PERFORMANCE STUDY – JANUARY 2017

VMWARE vSPHERE ENCRYPTED vMOTION ARCHITECTURE, PERFORMANCE, AND BEST PRACTICES

VMware vSphere 6.5

vmware®

Table of Contents

Executive Summary

With the rise in popularity of hybrid cloud computing, where VM-sensitive data leaves the traditional IT environment and traverses over the public networks, IT administrators and architects need a simple and secure way to protect critical VM data that traverses across clouds and over long distances.

The Encrypted vMotion feature available in VMware vSphere® 6.5 addresses this challenge by introducing a software approach that provides end-to-end encryption for vMotion network traffic. The feature encrypts all the vMotion data inside the vmkernel by using the most widely used AES-GCM encryption standards, and thereby provides data confidentiality, integrity, and authenticity even if vMotion traffic traverses untrusted network links.

Experiments conducted in the VMware performance labs using industry-standard workloads show the following:

- vSphere 6.5 Encrypted vMotion performs nearly the same as regular, unencrypted vMotion.
- The CPU cost of encrypting vMotion traffic is very moderate, thanks to the performance optimizations added to the vSphere 6.5 vMotion code path.
- vSphere 6.5 Encrypted vMotion provides the proven reliability and performance guarantees of regular, unencrypted vMotion, even across long distance networks.

Introduction

VMware vSphere® vMotion® [1] provides the ability to migrate a running virtual machine from one vSphere host to another, with no perceivable impact to the virtual machine's performance. vMotion brings enormous benefits to administrators—it reduces server downtime and facilitates automatic load-balancing.

During migration, the entire memory and disk state associated with a VM, along with its metadata, are transferred over the vMotion network. It is possible during VM migration for an attacker with sufficient network privileges to compromise a VM by modifying its memory contents during the transit to subvert the VM's applications or its guest operating system. Due to this possible security risk, VMware highly recommended administrators use an isolated or secured network for vMotion traffic, separate from other datacenter networks such as the management network or provisioning network. This protected the VM's sensitive data as it traversed over a secure network.

Even though this recommended approach adds slightly higher network and administrative complexity, it works well in a traditional IT environment where the customer owns the complete network infrastructure and can secure it. In a hybrid cloud, however, workloads move dynamically between clouds and datacenters over secured and unsecured network links. Therefore, it is essential to secure sensitive vMotion traffic at the network endpoints. This protects critical VM data even as the vMotion traffic leaves the traditional IT environment and traverses over the public networks.

vSphere 6.5 introduces Encrypted vMotion, which provides end-to-end encryption of vMotion traffic and protects VM data from eavesdropping occurrences on untrusted network links. Encrypted vMotion provides complete confidentiality, integrity, and authenticity of the data transferred over a vMotion network without any requirement for dedicated networks or additional hardware.

The sections that follow describe:

- vSphere 6.5 Encrypted vMotion technology and architecture
- How to configure Encrypted vMotion from the vSphere Client
- Performance implications of encrypting vMotion traffic using real-life workload scenarios
- Best practices for deployment

Encrypted vMotion Architecture

vMotion uses TCP as the transport protocol for migrating the VM data. To secure VM migration, vSphere 6.5 encrypts all the vMotion traffic, including the TCP payload and vMotion metadata, using the most widely used AES-GCM encryption standard algorithms, provided by the FIPS-certified vmkernel vmkcrypto module.

Encryption Protocol

Encrypted vMotion does not rely on the Secure Sockets Layer (SSL) or Internet Protocol Security (IPsec) technologies for securing vMotion traffic. Instead, it implements a custom encrypted protocol above the TCP layer. This is done primarily for performance, but also for reasons explained below.

SSL is compute intensive and completely implemented in user space, while vMotion, which constitutes core ESXi, executes in kernel space. This means, if vMotion were to rely on SSL, each encryption/decryption call would need to traverse across kernel and user spaces, thereby resulting in excessive performance overhead. Using the encryption algorithms provided by the vmkernel vmkcrypto module enables vMotion to avoid such a performance penalty.

Although IPSec can be used to secure vMotion traffic, its usability is limited in the vSphere environment because ESXi hosts support IPsec only for IPv6 traffic, but not for IPv4 traffic. Besides, implementing a custom protocol above the TCP layer gives vMotion the ability to create the appropriate number of vMotion worker threads, and coordinate efficiently among them to spread the encryption/decryption CPU load across multiple cores.

Encryption Algorithms

Encrypted vMotion relies on the AES-GCM (Advanced Encryption Standard / Galois Counter Mode) encryption algorithms provided by the FIPS-certified vmkcrypto module in the vmkernel for providing confidentiality and integrity together efficiently.

Defending Against Replays

Every vMotion message contains a unique replay attack counter generated internally; each counter is expected to be received only once. This counter is part of the authenticated associated data of AES-GCM, hence its integrity is protected; any modification of this counter is not possible without detection.

Workflow

Figure 1 below illustrates the Encrypted vMotion workflow.

