PERFORMANCE STUDY – MARCH 2017

VMWARE vSPHERE VIRTUAL MACHINE ENCRYPTION PERFORMANCE

VMware vSphere 6.5

Table of Contents

Executive Summary

VMware vSphere® virtual machine encryption (VM encryption) is a feature introduced in vSphere 6.5 to enable the encryption of virtual machines. VM encryption provides security to VMDK data by encrypting I/Os from a virtual machine (which has the VM encryption feature enabled) before it gets stored in the VMDK. In this paper, we quantify the impact of using VM encryption on a VM's I/O performance as well as on some of the VM provisioning operations like VM clone, power-on, and snapshot creation. We show that while VM encryption can lead to bottlenecks in I/O throughput and latency for ultra-high-performance devices (like a high-end NVMe drive) that can support hundreds of thousands of IOPS, for most regular types of storage, like enterprise class SSD or VMware vSAN™, the impact on I/O performance is very minimal.

Introduction

VM encryption supports the encryption of virtual machine files, virtual disk files, and core dump files. Some of the files associated with a virtual machine like log files, VM configuration files, and virtual disk descriptor files are not encrypted. This is because they mostly contain non-sensitive data and operations like disk management should be supported whether or not the underlying disk files are secured. VM encryption uses vSphere APIs for I/O filtering (VAIO), henceforth referred to as IOFilter. IOFilter is an ESXi framework that allows the interception of VM I/Os in the virtual SCSI emulation (VSCSI) layer. On a high level, the VSCSI layer can be thought of as the layer in ESXi just below the VM and above the VMFS file system. The IOFilter framework enables developers, both VMware and third party vendors, to write filters to implement more services using VM I/Os like encryption, caching, and replication. This framework is implemented *entirely* in user space. This allows the VM I/Os to be isolated cleanly from the core architecture of ESXi, thereby eliminating any potential issues to the core functionality of the hypervisor. In case of any failure, only the VM in question would be affected. There can be multiple filters enabled for a particular VM or a VMDK, and these filters are typically chained in a manner shown below, so that I/Os are processed by each of these filters serially, one after the other, and then finally either passed down to VMFS or completed within one of the filters. This is illustrated in Figure 1.

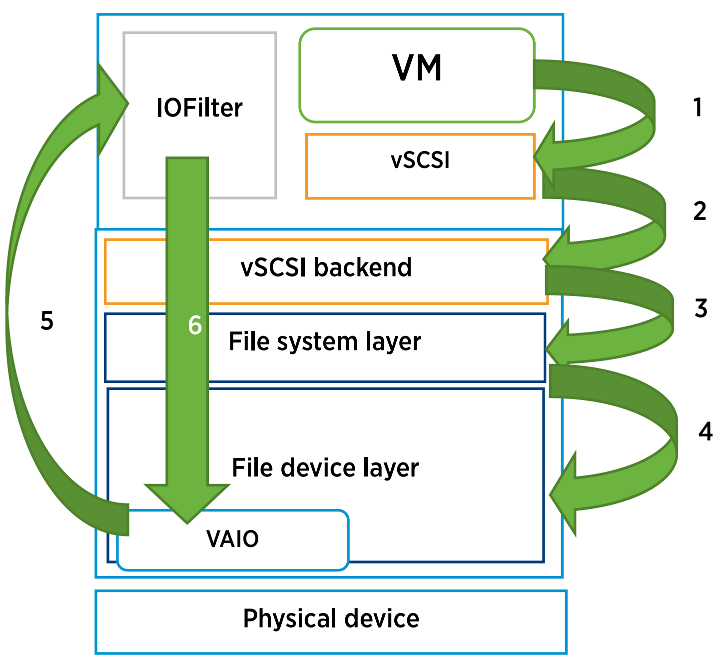

Figure 1. IOFilter design

VM Encryption Overview

The primary purpose of VM encryption is to secure the data in VMDKs, such that when the VMDK data is accessed by any unauthorized entity, it gets only meaningless data. The VM that legitimately owns the VMDK has the necessary key to decrypt the data whenever read and then fed to the guest operating system. This is done using industry-standard encryption algorithms to secure this traffic with minimal overhead.

While VM encryption does not impose any new hardware requirements, using a processor that supports the AES-NI instruction set would speed up the encryption/decryption operation. The results shown in this paper are from a slightly older server with processors that have AES-NI support, but without the speed-up and improvements in AES-NI seen in the latest processors.

Design

Figure 2 shows the various components involved as part of the VM encryption mechanism. It consists of an external key management server (KMS), the vCenter Server system, and an ESXi host or hosts. vCenter Server requests keys from an external KMS, which generates and stores the keys and passes them down to vCenter Server for distribution. An important aspect to note is that there is no “per-block hashing” for the virtual disk. This means, VM encryption provides *data protection against snooping* and not against *data corruption* since there is no hash for detecting corruption and recovering from it. For more security, the encryption takes into account not only the encryption key, but also the block’s address. This means two blocks of a VMDK with the same content encrypt to different data.

