VMware vSphere Snapshots: Performance and Best Practices

Performance Study - November 8, 2021

Updated to include a new section on VM snapshots in vSphere/Kubernetes environments.

vmware®

Table of Contents

Introduction

The virtual machine snapshot is a commonly used feature in the VMware vSphere® environment.

One of the most common questions we get is, “What is the performance impact of VM snapshot creation/deletion on the performance of guest applications running inside the VMs?”

In this technical paper, we discuss VM snapshot operation and its impact on the performance of guest applications and how it affects other VM provisioning operations. Our study explores these performance aspects in VMFS, vSAN, and vVOL environments with a variety of workloads and provides recommendations. In addition, we look at VM snapshot performance in a vSphere/Kubernetes environment.

What is a Snapshot?

A snapshot preserves the state and data of a VM at a specific point in time.

- The state includes the VM’s power state (for example, powered on, powered off, suspended).
- The data includes all the files that make up the VM. This includes disks, memory, and other devices, such as virtual network interface cards.

A VM provides several operations for creating and managing snapshots and snapshot chains. These operations let you create snapshots, revert to any snapshot in the chain, and remove snapshots. You can create extensive snapshot trees.

For more information on vSphere snapshots, refer to the VMware documentation “Using Snapshots to Manage Virtual Machines.”

Snapshot Formats

When you take a snapshot of a VM, the state of the virtual disk is preserved, the guest stops writing to it, and a delta or child disk is created. The snapshot format chosen for the delta disk depends on many factors, including the underlying datastore and VMDK characteristics, and has a profound impact on performance.

SEsparse

SEsparse is the default format for all delta disks on VMFS6 datastores. SEsparse is a format similar to VMFSsparse (also referred to as the redo-log format) with some enhancements.

vSANSparse

vSANSparse, introduced in vSAN 6.0, is a new snapshot format that uses in-memory metadata cache and a more efficient sparse filesystem layout; it can operate at much closer base disk performance levels compared to VMFSsparse or SEsparse.

vVols/native snapshots

In a VMware virtual volumes (vVols) environment, data services such as snapshot and clone operations are offloaded to the storage array. With vVols, storage vendors use native snapshot facilities, hence vSphere snapshots can operate at a near base disk performance level.

In this technical paper, we discuss the VM snapshot performance when using different datastores supported in the vSphere environment.

Testbed Details

vSphere version: ESXi 7.0 U2

Snapshot configurations:

1. Snapshot destination
   a. Create snapshots in the same datastore as the parent
2. Snapshot disk state
   b. Dependent disk
3. Snapshot chain length → 1 – 12
4. Memory flag set to true
5. Quiesce flag set to false

Configuration for VMFS snapshot test:

- **Host:** Dell PowerEdge R940
- **CPU:** Intel Xeon Gold 5120 CPU @ 2.20GHz (56 cores)
- **Memory:** 1TB RAM
- **Local VMFS6:** Dell SATA TLC SSD SSDSC2KG960G7R

Configuration for vSAN test:

- **vSAN version:** 7.0 U2
- **Nodes in cluster:** 4
- **Hosts:** 4 Dell PowerEdge 740 servers

- **CPU:** Intel Xeon Processors Gold 6148 CPU @ 2.40GHz "Skylake" (40 cores)
- **Memory:** 766GB RAM
- **vSAN datastore:** 2 disk groups per host - NVME cache disk + 2 SATA SSD capacity disks

Configuration for vVOL test:

- **Storage array:** Dell EMC PowerStore (NVME backed)
- **Host:** Dell PowerEdge R740
- **CPU:** Intel Xeon Processors Platinum 8260 CPU @ 2.40GHz "Cascade Lake" (48 cores)
- **Memory:** 766GB RAM

Configuration for Kubernetes test:

- **Kubernetes:** v1.16.3
- **Docker:** 18.06.2-ce
- **Guest OS:** Ubuntu 18.04.1
- **VM config:** 16 vCPUs, 64 GB memory, 7 vmdks/persistent volumes
- **Storage:** VMFS volume over SSD SAN

Workloads

FIO workload

FIO v3.7 is an I/O microbenchmark used to measure file system I/O performance. The performance metric is I/Os per second (IOPS).

Software:

- **VM config:** 4 vCPUs, 32GB memory, 2 vmdks (100GB system disk, 50GB data disk)
- **FIO benchmark parameters (2 sets of performance data with random I/O and sequential I/O):**

```
-ioengine=libaio -iodepth=32 –rw=randrw –bs=4096 -direct=1 -numjobs=4 -group_reporting=1 -
size=50G –time_based -runtime=300 -randrepeat=1

-ioengine=libaio -iodepth=32 –rw=readwrite –bs=4096 -direct=1 -numjobs=4 -group_reporting=1 -
size=50G –time_based -runtime=300 -randrepeat=1
```

JVM workload

We used the industry-standard SPECjbb 2015 benchmark as the client load generator. The performance metric for this workload is average processed requests (PR) over 10 iterations.

> NOTE: Our results are not comparable to SPEC's trademarked metrics for this benchmark.

Software:

- **VM config:** 4 vCPUs, 32GB memory, 1 VMDK (16GB disk)
- **Guest/application:** CentOS / JVM
- **SPECjbb benchmark parameters:**

```
JAVA_OPTS :-Xms30g -Xmx30g -Xmn27g -XX:+UseLargePages -XX:LargePageSizeInBytes=2m -XX:-
UseBiasedLocking -XX:+UseParallelOldGC specjbb.control (type/ir/duration): PRESET:10000:300000
```

OLTP database workload

We used the open source HammerDB benchmark as the client load generator with the TPC-C benchmark workload profile. The performance metric is transactions per minute (TPM).

