Impact of Using the Combined Feature in SIOC

In this experiment, we test the effect of using all the SIOC features together to form a policy. We use the default policies as specified in Table 1, where there are three different priority settings: high, medium, and low; and their assignments are shown in Table 8.

	High	Medium	Low
Number of VMs per host	2	4	2
Block Size	256K	256K	256K
Outstanding IOs	10	10	10

Table 8. Combined test setting

Even though the I/O patterns of sequential write, sequential read, random write, and random read remain the same as the tests above, we have changed block size and number of outstanding I/Os. We use a large block size and low outstanding I/Os to mimic the environment for database and logging applications. The test result is shown in Figure 5.

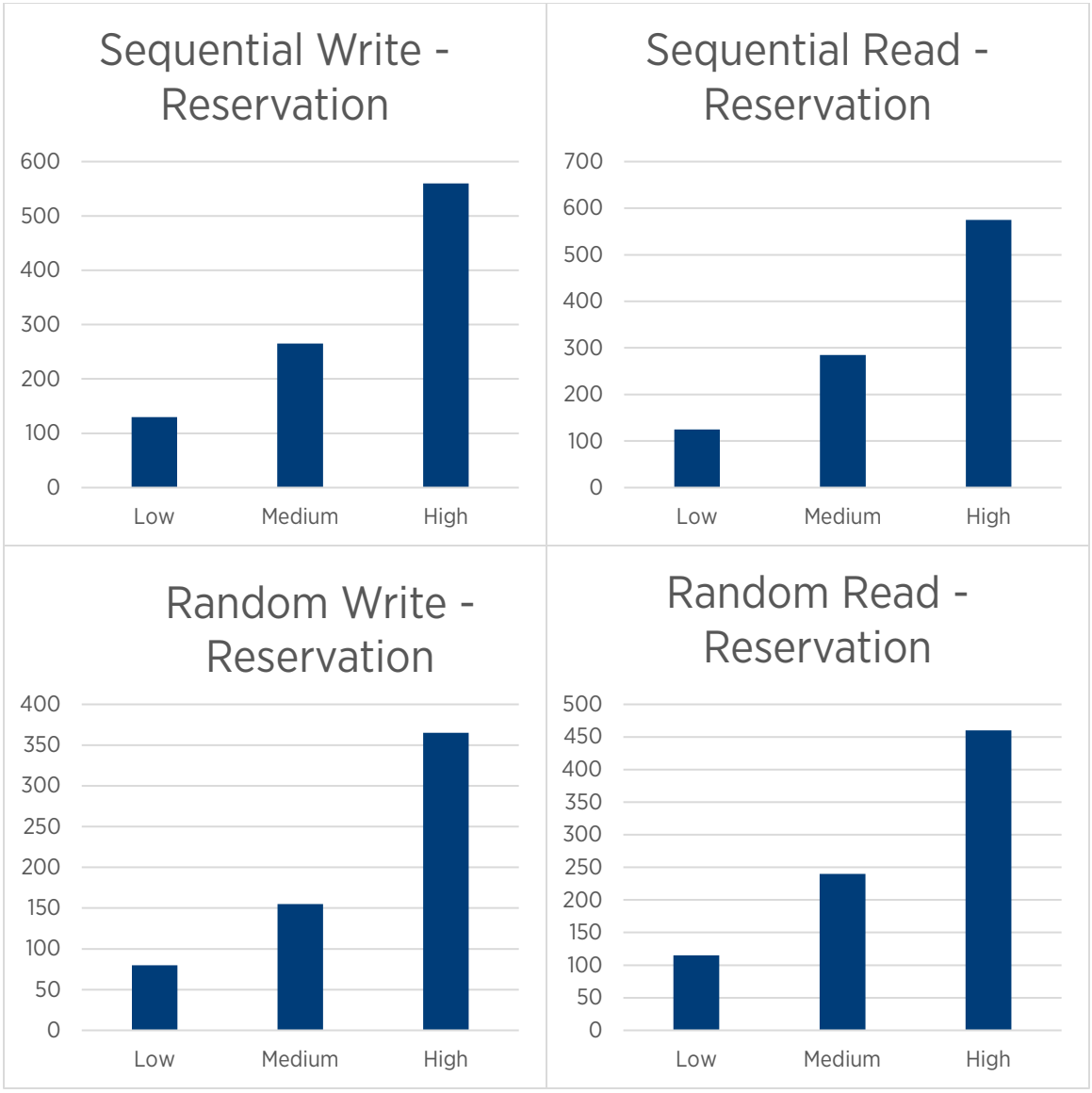


Figure 5. I/O performance using policies

As we can see in the result above, the performance difference is clearly demonstrated. All the reservation settings, 10, 50, and 100 for low priority, medium priority, and high priority VMs are met. The performance ratio of 4:2:1 specified by share is also guaranteed. The high priority VMs achieve better performance when there are spare IOPS.

Test 3: Performance Impact on Datacenter from Using SIOC

In this experiment, we simulate the environment of a real datastore. We collected data from a long period (40 minutes), and there are three types of applications running.

First, we have the logging machines, which are responsible for collecting and logging system information into the database. It is not an important workload, but it always runs in the background and causes high latency because of its large I/O size. In this experiment, there are two logging machines running for the entire duration of the experiment, which is 40 minutes.