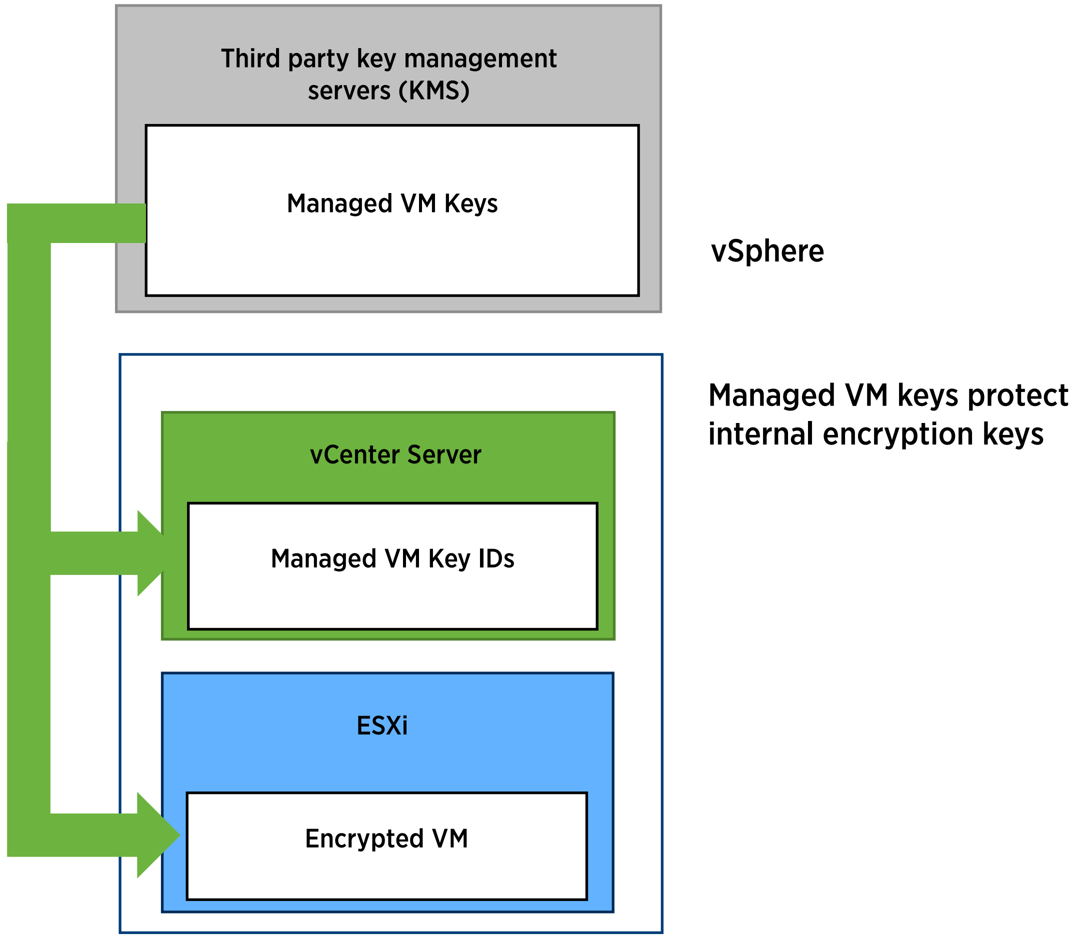

Figure 2. VM Encryption components

Key Management

To visualize the mechanism of encryption (and decryption), we need to look at how the various elements in the security policy are laid out topologically. The KMS is the central server in this security-enabled landscape. Figure 3 shows a simplified topology.

The KMS is a secure, centralized repository of cryptographic keys. There can be more than one KMS configured with a vCenter. However, they need to be configured such that only KMSs that replicate keys between themselves (usually from the same vendor) should be added to the same KMS cluster. Otherwise, each KMS should be added under a different KMS cluster. One of the KMS clusters must be designated as default in vCenter. Only Key Management Interoperability Protocol (KMIP) v1.1 compliant KMSs are supported and vCenter Server is the client of KMS. Using KMIP enables vCenter Server to talk to any KMIP-compliant KMS vendor. Before transacting with the KMS, vCenter Server must establish a trust connection with it, which needs to be done manually.

Initially the ESXi hosts do not have the necessary keys to perform cryptographic operations like encrypting and decrypting guest data. vCenter Server obtains the keys from the KMS and then pushes them down to the hosts. These keys are called key encryption keys (KEK). The host generates the data encryption keys (DEK), which are then used for encrypting and decrypting virtual machine files. KEKs are used to encrypt the DEKs, and these encrypted DEKs are stored in configuration files. Once encrypted, the KEK for the virtual machine needs to be in ESXi memory for the VM to be powered on. If for some reason the ESXi host is power cycled or the encrypted virtual machine is unregistered and then re-registered, vCenter gets the KEK from KMS again and pushes it to ESXi. KEKs are stored only in the KMS where they are generated and not persisted anywhere on vSphere. KMS should be highly available, or keys should be replicated between multiple KMS instances added to the same KMS cluster for accessibility of KEKs.