> NOTE: This is a non-compliant, fair-use implementation of the TPC-C workload for testing the benefits of a feature; our results should not be compared with official results.

Software:

- **VM config:** 12 vCPUs, 32GB memory, 3 VMDKs (100GB system disk, 250GB database disk, 100GB log disk)
- **Guest/application:** RHEL 7.6 / Oracle Database 12.2
- **Benchmark:** HammerDB using a TPC-C database size 1000 warehouses
- **HammerDB client:** 10 users with 500ms think-time, 5 min ramp-up, 5 min steady-state

Weathervane workload (vSphere/Kubernetes)

Weathervane 2.1 emulates a distributed, modern application workload in the vSphere/Kubernetes environment. The Weathervane (WV) application uses a realistic multi-tier web application that includes both stateless (including Tomcat, RabbitMQ and Zookeeper) and stateful (including PostgreSQL and Cassandra) services.

Weathervane benchmark parameters:

```
users:5000 configurationSize:small2 numAppInstances:1 qosPeriodSec:600
```

Redis container workload (vSphere/Kubernetes)

Redis 6.2.6 is an open-source, in-memory, key-value data store. Memtier_benchmark 1.3.0 is used as the load generation tool.

Redis/Memtier benchmark parameters:

```
Redis pod config:
--appendonly:yes

Memtier benchmark parameters:
--threads=10 –test-time=600 --pipeline=5 --ratio=50:50 -c 20 –key-pattern=R:R
```

Performance Results

VM snapshot impact on guest performance

First, we discuss the impact of VM snapshots on the performance of guest applications running inside the VM using popular benchmark workloads.

For each of the workload scenarios, we use the VM performance without any snapshots as the baseline. In our testing, we varied the number of snapshots from 1 to 12. With the addition of each new VM snapshot, we reran the benchmark to capture the new performance numbers.