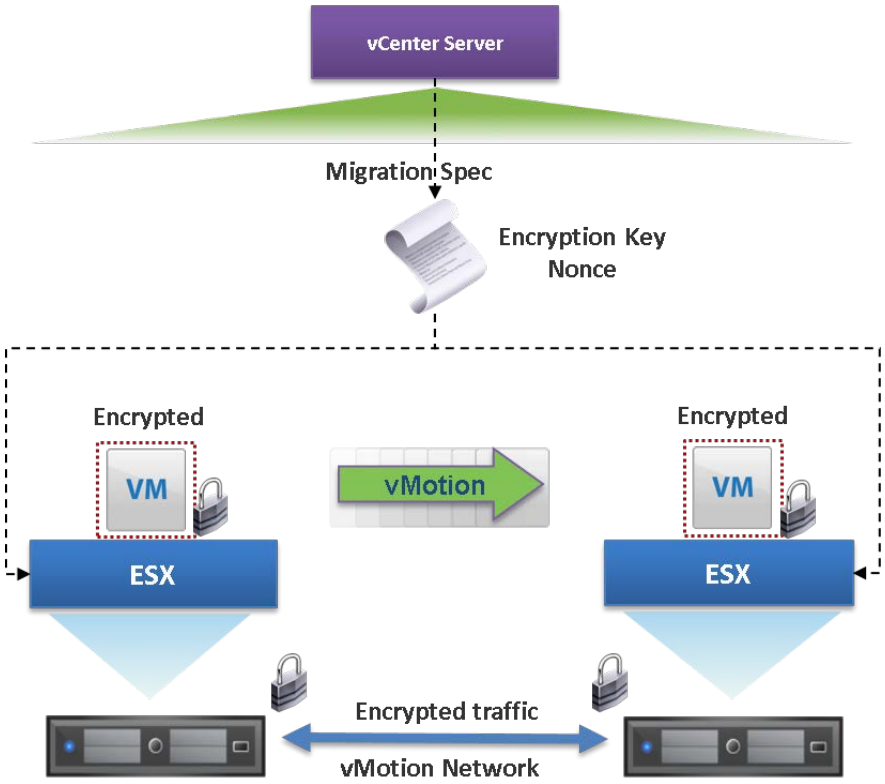

Figure 1. Workflow of Encrypted vMotion

As shown in the figure, vCenter Server prepares the migration specification that consists of a 256-bit encryption key and a 64-bit nonce, then passes the migration specification to both source and destination ESXi hosts of the intended vMotion. Both the ESXi hosts communicate over the vMotion network using the key provided by vCenter Server. The key management is simple: vCenter Server generates a new key for each vMotion, and the key is discarded at the end of vMotion. As noted in Figure 1, encryption happens inside the vmkernel, hence there is no need for specialized hardware.

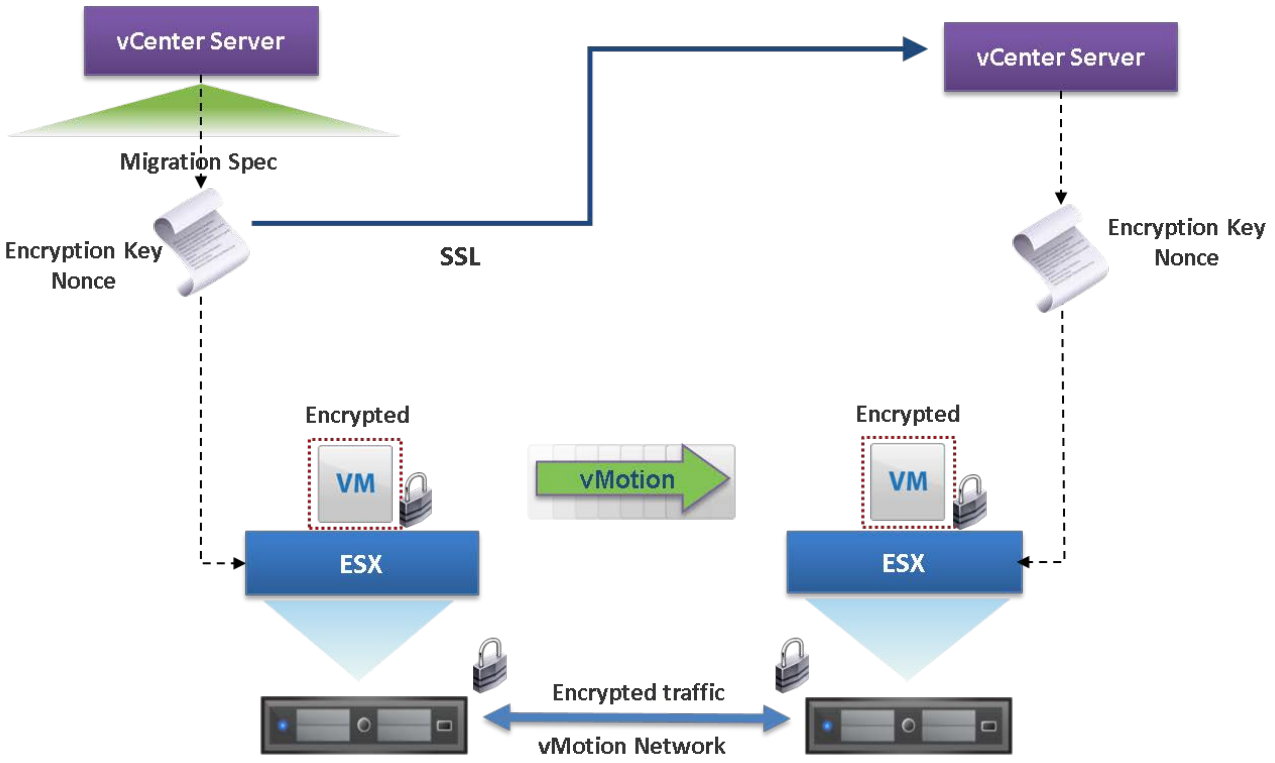

Figure 2. Workflow of Encrypted Cross vCenter Server vMotion

Figure 2 above illustrates the Encrypted vMotion workflow across vCenter Servers. The workflow of Encrypted Cross vCenter Server vMotion is very similar to the Encrypted vMotion workflow, except that the source vCenter Server that generates the migration specification passes it to the destination vCenter through an SSL channel.

In summary, vSphere 6.5 Encrypted vMotion achieves confidentiality, integrity, and authenticity of data transferred over the vMotion network as follows:

- Confidentiality and Integrity: AES-GCM
- Authentication: vCenter Server is the trusted 3rd party

Encrypted vMotion Configuration

Encrypted vMotion can be configured on a per-VM basis, with support for three modes of configuration: Disabled, Opportunistic, and Required.