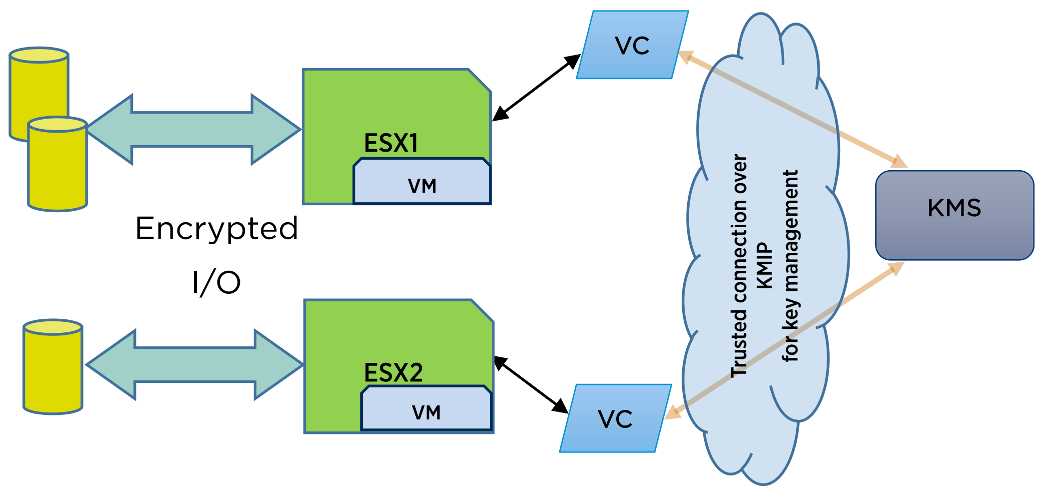

Figure 3. Encryption-enabled vCenter Server (VC) topology

Performance Study

In this paper, we present the results of detailed I/O experiments that quantify CPU cost and I/O throughput and latency when enabling a VM with encryption. We first present the steady-state I/O performance, comparing a VM encryption–enabled case versus a VM encryption–disabled one, and then we present some of the VM provisioning operations like VM power-on, VM clone, and VM snapshot.

Any encryption feature consumes CPU cycles and any I/O filtering mechanism consumes at least minimal I/O latency overhead. The impact of such overheads largely depends on two aspects:

- The efficiency of implementation of the feature/algorithm
- The capability of the underlying storage

If the storage is slow (like a locally attached spinning drive), the overhead caused by I/O filtering is very minimal and has little impact on the overall I/O latency and throughput. However, if the underlying storage is very high-performance, any small overheads added by the filtering layers can have a non-trivial impact on I/O latency and throughput. For this purpose, our performance study covers three sets of results:

- Intel S3700 SSD (capable of about 70,000 random IOPS) [1]
- Samsung PM1725 NVMe device (capable of up to 750,000 IOPS) [2]
- Three-node hybrid VMware vSAN cluster [3]

Experimental Setup

The testbed consists of a single ESXi server in the local storage case and three ESXi servers in the VMware vSAN storage case. The ESXi hosts each contain an IOAnalyzer VM [4], which essentially generates I/O workload using Iometer [5]. vCenter Server manages the hosts and the connection to an external key management server (KMS), with which a trusted connection is established.

Server Hardware

- Dell PowerEdge R720
- 2 x 8-core Intel Xeon Processors E5-2650 v2 (“Ark”) @ 2.60GHz
- 128GB Memory

Server Storage

- 1 VMFS datastore backed by an Intel S3700 400GB SSD device
- 1 VMFS datastore backed by a Samsung PM1725 1.5TB NVMe device

Workload and Virtual Machine Configuration

- Iometer (version 1.1.0) used as synthetic benchmark
- Following I/O profiles used:
  - I/O size of 512KB with sequential workloads (100% reads, 100% writes)
  - I/O size of 4KB with random workloads (100% reads, 100% writes)
- Number of vCPUs per VM equals the number of Iometer workers used, and each worker generates I/O on separate VMDKs. For our experiments, the number of VMDKs (and therefore Iometer workers) is varied from 1 to 8.
- For each virtual disk, there were 8 outstanding I/Os for the 512KB case and 32 outstanding I/Os for the 4KB case
- Each VMDK is 2GB in size
- Experiments conducted *after* the VMDK had been written to completely, and as such did not take ”first write” into consideration
- For VMware vSAN:
  - 1 disk group with Intel S3700 SSD drive + 4 x 1.1TB HGST drives [6]
  - All VMDKs created with HostFailuresToTolerate (HFT) = 1

Metrics

- For larger sequential workloads (512KB), we show throughput in megabytes per second (MBps), whereas for smaller random workloads (4KB), we show I/Os per second (IOPS).
- I/O latencies are shown in microsecond granularity and the CPU cost is shown as the number of CPU cycles per I/O.
- Each test is run for 300 seconds and for at least three iterations.

Server Software Configuration

- Guest operating system version: Ubuntu 12.04 64-bit
- ESXi version: 6.5

Results

I/O Performance

On Intel S3700 SSD Storage

In this section, we present the results of I/O experiments done on a locally attached SAS SSD storage that is of mid-level capability. The SSD drive we used is capable of doing about 75,000 random read IOPS as per the device specification [1]. This capability is similar to what we can expect from an enterprise storage array.

The first set of results are bandwidth-oriented as we focus on large, sequential I/Os. For this purpose, we tested 512KB sequential reads and writes comparing the I/O throughput, I/O latency, and CPU cost per I/O of a regular virtual machine with encryption enabled. Write workloads exploit the encryption workflow and the read workloads exploit the decryption workflow.