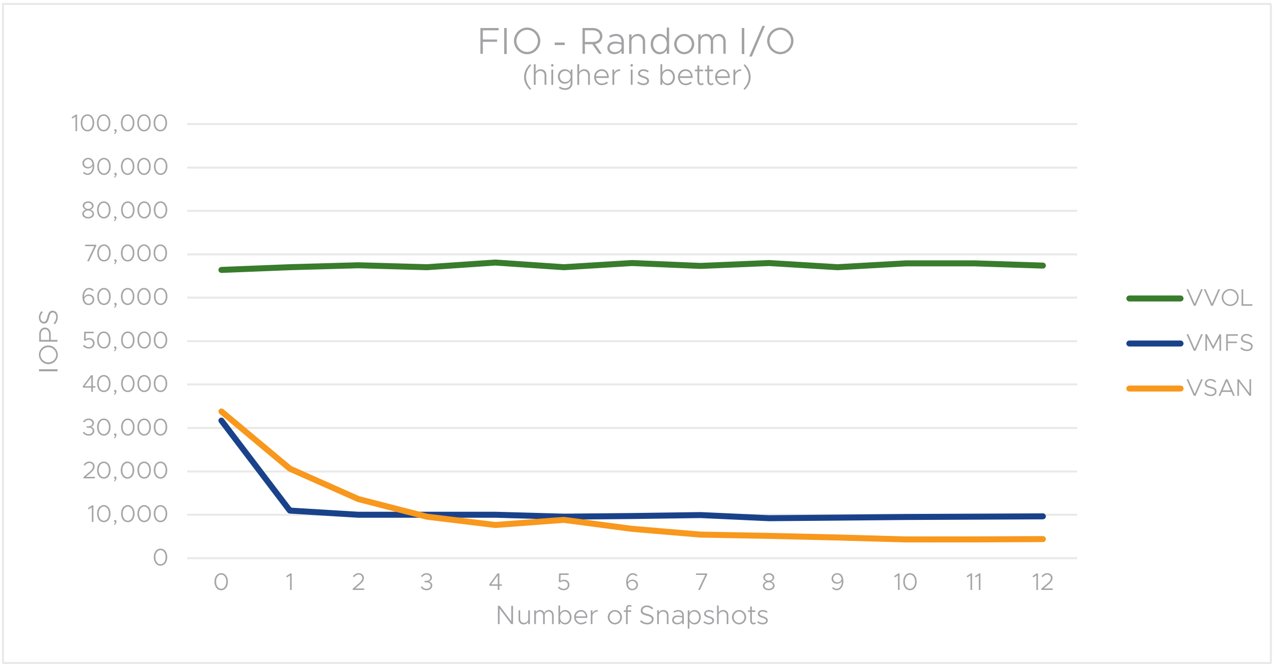


Figure 1. FIO - Random I/O performance with snapshots on VMFS, vSAN, and vVOL

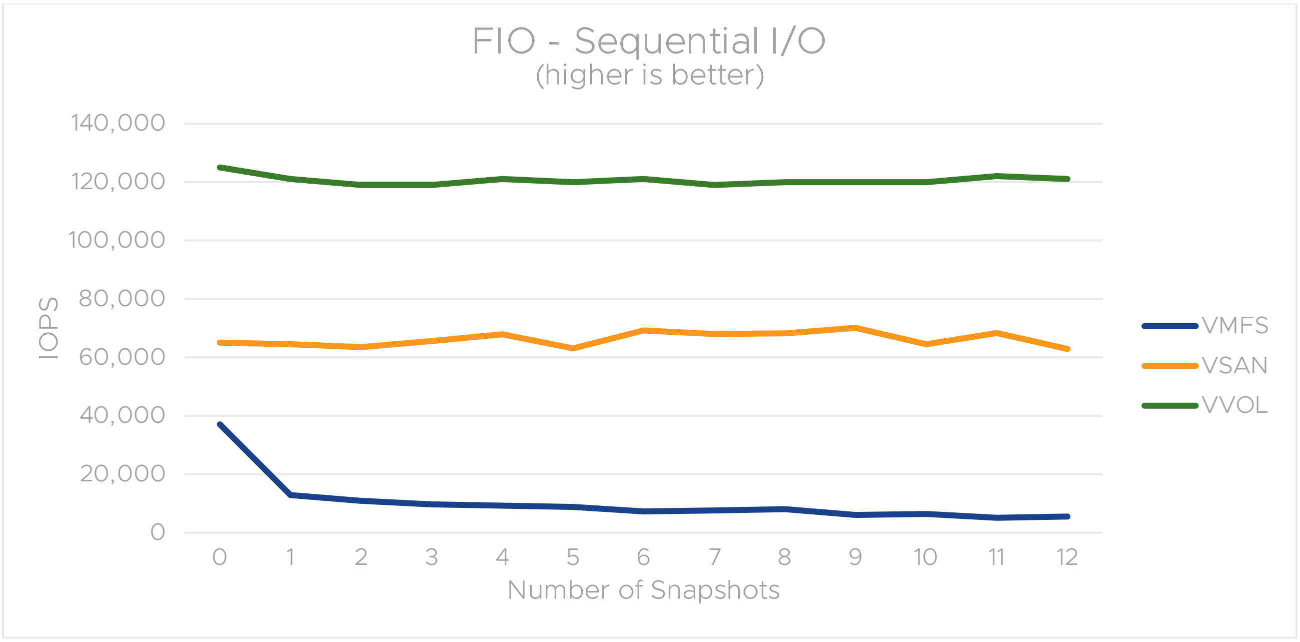


Figure 2. FIO - Sequential I/O performance with snapshots on VMFS, vSAN, and vVOL