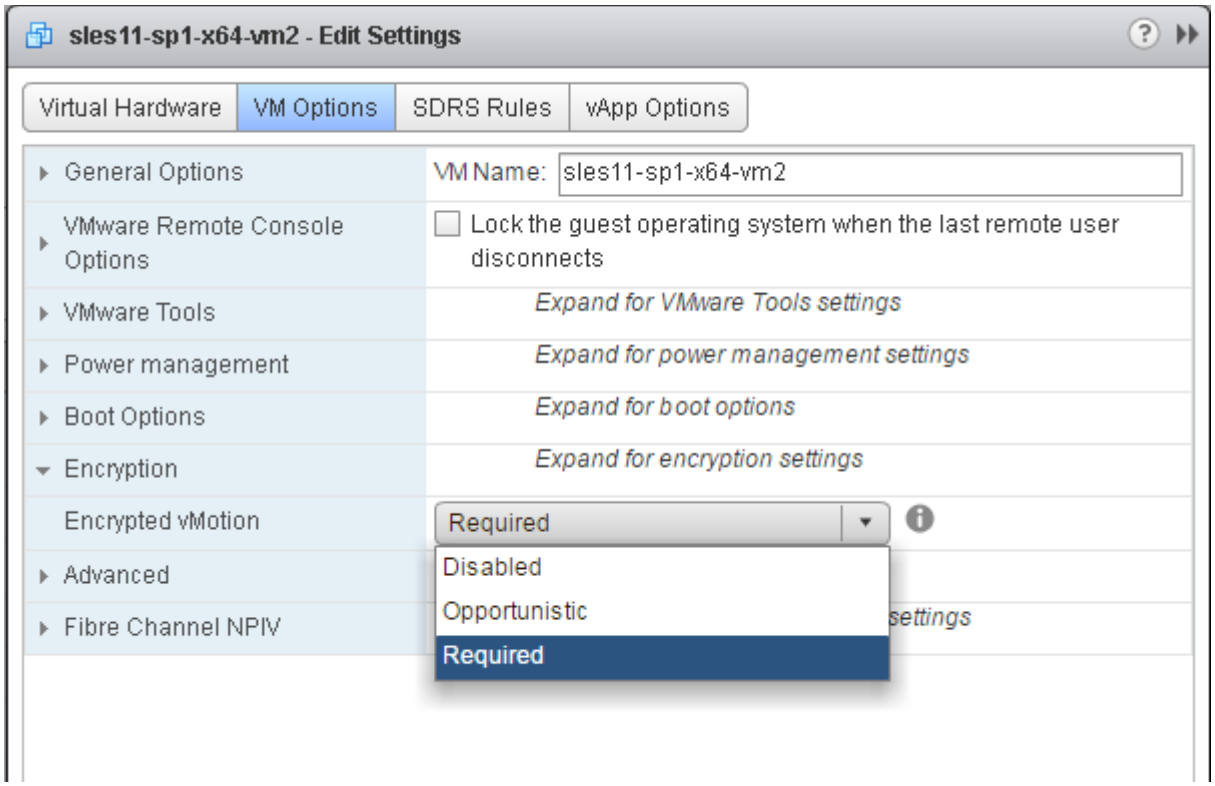

Figure 3. Encrypted vMotion configuration modes

Figure 3 shows how a VM can be configured for Encrypted vMotion.

- **Disabled:** vMotion traffic is not encrypted, and VM data is vulnerable to man-in-the-middle attacks.
- **Opportunistic (Default):** vMotion traffic is encrypted when both the source and destination hosts of the intended migration are capable of supporting Encrypted vMotion (that is, both hosts are ESXi 6.5). If a virtual machine running on ESXi 6.5 is migrated to a host running an earlier ESX/ESXi version, vMotion traffic is not encrypted.
- **Required:** vMotion will be restricted to the hosts (that is, hosts running ESXi 6.5) that support Encrypted vMotion. For instance, in a DRS enabled cluster, DRS will select only the destination ESXi host that supports encrypted vMotion.

Performance Test Configuration and Methodology

This section describes the testbed configuration and general testing methodology.

Test Configuration

vSphere 6.5 vMotion performance tests used a pair of identical Dell PowerEdge R730 servers with Intel Broadwell–based processors. Each of the servers was configured with dual-socket, twelve-core 3GHz Intel Xeon E5-2687W processors, 1TB of RAM, four 10 GbE Intel network adaptors, and Intel NVMe storage. Both the servers also were connected to a common EMC VNX5500 SAN array.

Measuring Encrypted vMotion Performance

The following metrics were used to understand the performance implications of vSphere 6.5 Encrypted vMotion.

- **Migration Time:** The total time taken for the migration to complete, beginning from the initiation of the migration.
- **Execution Switch-Over Time:** The time during which the virtual machine is quiesced to enable virtual machine switchover from the source host to the destination host.
- **Guest Penalty:** The performance impact (latency and throughput) on the applications running inside the virtual machine during and after the migration.
- **CPU Overhead:** The CPU costs of encrypting/decrypting the vMotion traffic

To investigate the performance implications of vSphere 6.5 Encrypted vMotion, two critical tier-1 applications were considered:

- **Redis**, an in-memory key-value database application characterized by memory-intensive operations, was used to investigate the encrypted vMotion performance implications in private cloud deployment scenarios.
- **SQL Server**, a traditional database server application characterized by disk-intensive operations, was used to investigate the encrypted vMotion performance implications during long distance migrations.

In order to compare performance fairly, we ran the same workload on the same virtual machine with and without encryption enabled on vMotion traffic.

Encrypted vMotion Performance in a Private Cloud Deployment

We chose Redis, an in-memory key-value database, as the candidate with which to study the vSphere 6.5 Encrypted vMotion performance implications in private cloud deployment scenarios. Redis is the most popular implementation of a key-value database, according to DB-Engines ranking [2]. Redis is very demanding on several subsystems, including CPU and memory, which makes it very effective as a whole system benchmark.

Performance Impact on Duration and Switch-Over Time

In this study, we investigate the impact of encryption on vMotion duration and switch-over time under several test scenarios.

1. In the first set of scenarios, we migrated an active Redis VM with a varying mix of read and write operations.
2. In the second set of scenarios, we migrated an idle Redis VM while varying the vCPU and memory sizes.

The test scenario for the active Redis VM migration tests includes the following:

- **VM:** Four instances of Redis 3.0.3 server were deployed in a single virtual machine configured with 8 vCPUs and 150GB memory
- **Guest OS**: SUSE Linux Enterprise Server 11 x64
- **Key space:** Each Redis server used a key space of 100 million keys
- **Benchmark:** Four redis-benchmark instances ran a mix of SET and GET operations; this benchmark is included with the Redis distribution
- **vMotion network:** Four 10 Gigabit Ethernet (GbE) ports, resulting in a total of 40 GbE network bandwidth

The following table shows the operations mix during different load scenarios.

Scenario	GET Operations per second	SET Operations per second
Idle	0	0
100% GET / 0% SET	200,000	0
75% GET / 25% SET	150,000	50,000
50% GET / 50% SET	100,000	100,000
25% GET / 75% SET	50,000	150,000
100% GET / 0% SET	0	200,000

Table 1. GET and SET operations mix during different load scenarios

Table 1 shows the aggregate GET and SET operations per second served by all the Redis server instances during different load scenarios. Since Redis is an in-memory database application, the CPU load on the virtual machine was similar in all scenarios, irrespective of the mix in memory read I/O and memory write I/O operations. During the steady-state period of the benchmark, the CPU utilization (esxtop %USED counter) of the virtual machine was about 550%, which is equivalent to a 70% total CPU utilization.