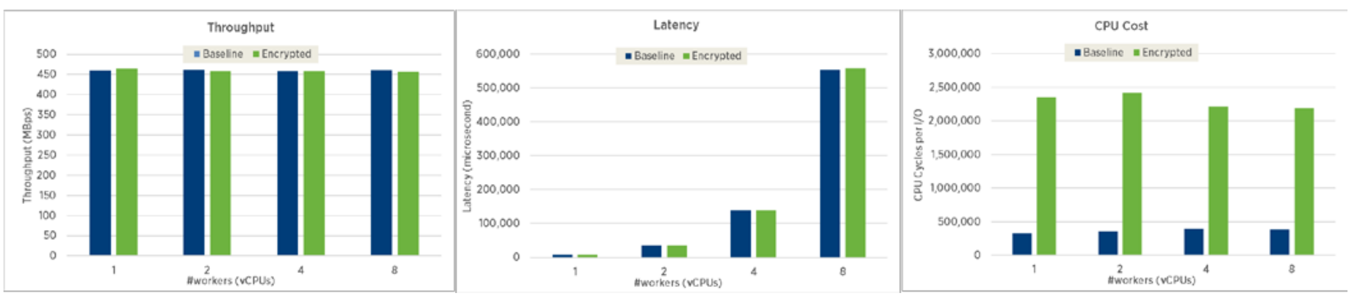

Figure 4. 512KB sequential write results for SSD

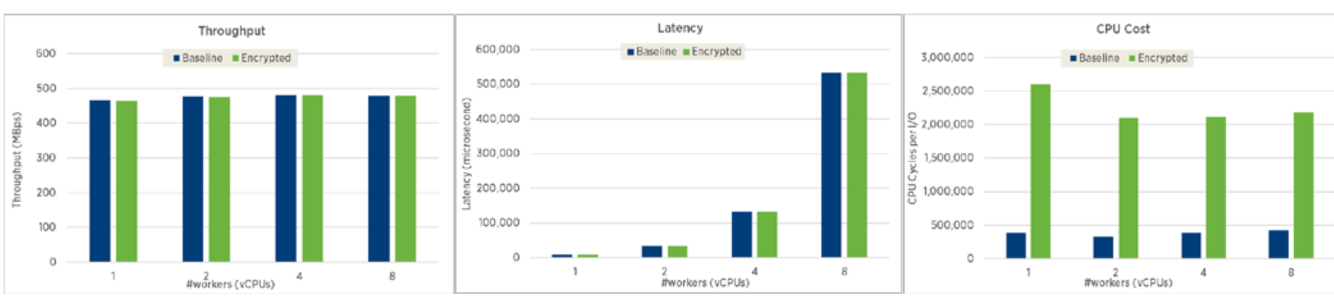

Figure 5. 512KB sequential read results for SSD

Figure 4 and Figure 5 show that in terms of bandwidth, VM encryption does not add any noticeable overhead even when the I/O bandwidth is in the range of 450-475 MBps. Also, when we increase the number of workers issuing I/O, the number of I/Os in-flight that are being processed by ESXi is also increased. With VM encryption enabled, we do not see any noticeable overhead in terms of latency even for a high number of outstanding I/Os.

However, in terms of CPU cost, we see high CPU cycles per I/O when VM encryption is enabled due to the encryption of large quantities of data. When there are spare CPU cycles, this does not have an adverse effect on application performance. But if there is scarcity of CPU resources, VM encryption can add significant overhead to other applications. Therefore, the ESXi server requires a sufficient amount of CPU resources when VM encryption is enabled.

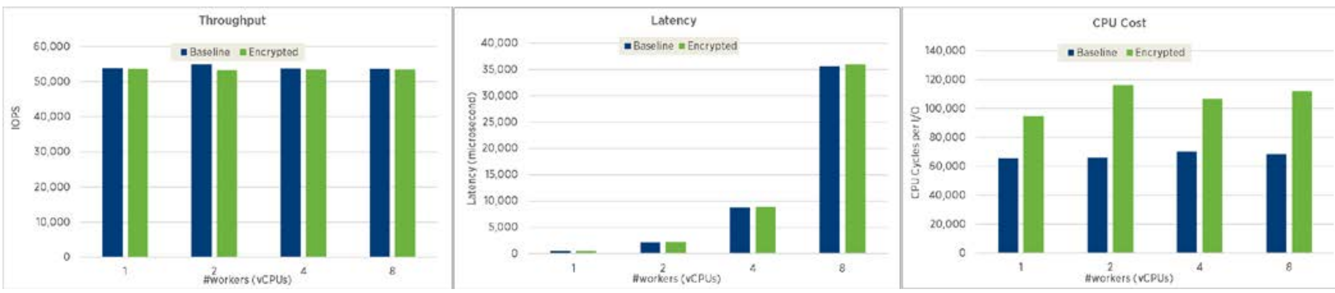

Figure 6. 4KB random write results for SSD

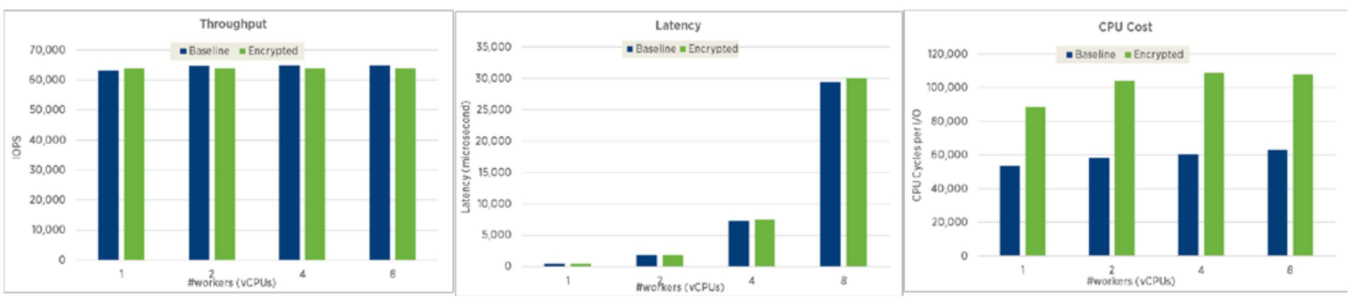

Figure 7. 4KB random read results for SSD

In Figure 6 and Figure 7, we show the impact of VM ecryption on small, random I/Os that are limited by the number of I/Os processed per second. For this case, we tested with 4KB random read and write workloads. Even for these workloads, we see that the VM encryption impact is minimal in terms of throughput and latency. In terms of CPU cycles per I/O, we see that encrypted I/Os consume roughly double the amount of CPU cycles compared to non-encrypted I/Os.