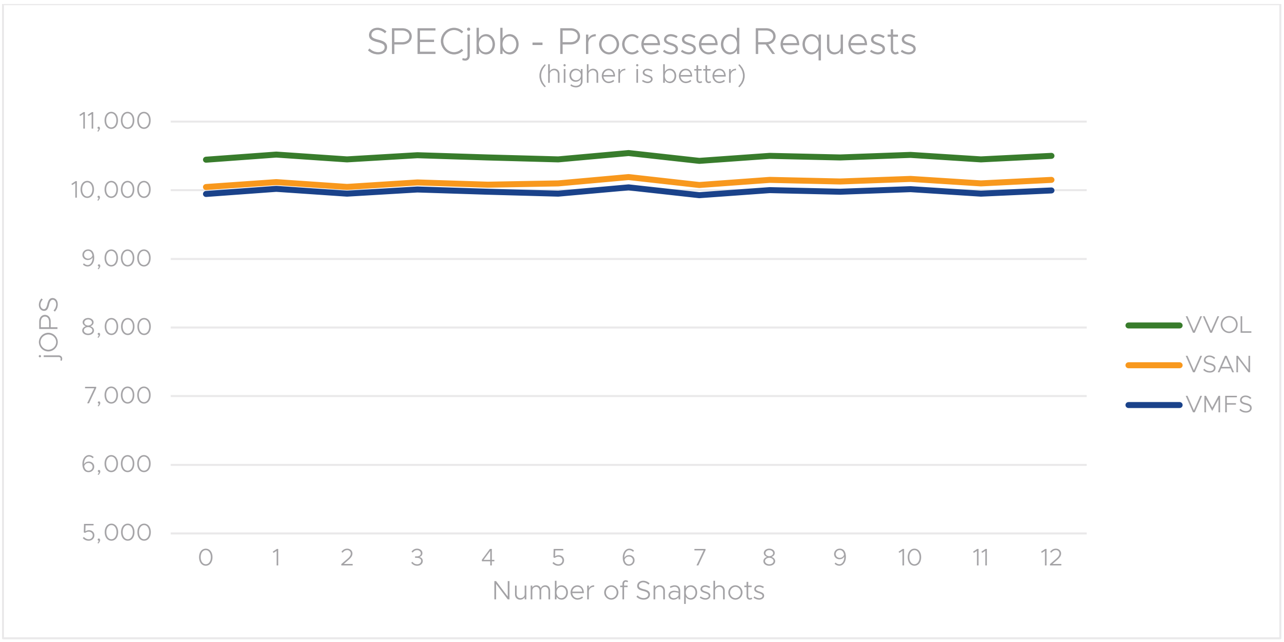


Figure 3. SPECjbb performance with snapshots on VMFS, vSAN, and vVOL

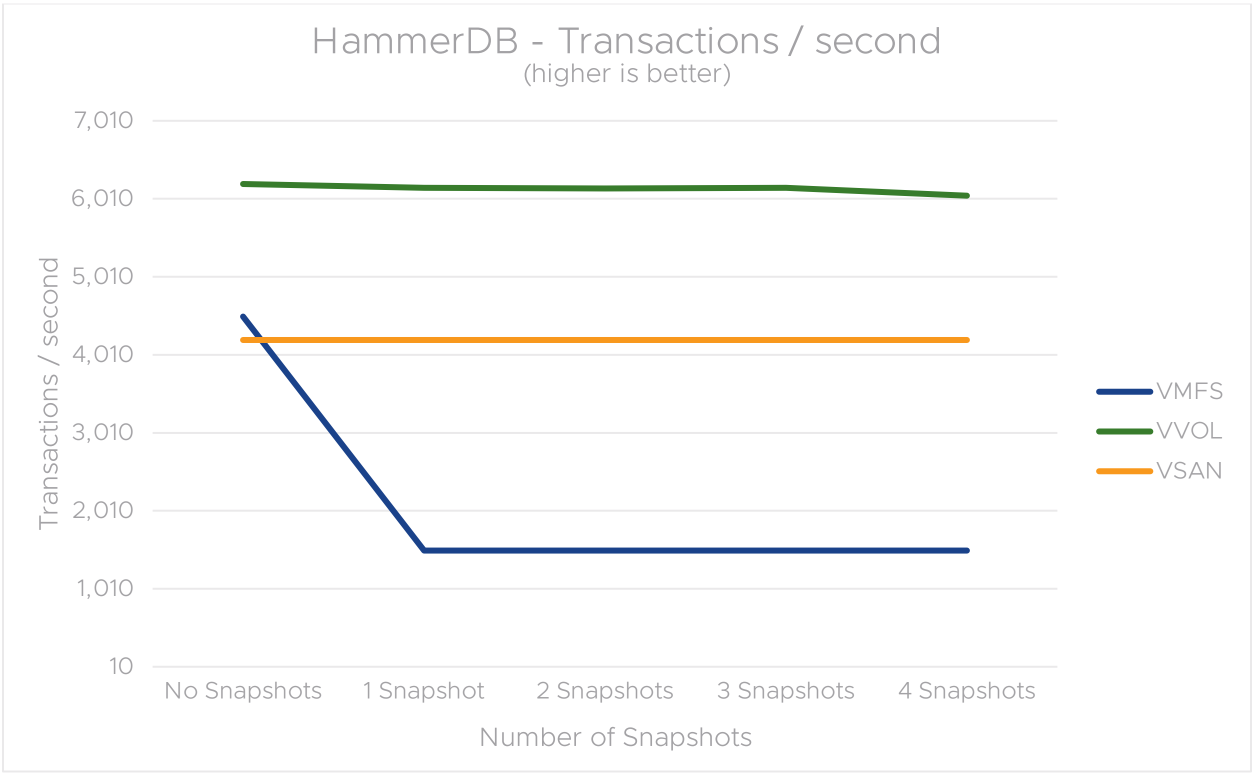


Figure 4. HammerDB performance with snapshots on VMFS, vSAN, and vVOL

The above figures show the impact of VM snapshots on guest application performance on all three datastores. The baseline guest application performance without any snapshots is different on all three datastores; this is expected due to the differences with underlying hardware. The main focus of this study is to understand the impact on the guest application performance with snapshots.

The presence of the VM snapshot has the most impact on the guest application performance when using the VMFS datastore. As can be seen in Figures 1 and 2, FIO performance (random and sequential I/O) on VMFS drops significantly with the first snapshot. Figure 4 shows a similar drop in performance on VMFS with the HammerDB workload as well. In general, we observe guest applications with a significant disk I/O component losing nearly 65% throughput on the VMFS datastore in presence of a single VM snapshot.

The guest performance impact in the presence of snapshots on VMFS is due to the nature of the SEsparse re-do logs. When I/O is issued from a VM with a snapshot, vSphere determines whether the data resides in the base VMDK (data written prior to a VM snapshot operation) or if it resides in the redo-log (data written after the VM snapshot operation) and the I/O is serviced accordingly. The resulting I/O latency depends on various factors, such as I/O type (read vs. write), whether the data exists in the redo-log or the base VMDK, snapshot level, redo-log size, and type of base VMDK.

In comparison to VMFS, we observe the presence of VM snapshots on a vSAN datastore has minimal impact on guest application performance for workloads that have predominantly sequential I/O. In the case of random I/O tests, similar to the VMFS scenario, we observe substantial impact on guest performance on vSAN.

Of all the three scenarios, the presence of VM snapshots has the least impact on guest performance when using vVOL, thanks to its native snapshot facilities. In fact, our testing showed the impact was nearly zero even as we increased the snapshot chain.

SPECjbb performance remained unaffected in the presence of snapshots (figure 3) on all the test scenarios. This is expected since it does not have any disk I/O component because there isn't any I/O to the VM delta disks.

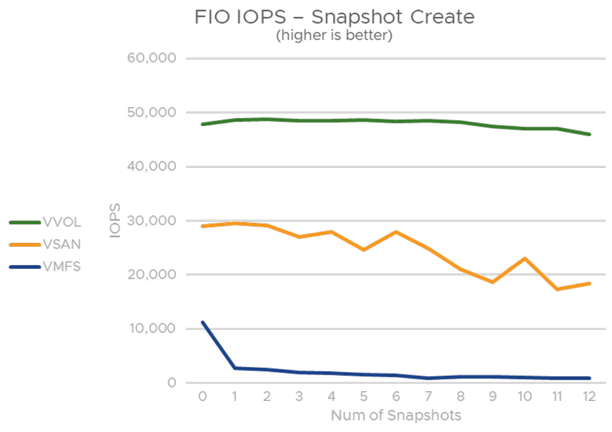
Impact of snapshot removal on guest performance

Deleting a snapshot consolidates the changes between snapshots and writes all the data from the delta disk to the parent snapshot. When you delete the base parent snapshot, all the changes merge with the base VM disk.