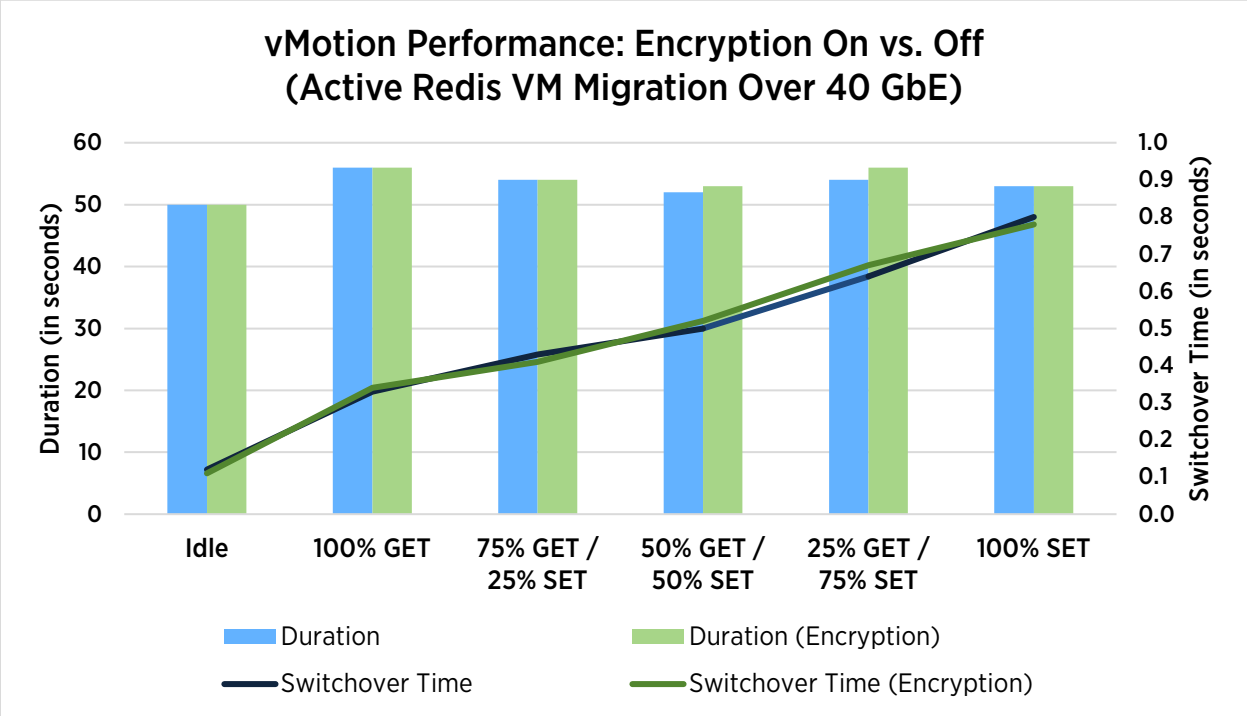

Figure 4. We compared the migration time of one active VM running a Redis workload of varied GET and SET operations while not encrypted (blue) and then while encrypted (green); performance is almost the same with and without encryption

Figure 4 compares the vMotion duration and switch-over time under different test scenarios with and without encryption enabled.

Let us first consider the vMotion duration shown by the bars plotted on the primary y-axis. The figure shows identical performance in duration in a majority of the test scenarios with and without encryption enabled on vMotion traffic. In the other test scenarios, the difference is less than 5%.

The line graphs plotted on the secondary y-axis show vMotion switch-over time. The figure shows a gradual increase in the switch-over time corresponding to the increase in SET operations in the workload mix. For instance, the switch-over duration increased from about 0.3 seconds in the "100% GET operations" test scenario to about 0.8 seconds in the "100% SET operations" test scenario. This is expected because, unlike a GET operation that only accesses a memory page, a SET operation results in modifying a memory page.

Write-intensive workloads (where the virtual machine is dirtying large amounts of memory pages) tend to increase the vMotion switch-over time due to the need to send more dirty memory pages during the vMotion switch-over duration. Read-heavy workloads (where the virtual machine is only modifying small amounts of memory pages), on the other hand, require very few memory pages to be sent during switch-over time, resulting in a lower switch-over time. The figure shows that in all the test scenarios, irrespective of the write I/O mix, the impact of encryption was minimal on vMotion switch-over time.

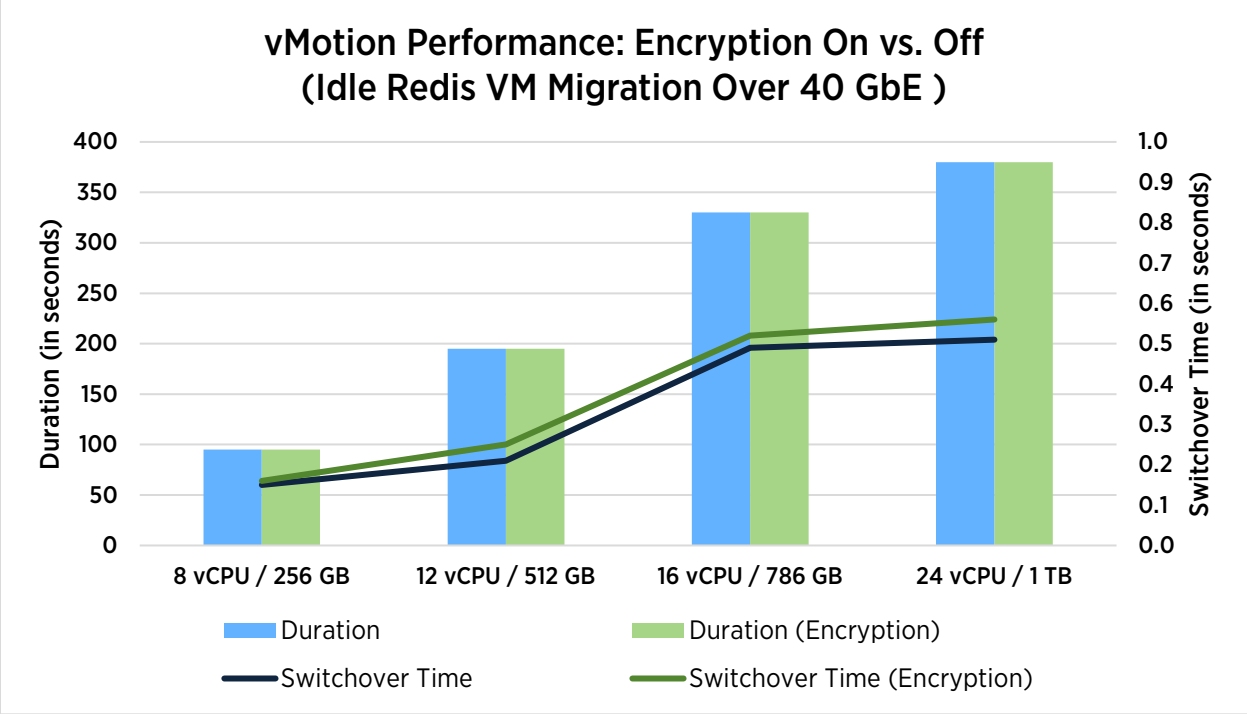

Figure 5. We compared the migration time of one idle Redis VM while it was not encrypted with the same idle VM when it was encrypted; performance is very similar with and without encryption