On Samsung PM1725 NVMe Storage

In this section, we present the results of I/O experiments done on an NVMe device that has ultra-low latency performance and high throughput. The device is capable of doing up to 750,000 read IOPS per the device specification [2].

As with the tests run with an Intel S3700 SSD, the first set of results here are bandwidth-oriented and the second set of results are IOPS-oriented.

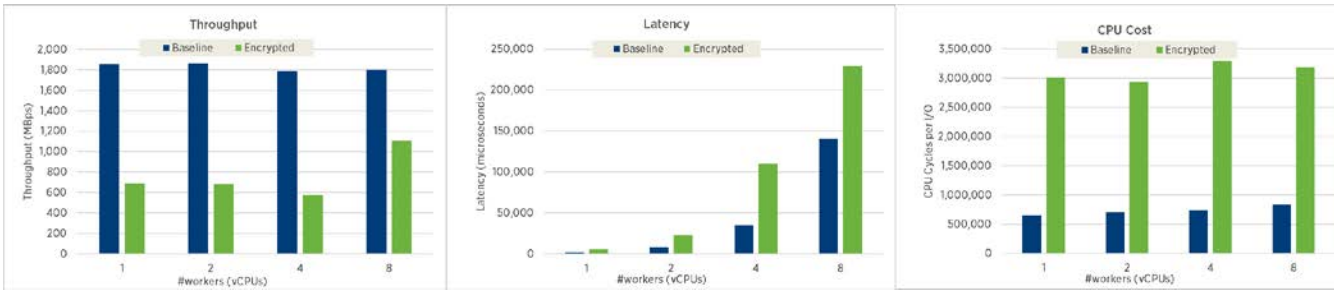

Figure 8. 512KB sequential write results for NVMe

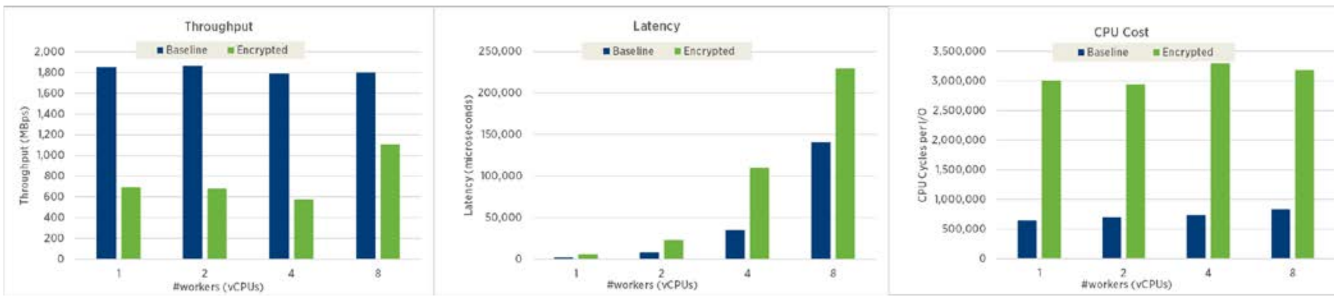

Figure 9. 512KB sequential read results for NVMe

As seen from Figure 8 and Figure 9, with the highly performant NVMe device, we see a significant impact in terms of I/O throughput and latency when using VM encryption. The bandwidth with encryption is about 30-50% of baseline performance. There is also a proportional increase in the I/O latency. Because the device performance is high, the per-I/O latency we add in the IOFilter path for encryption (in the case of writes) and decryption (in the case of reads), which is in the order of a few microseconds, quickly add up and show as a bottleneck in the figures.

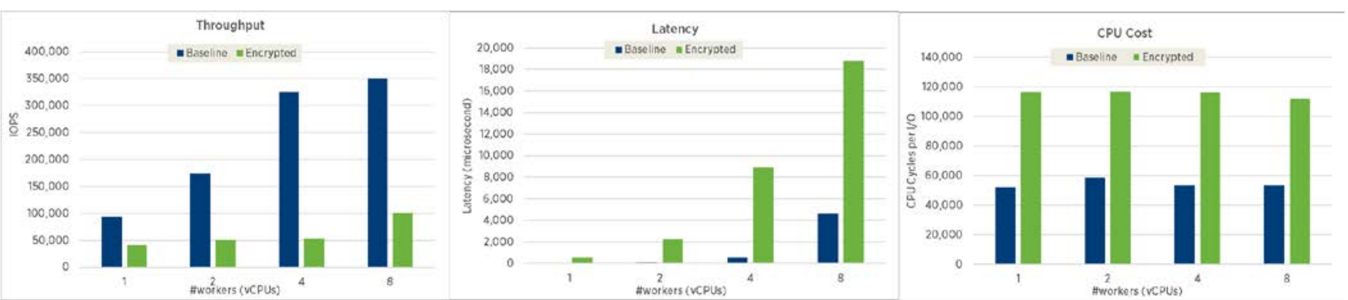

Figure 10. 4KB random write results for NVMe

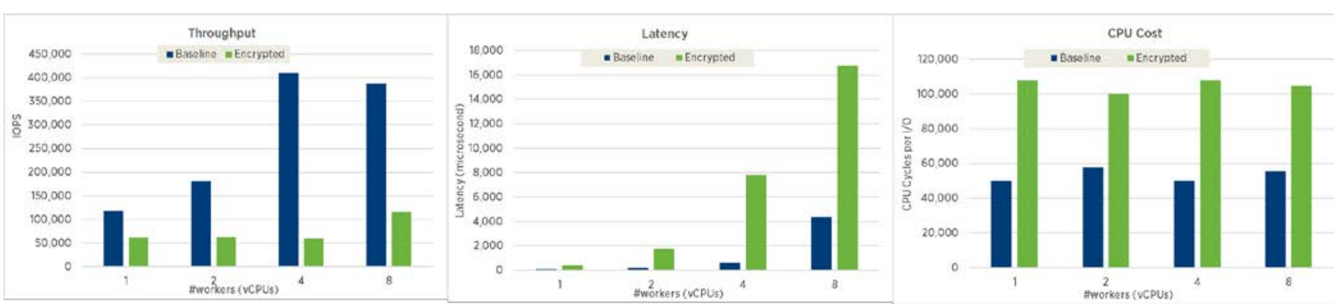

Figure 11. 4KB random read results for NVMe

From Figure 10 and Figure 11, we see a similar pattern as shown in the sequential workload case, where the encryption and decryption of each I/O results in the IOPS not scaling as much as the baseline case.