Next, we have analysis workloads that are run by the staff working in the datacenter. This workload simulates research and development, so it has a higher priority than the logging machines. It runs occasionally and keeps running for a limited period. In this experiment, the analysis workload starts at 10 minutes after the logging VMs start and lasts 30 minutes. There are four VMs used for analysis.

Finally, we have two production virtual machines that are used to simulate end-user application requests. It is important to deliver time-sensitive service to end-users, so these VMs have high priority. A user request usually does not take a long time, so it runs for 10 minutes—starting at 20 minutes after the logging VMs start and ending at 30 minutes. All the configurations are summarized in Table 9.

Type of VM	Priority	Start Time (min)	End Time (min)	# of VMs
Logging	Low	0	40	2
Analysis	Medium	10	40	4
Production	High	20	30	2

Table 9. Datacenter test configuration

The results are reported in Figure 6. As we can see, from 0-10 minutes, there are only low priority VMs. Their IOPS are always limited by the SIOC settings, so the system workload remains at a low level. In the period to 10–20 minutes and 30–40 minutes, medium priority VMs come into play. This workload fills all the spare IOPS.

One interesting thing is that low priority applications give up IOPS when their quota is used up at each second. Therefore, higher priority VMs achive more IOPS at the end of each second, and this causes the spikes in performance numbers.

During the period of 20–30 minutes, high priority VMs are introduced and, hence, the medium priority VMs must give up part of their IOPS to the high performance VMs.

We tested all four I/O patterns, and the system acts as expected.