To understand the impact of VM snapshot consolidation/removal, we considered a FIO test scenario that included 16KB I/O sizes with a 50:50 mix of sequential and random I/Os.

```
fio -ioengine=libaio -iodepth=32 –rw=randrw --percentage_random=50,50 --bs=16384 -direct=1 -
numjobs=4 -group_reporting=1 -size=50G –time_based -runtime=300 -randrepeat=1
```

As in the previous scenario, we increased the number of snapshots from 1 to 12 and captured guest application performance after the addition of each new VM snapshot. We then reversed the workflow by deleting the VM snapshots and captured the guest application performance after the removal of each VM snapshot.



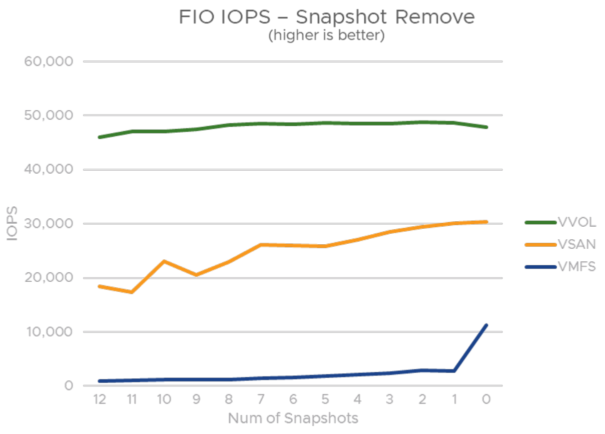


Figure 5. FIO performance with snapshot create

Figure 6. FIO performance with snapshot delete

Figure 5 shows the impact on guest application performance with the addition of each snapshot. As in the previous 100% random I/O scenario (figure 1), we see a significant impact on guest performance when running the 50% random I/O test on both VMFS and vSAN in the presence of snapshots. Once again, the guest performance on vVOL remains unaffected.

Figure 6 shows we're able to recover the performance with each snapshot deletion on all the datastores. After all of the snapshots are deleted, the guest performance is similar to what it was prior to creating any snapshots. In fact, figure 6 is almost a mirror image of figure 5.

Workloads with large I/O sizes

We extended our test coverage to include larger I/O block size scenarios. There were 100% read and 100% write tests performed for sequential I/Os with 32KB, 64KB, and 512KB block sizes. We used the following commands:

```
READ: -ioengine=libaio -iodepth=32 –rw=read bs=<32k|64k|512k> -direct=1 -numjobs=4 -
group_reporting=1 -size=50G –time_based -runtime=300 -randrepeat=1

WRITE: -ioengine=libaio -iodepth=32 –rw=write bs=<32k|64k|512k> -direct=1 -numjobs=4 -
group_reporting=1 -size=50G –time_based -runtime=300 -randrepeat=1
```