Figure 5 shows the vMotion duration and switch-over time in idle virtual machine test scenarios in which we varied vCPU and memory sizes. The idle virtual machine test presents a scenario in which the virtual machine is idle (CPU activity is zero) but its memory has been completely touched and modified by the guest. In other words, this scenario is different from a freshly booted virtual machine that is assigned zero pages (pages that contain nothing but zeroes). vMotion has optimizations to reduce the amount of data transferred for zero pages.

Let us first consider the vMotion duration shown by the bars plotted on the primary y-axis. The figure shows nearly identical performance in duration in all the scenarios with and without encryption enabled on vMotion traffic.

The line graphs plotted on the secondary y-axis show vMotion switch-over time. The figure shows a gradual increase in the switch-over time corresponding to the increase in VM configuration. For instance, the switch-over duration increased from about 0.15 seconds in the "8vCPU/256GB" test scenario to about 0.5 seconds in the "24vCPU/1TB" test scenario. This is because the large VM configurations tend to increase the VM power-on

time on the destination host, resulting in a higher switch-over time. The figure shows that in all the test scenarios, irrespective of the VM configurations, the impact of encryption was very marginal on vMotion switch-over time. For instance, in the "24vCPU/1TB" test scenario, the increase in the switch-over time was about 0.05 seconds when vMotion traffic was encrypted.

Both figures 4 and 5 show that the impact of encrypting vMotion traffic is quite minimal on both vMotion duration and switch-over time.

Performance Impact on Throughput and CPU Usage

To investigate the peak vMotion throughput performance, we simulated a very heavy memory usage footprint in the virtual machine. In the test scenario, we used a VM configured with 16 vCPUs and 512GB memory.

The test scenario includes the following:

- **VM:** Eight instances of Redis 3.0.3 server deployed in a single virtual machine configured with 16 vCPUs and 512GB memory
- **Key space:** Redis key space of 2 billion keys
- **Benchmark/Workload:** redis-benchmark, 160 clients, pipeline: 5 requests, 100% SET operations

The workload consisted of 160 clients, each of which repeatedly ran a SET operation of 1 byte using a random key for every operation, chosen from a space of 2 billion possible keys. The workload was extremely CPU and memory intensive, because each SET operation resulted in a dirty memory page. During the steady state of the benchmark, the eight Redis server instances were together serving over 2 million requests per second.

We migrated this virtual machine under different test scenarios:

- vMotion over 10 Gigabits per second (Gb/s) network
- vMotion over 20Gb/s network,
- vMotion over 30Gb/s network
- vMotion over 40Gb/s network.

We used esxtop to capture the network throughput and CPU utilization on the source and destination hosts during the steady state.

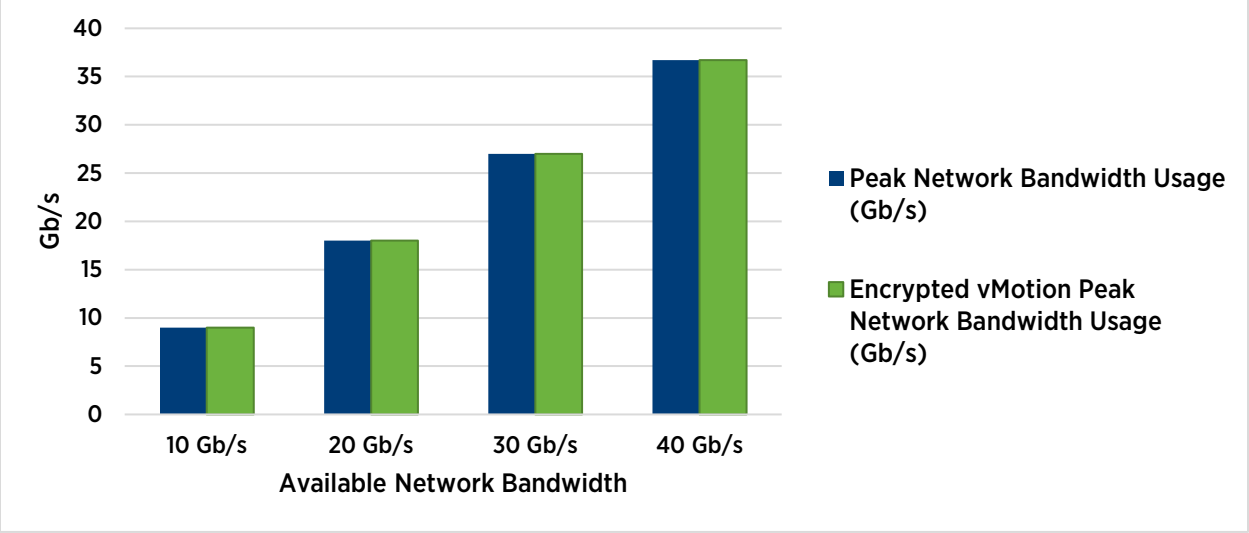

Figure 6. vMotion peak network bandwidth usage

Figure 6 shows that vMotion reaches line rate in all four scenarios with and without encryption enabled.

Figures 7 and 8 show the CPU overhead of encrypting vMotion traffic on source and destination hosts, respectively. The CPU usage is plotted in terms of the CPU cores required by vMotion.

For instance, let us first consider the CPU usage for the 20 GbE scenario. Figure 7 shows that on the source host, vMotion without encryption required about 1 core, while encrypted vMotion required a little over 2 cores. In comparison, Figure 8 shows that on the destination host, vMotion without encryption required about 2.5 cores, while encrypted vMotion required a little over 3 cores.