While we see significant I/O performance impact when using the VM encryption feature on an ultra-fast storage device, this impact may be reduced when using more recent servers that have processors with a faster and improved AES-NI implementation. We plan to publish those results as a separate paper.

On VMware vSAN Storage

In this section, we present the results of I/O experiments done on VMware vSAN. For these experiments, we used a three-node VMware vSAN cluster with a single disk group, where each host comprises 1 Intel S3700 SSD and 4 x 1.1TB Hitachi 10K RPM hard drives. Figure 12, Figure 13, Figure 14, and Figure 15 show the performance comparison of different workloads with and without encryption.

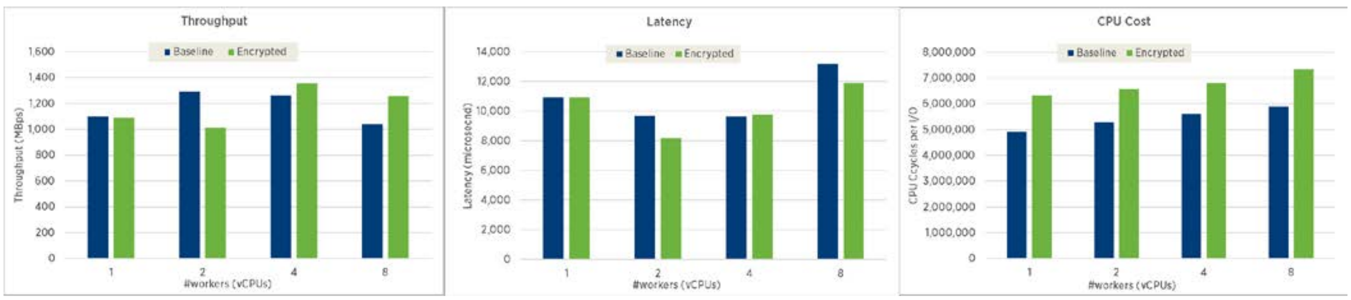

Figure 12. 512KB sequential read results for vSAN

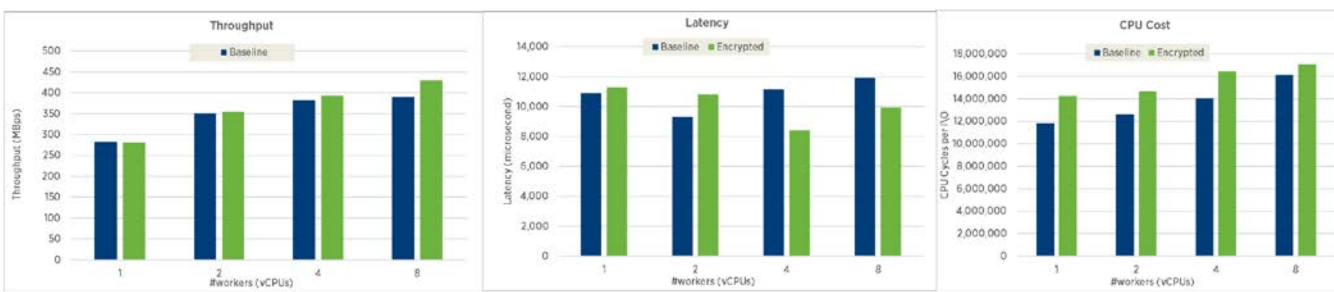

Figure 13. 512KB sequential write results for vSAN

The bandwidth-oriented workloads in Figure 12 and Figure 13 show that there is no noticeable overhead for write workloads in terms of throughput, while latency is slightly affected. In terms of CPU cost per I/O, there is a small increase of less than 20% when VM encryption is enabled.