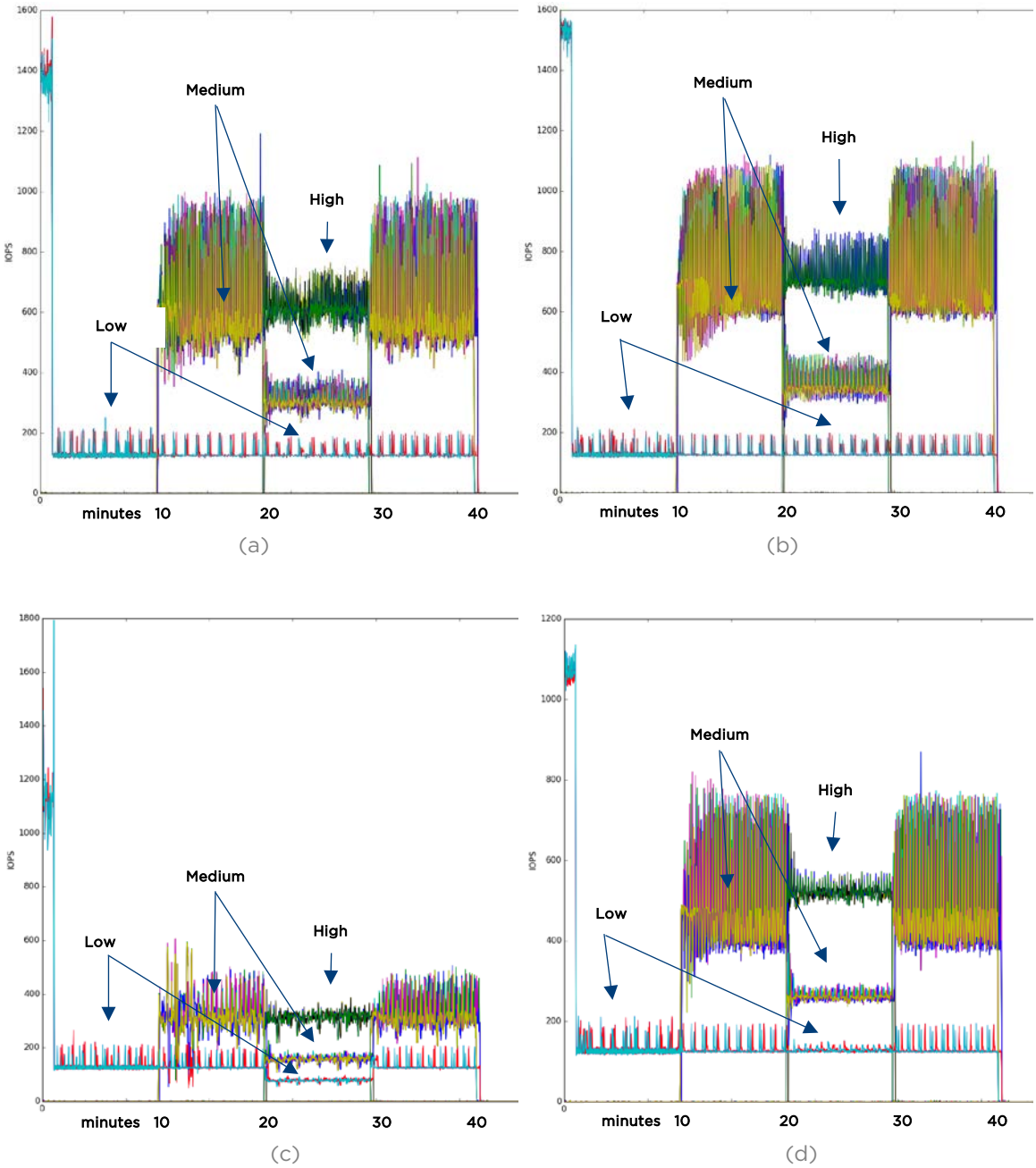


Figure 6. Performance impact of using SIOC in datacenter. (a) sequential write, (b) sequential read, (c) random write and (d) random read

Test 4: Performance Impact on Uneven Configuration Using SIOC

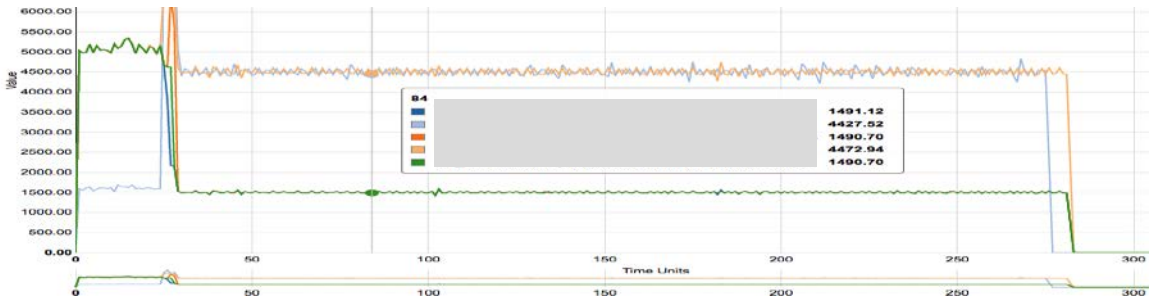
SIOC should guarantee that the performance of a virtual device, when using the same datastore, should be controlled only by its own policy, even if the computational environment is very different between hosts. To prove this, we run I/O tests using an uneven configuration.