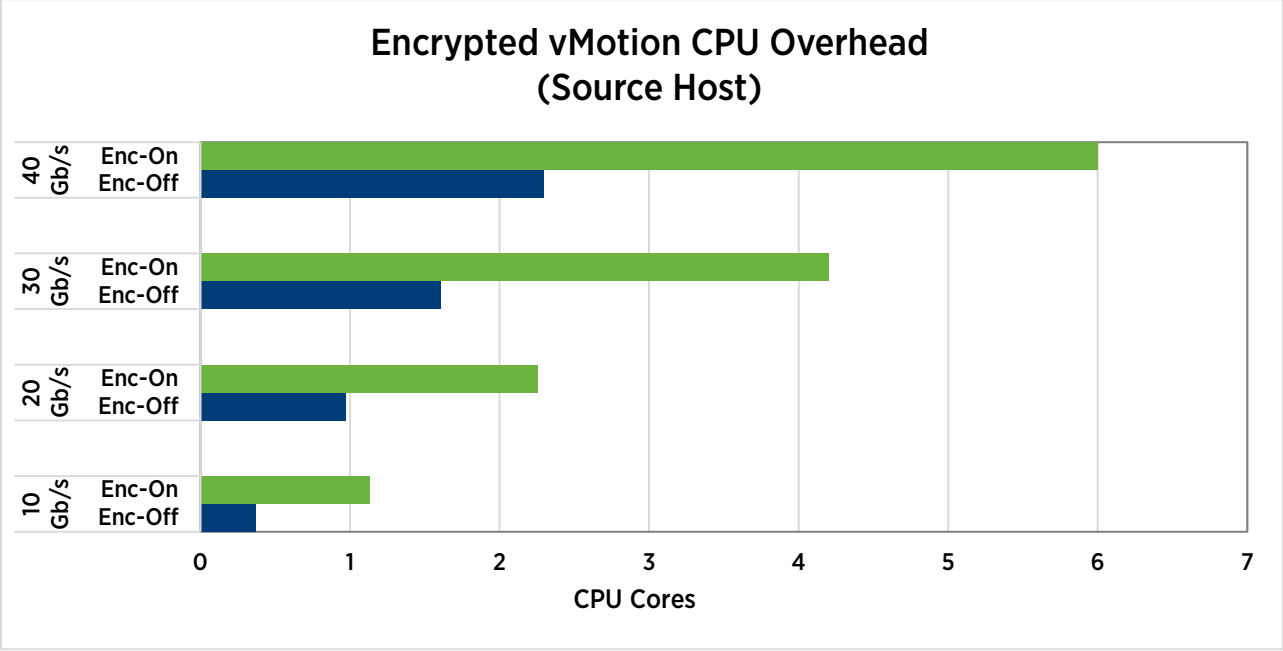

Figure 7. Encrypted vMotion CPU overhead on the source host

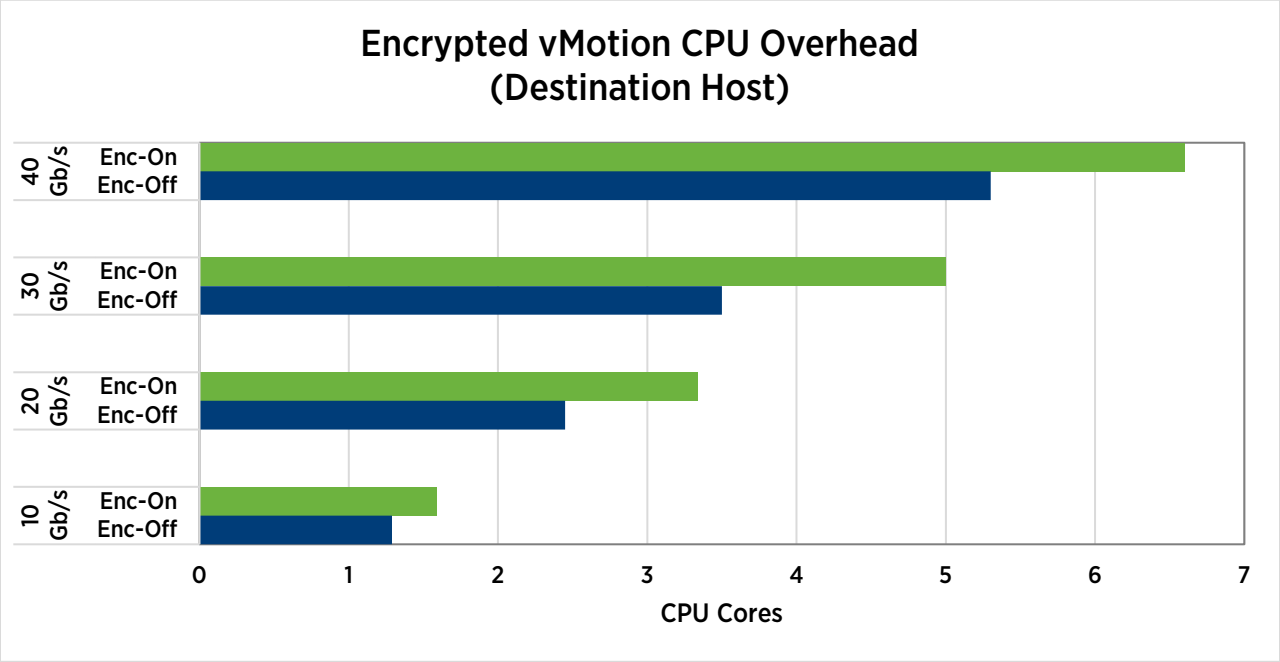

Figure 8. Encrypted vMotion CPU overhead on the destination host

vMotion CPU usage in figures 7 and 8 reveal two interesting trends. First, we note that vMotion in general needs more CPU on the destination host than on the source host. This is because the vMotion receive code path is more expensive than the vMotion transmit code path. Second, we observe that although vMotion requires less CPU on the source host than on the destination host, the encryption overhead is higher on the source host. This is because encryption is much more compute intensive than decryption.

In summary, the figures show that CPU requirements of encrypted vMotion are very moderate. For every 10Gb/s of vMotion traffic, encrypted vMotion only requires less than one core on the source host and less than half a core on the destination host for all the encryption-related overheads.

Encrypted vMotion Performance Over Long Distance

Database workloads are widely acknowledged to be very resource intensive. They are characterized by high consumption of CPU, memory, and storage resources, hence they serve as a very effective test of vMotion performance. This study investigates the impact of long distance encrypted vMotion on Microsoft SQL Server online transaction processing (OLTP) performance.

Test Methodology

Figure 9 shows the changes we made to our testbed for long distance migration tests. As shown in the figure, vMotion traffic between the source and destination vSphere hosts passed through a Maxwell physical appliance to induce latency. We used a 1 GbE link for the vMotion network to simulate a realistic bandwidth for long distance migration.