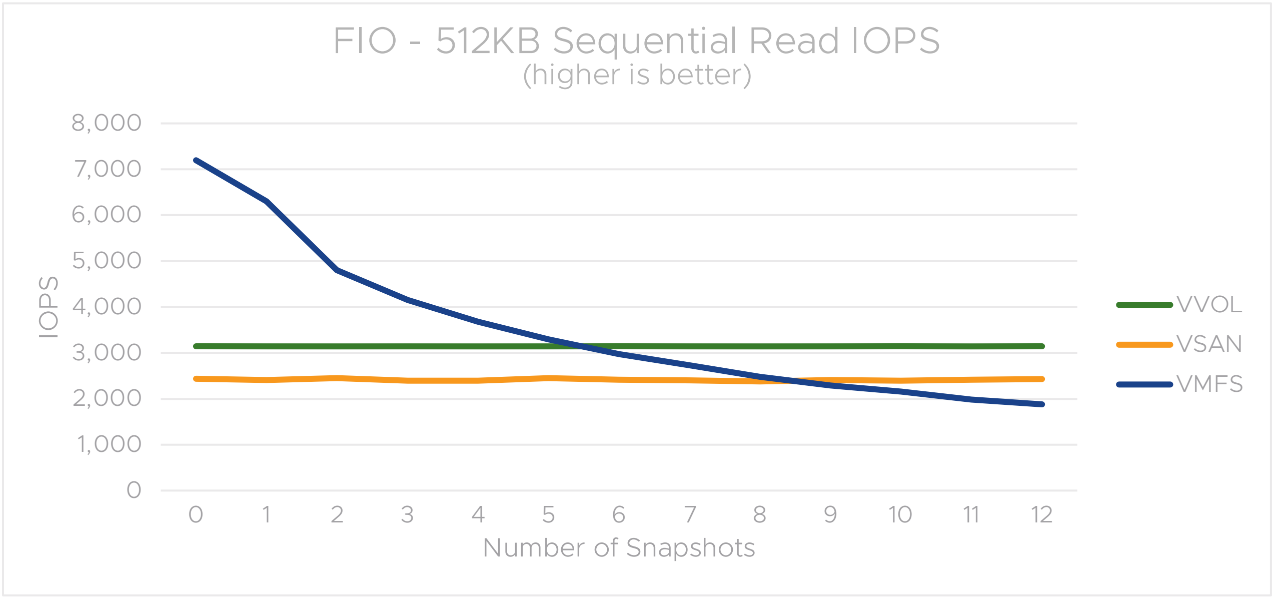


Figure 7. FIO read IOPS (512KB block size) performance

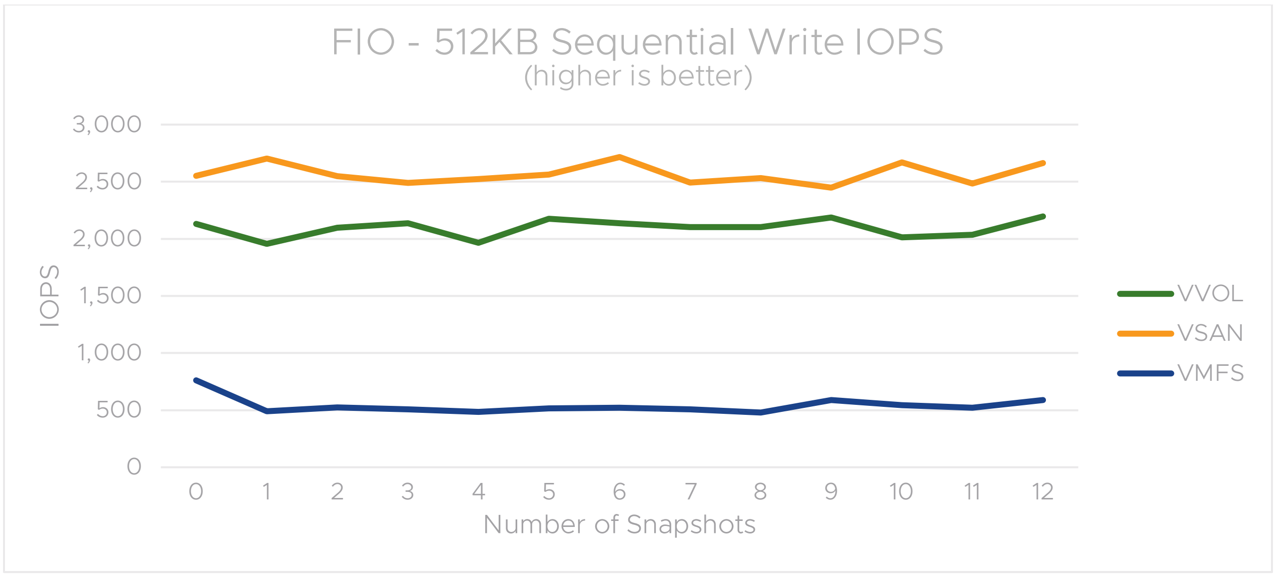


Figure 8. FIO write IOPS (512KB block size) performance

First, we note that the baseline FIO throughput (with no snapshots) when using 512KB I/O sizes is much lower than the throughput observed with the 4KB I/O size in our prior experiments. This is expected since large block sizes generate lower IOPS. We noticed this across all our test environments (VMFS, vSAN, and vVOL). The main focus of this study is to understand the impact on guest application performance with snapshots when using large I/O sizes.

For 512KB I/O, we saw a drop of about 33% in read performance in VMFS with the first two snapshots. There was a drop of about 6% on average with subsequent snapshots after that. The write performance dropped 35% with the first snapshot and this was maintained with subsequent snapshots. There was no drop in performance in either read and write IOPS for vVOL and vSAN in this case. This is similar to what we observed in our earlier experiments with 4k block size sequential I/Os. Experiments with 32KB and 64KB block size yielded similar results.

VM snapshot impact on guest performance

The below table summarizes guest application performance loss in the presence of one snapshot with a variety of workloads on all the three datastores (VMFS, vSAN, and vVOL). For each of the datastore scenarios, the performance loss shown with one snapshot is relative to the baseline performance without any snapshot on the respective datastore.

	SPECjbb	HammerDB	FIO (sequential I/O)	FIO (random I/O)	FIO (large sequential I/O)
VMFS	No impact	65% loss	65% loss	65% loss	70 % loss
vSAN	No impact	No impact	No impact	35% loss	No Impact
vVOL	No impact	No impact	No impact	No impact	No Impact

Table 1. Guest performance loss with 1 snapshot

Impact on VM clone performance

We also evaluated the impact of VM snapshots on provisioning operations such as VM clone. Specifically, we evaluated how the snapshot chain length of the parent VM affects the clone duration. VM cloning of the parent VM without any snapshots was used as the baseline.

Workflow	1. Create a snapshot of the parent VM 2. Run FIO workload inside the parent VM 3. Create a full clone (child VM) from the parent VM and measure the duration of the clone operation 4. Delete the cloned VM 5. Repeat the above steps 1, 2, 3, 4 in a loop