On host one, there are twelve VMs running at the same time, competing for the IOPS of the storage device. On the second host, there are only four VMs using the same datastore. The first VM of each host are high priority VMs, and the remaining VMs are low priority VMs. The policy for high priority and low priority are the same between hosts.

The configuration is summarized in Table 10.

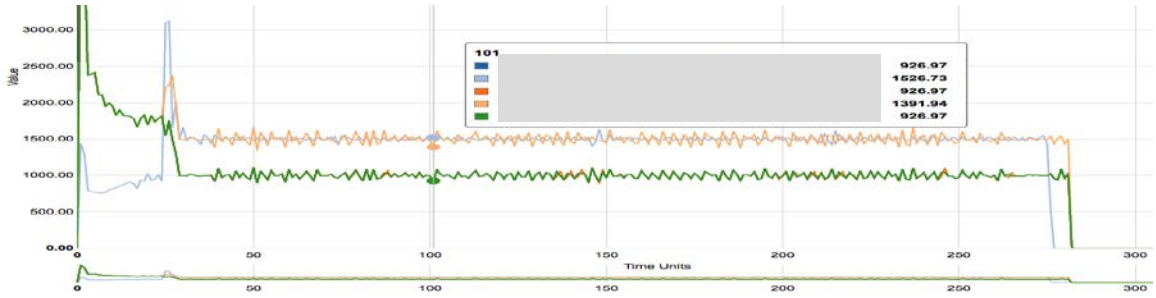
	Total VMs	High Priority	Low Priority
Host 1	12	1	11
Host 2	4	1	3

Table 10. Configuration for uneven tests



(a)

(b)



(c)

(d)

Figure 7. Performance impact of using SIOC on Uneven Settings: (a) sequential write, (b), sequential read, (c) radom write, (d) and random read (host names are hidden)

The result is shown in Figure 7, where we can see that the performance gaps between VMs of the same configuration (high priority or low priority) is very small, even though the computational environment is different between the VMs. Therefore, SIOC meets our expectation for managing the datastore performance based purely on policies.

Conclusion and Future Work

We tested each of the three features in SIOC separately, and then we combined all the features to have an overall evaluation. We used SIOC in a simulated production environment to check its efficiency. We also investigated SIOC's capability of managing datastore performance based on policy, even under an unbalanced topology. From this research, we reached the following conclusions:

- Each of the three features: share, limit, and reservation can effectively describe the priority of an application and schedule IOPS accordingly. SIOC smartly distributed the IOPS to make full use of the hardware potential.
- Policies, created from combined SIOC features, provide an efficient way to manage datastore performance. We used the default policies in all our tests and found that the performance resources were distributed correctly.
- In the datacenter environment simulated at the VMware performance lab, we found that SIOC can effectively manage a datacenter by treating logging applications, analysis applications, and production applications in different manners. SIOC not only distributed resources per policy, but also reached the goal to maximize the overall system performance.
- SIOC demonstrated its ability in managing VMs in different computational environments. Even under an unbalanced distribution of computational resources, SIOC still guaranteed the performance requirement specified in the policies.

This paper focused on the performance improvements on SSD datastores using VMFS. In the future, we will investigate SIOC performance on both traditional HDD storage and on NFS datastores.

References

[1] Jens Axboe. Flexible I/O Tester. https://github.com/axboe/fio

[2] VMware, Inc. (2017) vSphere Resource Management, Chapter 8: Managing Storage I/O Resources. http://pubs.vmware.com/vsphere-65/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-65-resource-management-guide.pdf

About the Author

Yifan Wang is a performance engineer in the I/O Performance team at VMware. Yifan is responsible for evaluating and improving the performance of VMware storage products, like Virtual Volumes, Virutal SAN, and SIOC. He also has an interest in integrating machine learning ideas into performance debugging and evaluation.

Acknowledgements

I appreciate the assistance and feedback from many fellow VMware colleagues including, but not limited to Ruijin Zhou, Komal Desai, and Ravi Cherukupalli. I am also grateful to the support from the management team including Rajesh Somasundaran, Bruce Herndon, and Chuck Lintell. Lastly, the feedback and support from the I/O Performance team always plays an important role throughout the project.

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