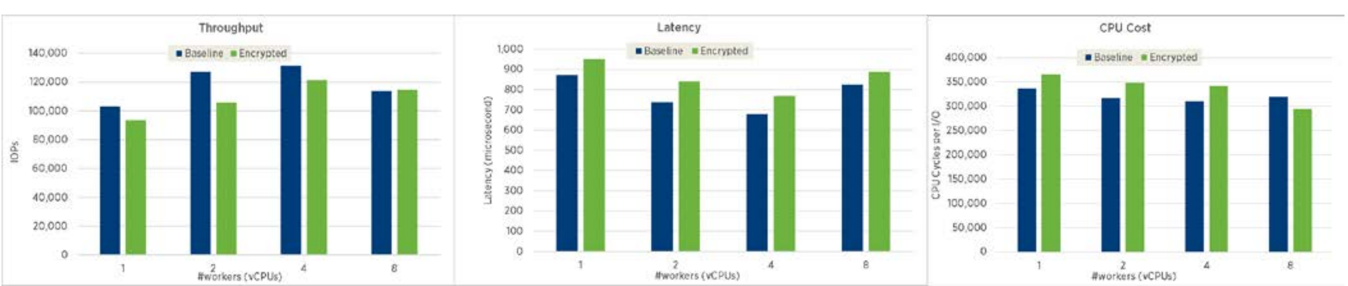

Figure 14. 4KB random read results for vSAN

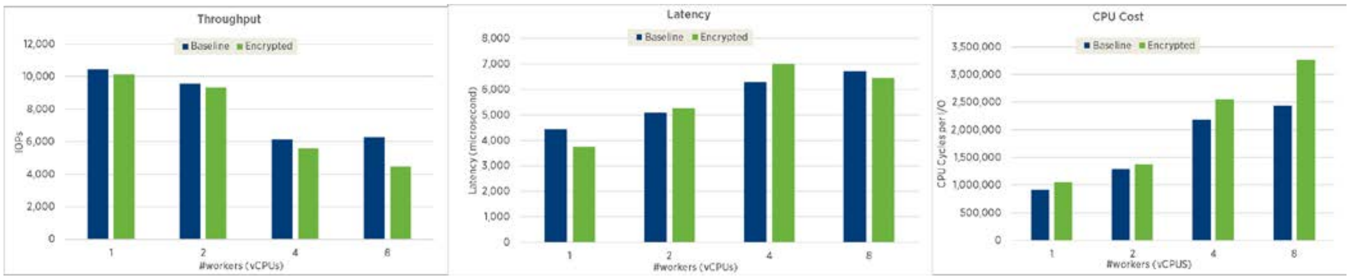

Figure 15. 4KB random write results for vSAN

For random, small-sized workloads, we see VM encryption adding an overhead of up to 20% in the worst case and a similar overhead for CPU cost per I/O. In Figure 15, we found an interesting behavior of write IOPS decreasing as the number of workers increase. Based on the VMware vSAN statistics, we found that this might be due to an increasing number of total outstanding I/Os handled by vSAN as we kept the number of outstanding I/Os per worker to be constant at 32. As the total number of outstanding I/Os increase, congestion occurs in the vSAN layers, bringing down the IOPS.

VM Provisioning Operations

In this section, we focus on three of the important VM provisioning operations: VM power-on, VM clone, and VM snapshot. For these experiments, we use three different classes of underlying storage: an SSD, an NVMe drive, and a three-node vSAN cluster. The testbed setup here is the same as that detailed in Experimental Setup.