Table 2. Workflow to study the impact of cloning on snapshot performance

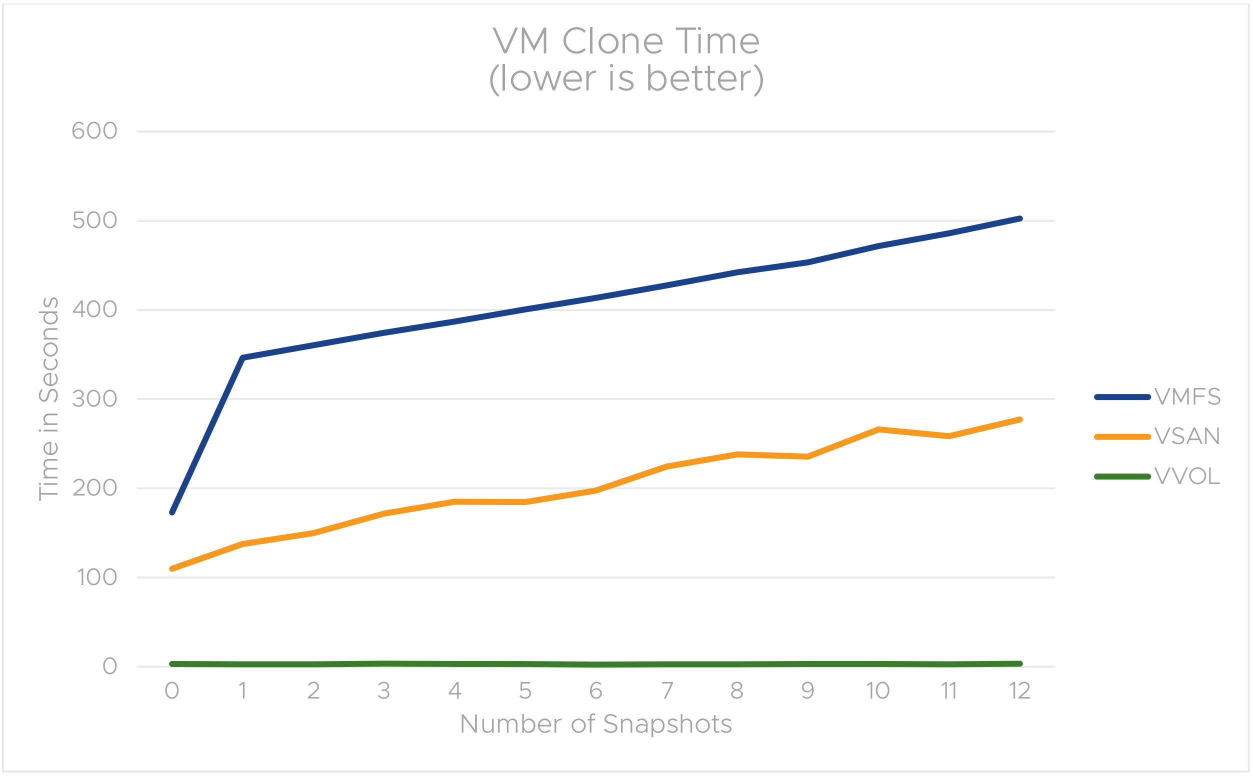


Figure 9. Comparison of VM clone operation with snapshots on VMFS, vSAN, and vVOL

On VMFS, the VM clone time increased by 100% with the first snapshot and increased by an average of 4% with incremental snapshots. We see a similar trend with vSAN too, with a 25% increase in VM clone time with the first snapshot and an increase of 6% on average with incremental snapshots. In comparison, there was no impact on VM clone duration when snapshots were created on the vVOL datastore.

VM snapshot performance in vSphere/Kubernetes environment

Container applications have been a tremendous success over the last several years and are gaining rapid adoption within the vSphere customer base. This study investigates the impact of VM snapshots in a vSphere/Kubernetes environment.

System under test

The system under test (SUT) in our snapshot performance testing here used a Kubernetes (K8s) cluster that consisted of a single K8s worker-node VM. The worker node ran all the Pods that belonged to the Weathervane auction application, as well as all the Redis Pods, thereby resulting in a heavy stress on all the VM system resources including CPU, memory, and disk. The client/load generator ran on a different physical host separate from the SUT.