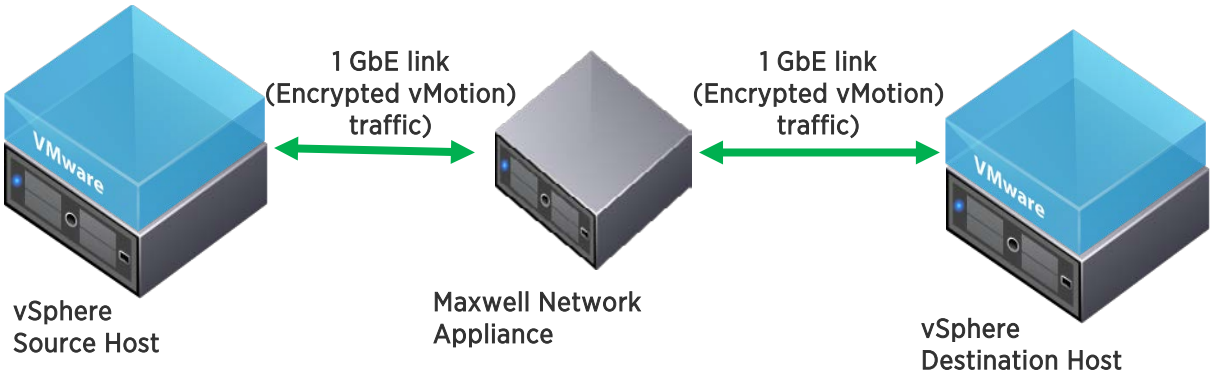

Figure 9. Testbed for long distance encrypted migrations

Load Generation Software

The open-source DVD Store Version 2 (DS2) [3] was used as the benchmark tool. DS2 simulates an online ecommerce DVD store, where customers log in, browse, and order products. It is designed to utilize a number of advanced database features, including transactions, stored procedures, triggers, and referential integrity. The main DS2 metric is orders per minute (OPM).

In our test, we configured the DVD store benchmark driver to enable fine-grained performance tracking, which helped to quantify the impact on SQL Server throughput (orders processed per second) during different phases of vMotion. Specifically, we edited the source code of the ds2xdriver.cs file with 1-second granularity, which resulted in the client reporting the performance data every 1 second (default: 10 seconds).

The test case used the following benchmark deployment scenario:

- Microsoft SQL Server was deployed in a Windows Server 2012 virtual machine configured with eight vCPUs, 24GB of memory, 40GB of system disk, and 30GB of database disk
- The DS2 workload used a database size of 12GB
- Benchmark: 5 DS2 users with 10 milliseconds (ms) think-time

Tests varied the Maxwell packet delay parameter (used to inject latencies) to simulate a 5ms to 150ms round-trip latency in the vMotion network. We subjected all of the vSphere 6.5 Encrypted vMotion network traffic from the source host to the destination host to the network latencies added by the Maxwell appliance. We also tested a baseline deployment scenario without any Maxwell delay. The baseline deployment scenario had a round-trip latency of 0.5ms. In all the test scenarios, we did not use the shared storage between the source and destination hosts. Hence, the entire state of the VM—including disk and memory contents—was transferred from the source host to the destination host.

Results

Figure 10 shows the total duration of vMotion during the migration of an active SQL Server VM at different network round-trip latencies. We initiated the migration during the steady-state period of the benchmark when the CPU utilization (esxtop %USED counter) of the VM was close to 110%, and the average read IOPS and average write IOPS were about 60 and 230, respectively.

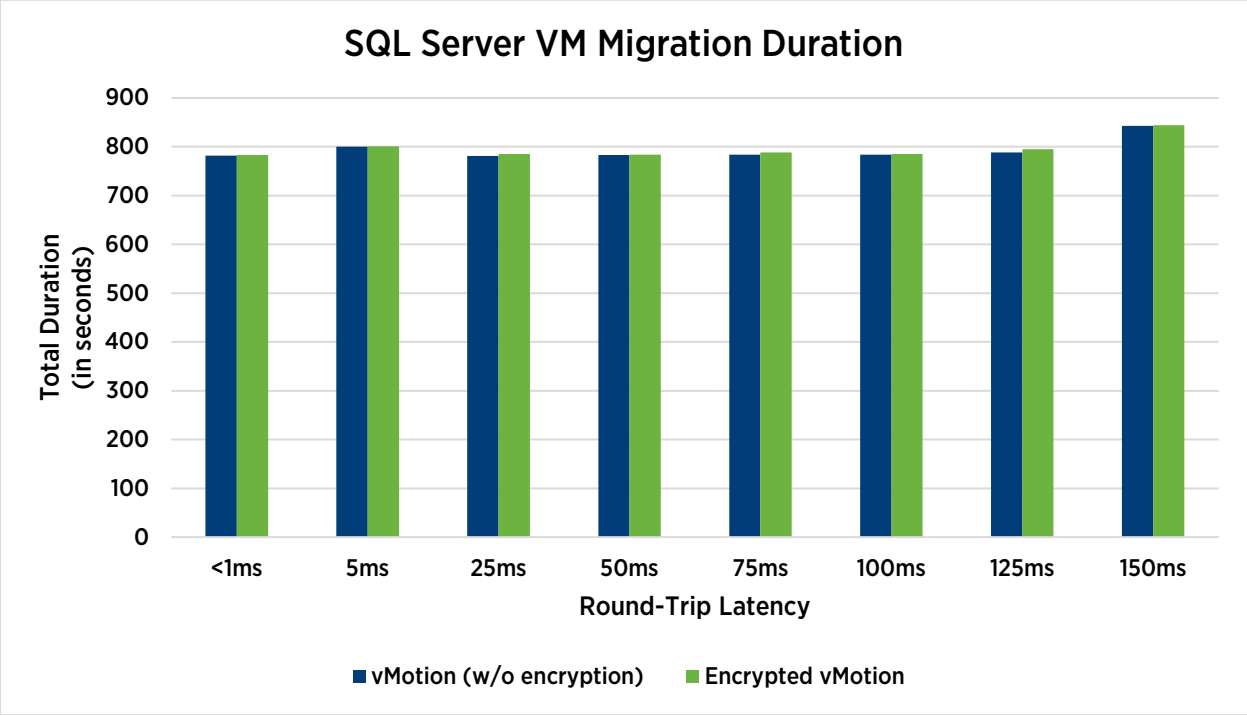

Figure 10. Encrypted vMotion duration at varying network latency

Figure 10 shows that the duration of the migration in the 150ms scenario was nearly identical to the duration in the 0.5ms baseline scenario. This was achieved because of the latency-aware optimizations in vMotion. The duration of the migration was nearly identical in all the scenarios with and without encryption.