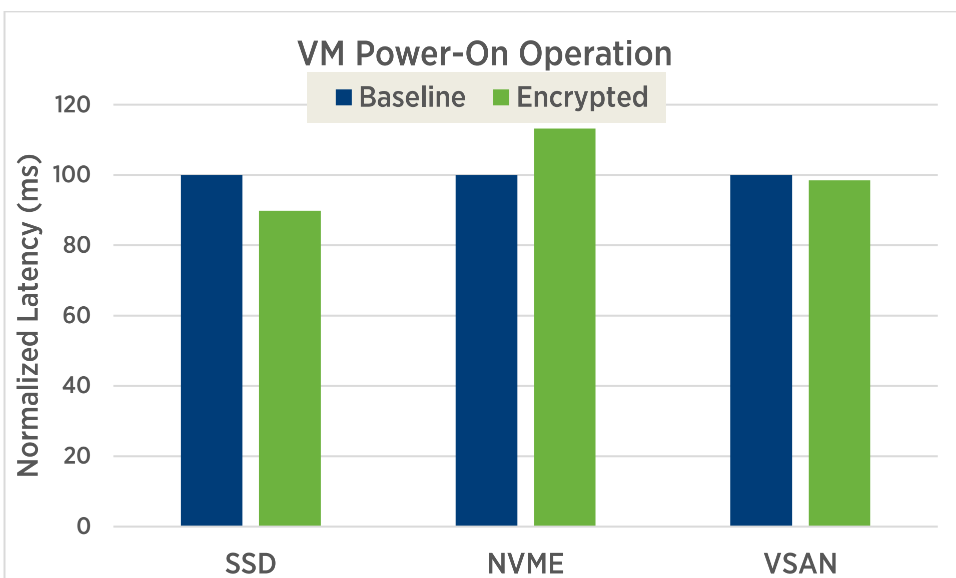

Figure 16. VM power-on operation

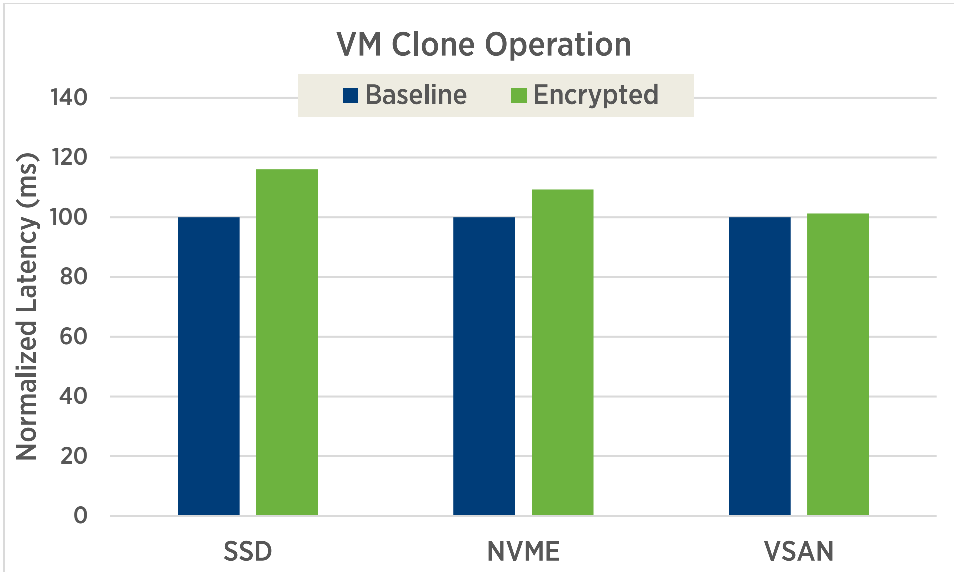

Figure 17. VM clone operation

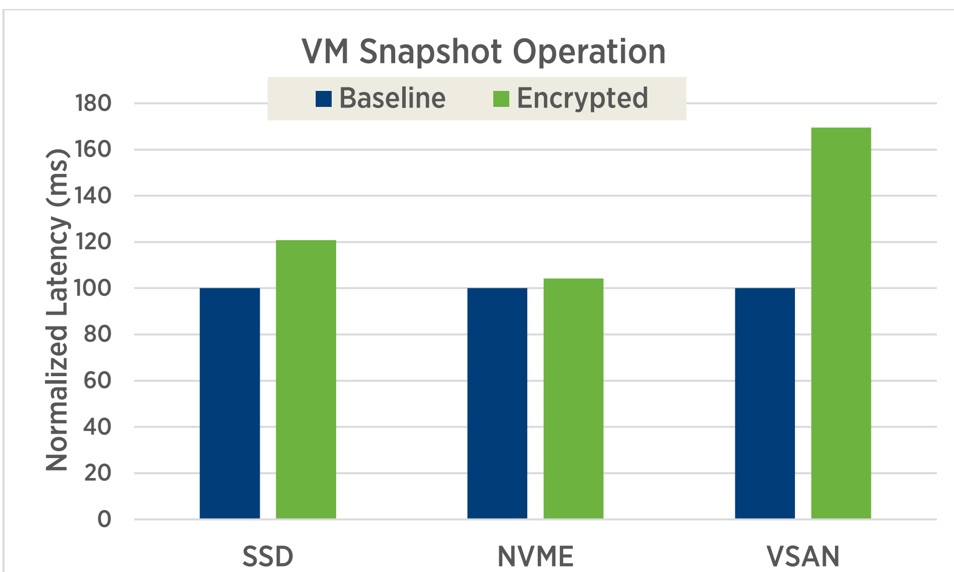

Figure 18. VM snapshot operation

Figure 16, Figure 17, and Figure 18 show the performance of virtual machine provisioning operations, where we normalize the latency to a baseline of 100 milliseconds for no encryption. As we can see, in the case of VM power-on and VM clone operations, the performance overhead of VM encryption is less than 20%, irrespective of the storage type. In the case of the VM snapshot operation run on a vSAN datastore, we see a noticeable overhead of about 70%. This is because with VM encryption enabled, the VAIO (IOFilter) framework creates additional files to store book-keeping information, which is created afresh whenever a snapshot is taken.

Because the overhead of the file create operation in vSAN is higher compared to VMFS datastores, we see a higher impact for VM encryption only in the case of the vSAN datastore.

Conclusion

VMware vSphere virtual machine encryption secures VM data at the cost of increased CPU cycles for encryption and decryption. For ultra-low latency devices like the NVMe drive we used, the impact of higher CPU cost directly translates to reduced throughput and increased I/O latency. However, for storage devices and subsystems in the latency range of a few hundred microseconds and above, the increased CPU cost does not translate to a significant increase in latency or a decrease in throughput. In the case of VM provisioning operations like VM power-on and clone, the overhead of encryption for the Linux VM we tested is less than 20% in the worst case and very minimal for most other cases.

References

[1] Intel Corp. (2012, October) Intel S3700 SATA SSD Data Sheet.
http://download.intel.com/newsroom/kits/ssd/pdfs/Intel_SSD_DC_S3700_Product_Specification.pdf

[2] Samsung Electronics Co., Ltd. (2015, October) Samsung PM1725 NVMe Data Sheet.
http://www.samsung.com/semiconductor/global/file/insight/2015/11/pm1725-ProdOverview-2015-0.pdf

[3] VMware, Inc. (2016) VMware vSAN Product Page.
http://www.vmware.com/products/virtual-san.html

[4] VMware, Inc. (2014, October) IO Analyzer.
https://labs.vmware.com/flings/io-analyzer

[5] Iometer.org. (2014) Iometer.
http://www.iometer.org

[6] HGST, Inc. (2015) HGST Ultrastar C10K1200 Data Sheet.
https://www.hgst.com/sites/default/files/resources/USC10K1200_ds_FINAL.pdf

[7] Intel Corp. (2012, August) Intel Advanced Encryption Standard (Intel AES) Instructions Set - Rev 3.01.
https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set

About the Authors

Aijaz Baig is a member of the VMware performance engineering group. His day-to-day work encompasses conducting experiments to characterize the performance of the ESXi storage stack (including VMware vSAN as well as emerging technologies like NVMe). He holds an MS in Electrical Engineering from the University of Linkoping, Sweden.

Sankaran Sivathanu leads an engineering team focusing on the performance of the core ESXi storage stack and is part of the VMware performance engineering group. His work at VMware includes performance characterization and tuning of traditional ESXi storage and VMware vSAN. He holds a PhD in Computer Science from the Georgia Institute of Technology.

Gnana Lakshmi is an engineer in the VMware performance engineering group. She works on performance aspects of vSphere management operations and vCenter Server performance characterization.

Acknowledgements

The authors would like to thank Julie Brodeur, Swapneel Kekre, Jesse Pool, and Adarsh Jagadeeshwaran for their valuable feedback on the paper.

Comments on this document: https://communities.vmware.com/docs/DOC-33163