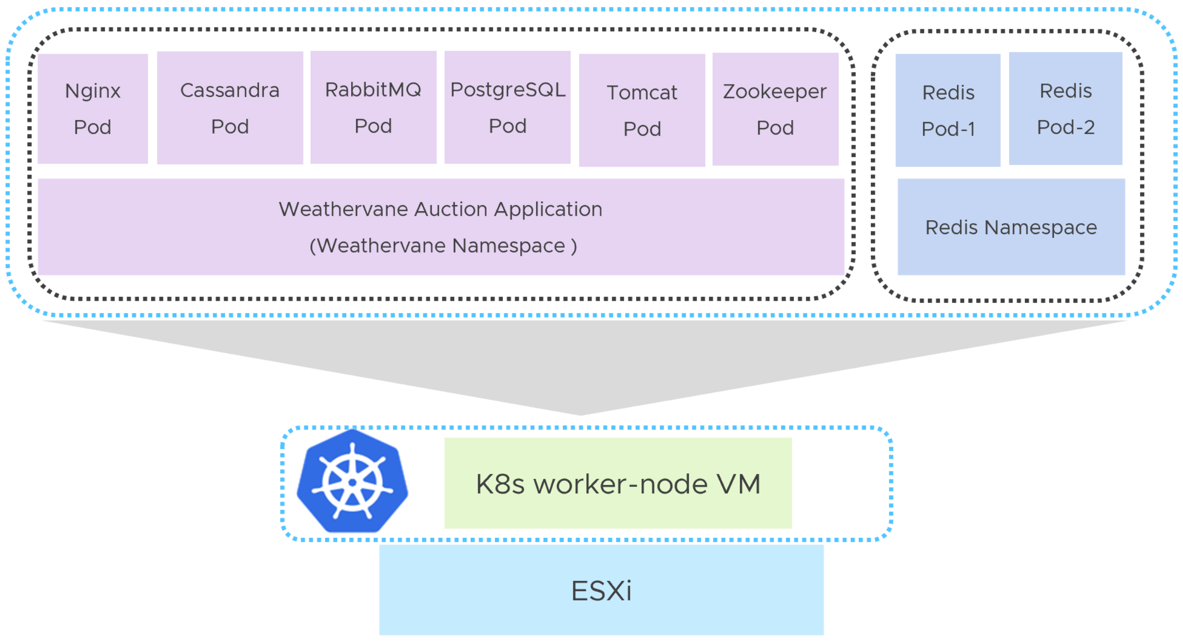


Figure 10. vSphere/K8s configuration used for VM snapshot performance testing

Performance testing workflow

We used the K8s worker-node VM performance without any snapshots as the baseline. With the addition of each new VM snapshot, we reran the benchmarks to capture the new performance numbers.

Workflow	1. Capture the baseline performance of K8s worker-node VM without any snapshots 2. Create a snapshot of the K8s worker-node VM 3. Run the benchmarks to capture new performance data 4. Repeat the above steps 2, 3, in a loop

Table 3. Workflow to study the impact of VM snapshots in vSphere/K8s environment

Results

The table below shows the usage of VM system resources of the K8s worker node during the steady state interval of the benchmarks in our baseline testing.

Config	CPU usage	Memory usage	Disk I/O	Disk I/O
0-snapshots (baseline)	712% (esxtop %USED counter)	64GB (esxtop memory GRANT)	150 READS/s, 5 MBREAD/s	220 WRITES/s, 10 MBWRTN/s

Table 4. Usage of K8s worker-node VM system resources during baseline testing

The table below shows the results of both the benchmarks in all our test scenarios in which we varied the number of VM snapshots from 0 (baseline) to 6.

Configuration	Redis Throughput	Weathervane Results
0 snapshots (baseline)	250K Ops/sec	Throughput:2195.90 (run:pass)
1 snapshot	264K Ops/sec	Throughput:2195.87 (run:pass)
2 snapshots	252K Ops/sec	Throughput:2187.03 (run:pass)
3 snapshots	250K Ops/sec	Throughput:2189.00 (run:pass)
4 snapshots	249K Ops/sec	Throughput:2185.00 (run:pass)
5 snapshots	246K Ops/sec	Throughput:2196.00 (run:pass)
6 snapshots	249K Ops/sec	Throughput:2195.21 (run:pass)

Table 5. Performance results with VM snapshots in vSphere/K8s environment

We see minimal impact on the guest performance in presence of VM snapshots in the vSphere/K8s environment. The results are not surprising since creating a snapshot of a VM (without the Kubernetes stack) is not much different than creating a snapshot of a K8s node VM. This is because both scenarios use VMDKs as the physical storage and the same snapshot technology.

In the vSphere/Kubernetes environment, the containerized applications running inside the Pods use persistent volumes for the persistent storage. Persistent volumes are essentially standard VMDKs used by VMs in a traditional vSphere environment.

Summary

Guest performance

- Guest application performance suffers greatly with I/O-based applications (for example: FIO and HammerDB) in the presence of snapshots, with as much as 85% reduction in guest IOPS when using a VMFS datastore.
- CPU/memory-heavy workloads (for example: SPECjbb) with no disk I/O component remain unaffected in the presence of snapshots.
- There is a minimal impact on guest performance when snapshots are created on a vVOL datastore.
- Although sequential I/O-based workloads have minimal impact, random I/O-based workloads suffer greatly in vSAN environment in the presence of snapshots.
- Guest applications recover their full performance when all the snapshots are deleted.

VM provisioning operations

There is no impact on VM clone duration when snapshots are created on the vVOL datastore.

vSphere/Kubernetes performance

The findings in the vSphere/Kubernetes section, along with prior performance data, confirm that the presence of VM snapshots has very minimal impact in both vSphere and vSphere/Kubernetes environments when running CPU and memory-heavy workloads with a sequential disk I/O profile or workloads with a minimal disk I/O component.

Recommendations

We suggest the following recommendations to get the best performance when using snapshots:

- The presence of snapshots can have a significant impact on guest performance, especially in a VMFS environment. Hence, the use of snapshots only on a temporary basis is strongly recommended and our findings show that the guest fully recovers the performance after deleting the snapshots.
- Using storage arrays like vVOL, which use native snapshot technology, is highly recommended to avoid any guest impact in the presence of snapshots. This is especially important when using I/O-intensive applications within the guest.
- If using vVOL is not an option, using vSAN over VMFS is recommended when using snapshots.
- Our findings show that performance degradation is higher as the snapshot chain length increases. Keeping the snapshot chain length smaller whenever possible is recommended to minimize the guest performance impact.

About the Author

Sreekanth Setty is a staff member with the performance engineering team at VMware. His work focuses on investigating and improving the performance of VMware's virtualization stack, most recently in the live-migration and clone space. He has published his findings in many technical papers and has presented them at various technical conferences. He has a master's degree in computer science from the University of Texas, Austin.

Tanmay Nag is a staff member with performance engineering team at VMware. His work focuses on investigating and tracking different aspects of vCenter and ESXi performance. Part of his work also involves automating tools and benchmarks for performance debugging and measurement. He's been part of VMware family for the last 14 years.

Acknowledgements

The authors extend sincere thanks to the members in their team, including Tony Lin and Julie Brodeur for their reviews.