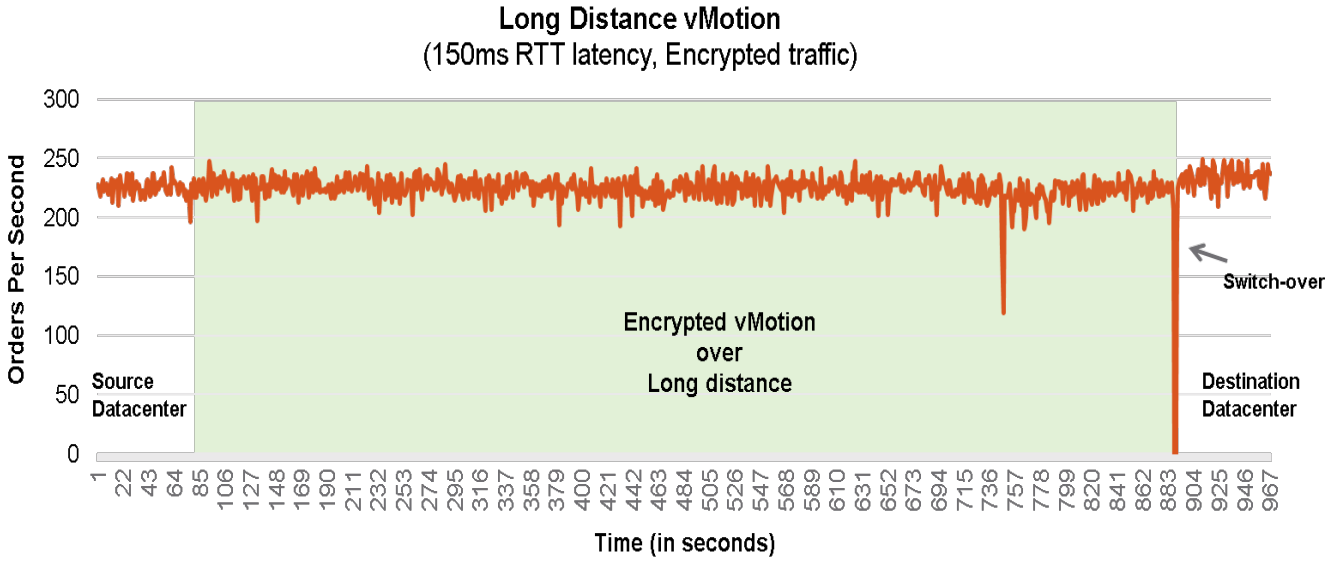

Figure 11. SQL Server performance before, during, and after long distance encrypted vMotion

Figure 11 plots the performance of a SQL Server virtual machine in orders processed per second at a given time—before, during, and after encrypted vMotion on a 150ms round-trip latency network. As shown in the figure, the impact on SQL Server throughput was minimal during vMotion. The only noticeable dip in performance was during the switch-over phase (in the range of 1 second) from the source to destination host. It took less than few seconds for the SQL server to resume its normal level of performance.

These results indicate vSphere 6.5 Encrypted vMotion performs well even on long distance area networks with up to 150ms round-trip latencies.

Encrypted vMotion Performance Best Practices

To obtain the best vSphere 6.5 Encrypted vMotion performance, we recommend the following:

- While Encrypted vMotion does not impose any new hardware requirements, it is highly recommended to use CPUs that support **Advanced Encryption Standard New Instruction Set** (or the **AES-NI**). AES-NI is an extension of x86 instruction set architecture that improves the speed of applications performing encryption and decryption using the Advanced Encryption Standard (AES).
- Encrypted vMotion uses additional CPU resources for encryption and decryption processing. The amount of CPU usage is less than a processor core for each 10Gb/s network interface on the source host, and less than half a core for each 10Gb/s network interface on the destination host. This usage is in addition to the vMotion CPU usage for network processing. The total amount of CPU resource for Encrypted vMotion thus requires less than two processor cores for each 10Gb/s network interface on the source host, and less than one and half cores for each 10Gb/s network interface on the destination host. Therefore, leaving some unreserved CPU capacity in a host can help ensure that vMotion tasks get the resources required in order to fully utilize available network bandwidth.
- Encrypted vMotion provides the same performance guarantees as regular, unencrypted vMotion. Hence, all past performance best practices recommended for regular, unencrypted vSphere 6.0 vMotion still apply to vSphere 6.5 Encrypted vMotion. See the *Performance Best Practices for VMware vSphere 6.0* [4] guide for full vMotion performance recommendations.

Conclusion

With recent advances in cloud computing and long distance mobility, hybrid cloud computing environments are increasingly viable and compelling. Administrators of these new geographically dispersed, multi-site, hybrid-cloud infrastructures need simple, reliable, and performant mechanisms to secure critical VM data that traverses on the public network links not owned by their organizations.

vSphere 6.5 vMotion introduces Encrypted vMotion that provides end-to-end encryption of vMotion traffic without any additional requirement for dedicated networks or additional hardware.

The experiments conducted in VMware performance labs using industry-standard workloads show that vSphere 6.5 Encrypted vMotion:

- Provides nearly identical performance as regular, unencrypted vMotion, with low CPU overheads
- Migrates workloads non-disruptively over long distances such as New York to London

References

[1] VMware, Inc. (2011, August) VMware vSphere vMotion Architecture, Performance and Best Practices in VMware vSphere 5.
http://www.vmware.com/techpapers/2011/vmotion-architecture-performance-and-best-practi-10202.html

[2] Monthly DB-Engines rankings of database management systems. (2016, November).
http://db-engines.com/en/ranking/

[3] Todd Muirhead and Dave Jaffe. (2007, December) linux.dell.com.
http://linux.dell.com/dvdstore/2007-12-03/

[4] VMware, Inc. (2015, June) Performance Best Practices for VMware vSphere 6.0.
http://www.vmware.com/files/pdf/techpaper/VMware-PerfBest-Practices-vSphere6-0.pdf

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 area. He has published his findings in a number of white papers and has presented them at various technical conferences. He has a master's degree in computer science from the University of Texas, Austin.

Acknowledgements

The author would like to sincerely thank Yanlei Zhao, Arun Ramanathan for reviews and contributions to the paper. He also would like extend thanks to members in his team including Juan Garcia-Rovetta and Julie Brodeur for their reviews.

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