Interrupt Frequency Optimization Method for Cryptographic Devices Based on Single-Root I/O Virtualization (Postprint)

Authors: Li Shuai, Sun Lei, Guo Songhui

Date: 2018-06-19T00:00:00+00:00

Abstract

To address the problem of excessive interrupt frequency in virtualization environments affecting the cryptographic computation performance of cryptographic devices, a performance optimization method for reducing interrupt frequency is proposed. First, an interrupt frequency control model is established, and its rationality and correctness are verified through experiments. Then, based on Single-Root I/O Virtualization (SR-IOV), a speed monitoring module is incorporated into the Virtual Function driver layer to monitor encryption speed variations in real time, and when this module detects a decrease in encryption speed, it automatically adjusts the Virtual Function interrupt frequency cap, thereby reducing the I/O transmission overhead caused by excessive interrupt frequency. Experimental results demonstrate that adjusting the interrupt frequency limit significantly improves the encryption speed of I/O-intensive encryption processes.

Full Text

Interrupt Frequency Optimization Method for Cryptographic Devices Based on Single-Root I/O Virtualization

Li Shuai, Sun Lei, Guo Songhui
(Information Engineering University, Zhengzhou 450001, China)

Abstract: To address the problem that excessive interrupt frequency in virtualized environments degrades the cryptographic computation performance of encryption devices, this paper proposes a performance optimization method for reducing interrupt frequency. First, an interrupt frequency control model is established and its rationality and correctness are verified through experiments. Then, based on single-root I/O virtualization, a speed monitoring module is incorporated into the virtual function driver layer to monitor encryption

speed changes in real time. When this module detects a decrease in encryption speed, it automatically adjusts the upper limit of the virtual function interrupt frequency, thereby reducing I/O transmission overhead caused by excessive interrupt frequency. Experimental results demonstrate that adjusting the interrupt frequency upper limit significantly improves encryption speed during I/O-intensive cryptographic processes.

Keywords: interrupt frequency; single-root I/O virtualization; control model; encryption speed

0 Introduction

Cloud computing represents a new computing paradigm following distributed computing, grid computing, and peer-to-peer computing. Built upon network, virtualization, and distributed computing technologies, it provides cloud users with powerful computational and storage capabilities. However, with the rapid development of cloud computing, its security challenges have become increasingly prominent. In 2018, the Cloud Security Alliance (CSA) announced twelve top security threats facing cloud computing, posing significant challenges to cloud security. In response, major cloud service providers (CSPs) have proposed their own solutions. For instance, to address data leakage in the cloud, Alibaba Cloud applies hardware security modules (HSM) in the underlying hardware, providing data encryption services to upper-layer applications through virtualization technology. HSM itself is a cryptographic device, also known as a cryptographic accelerator, which ensures secure key management while enabling fast cryptographic operations. Beijing Sansec Technology Co., Ltd., a cloud security service provider based on cryptographic technology, has also proposed its own solution, which uses cryptographic cards as physical support at the server side. These cryptographic cards employ single-root I/O virtualization (SR-IOV) technology to achieve high-performance resource sharing in cloud environments.

Cloud computing offers three distinct service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Its greatest advantage lies in service efficiency, which is primarily reflected in I/O transmission performance in cloud environments. Therefore, I/O performance in cloud environments is crucial for high-performance computer systems, and most cloud service providers have adopted corresponding I/O virtualization technologies to provide users with flexible and efficient resource sharing. The emergence of SR-IOV technology has significantly improved I/O transmission performance under virtualization conditions. It employs Passthrough I/O transmission without requiring the virtual machine monitor (VMM) to supervise virtual machines (VMs), enabling direct I/O transmission between hardware and VMs. During transmission, the input/output memory management unit (IOMMU) reduces overhead for memory protection and address translation, substantially improving transmission efficiency. However, when excessive external interrupt requests arrive at SR-IOV devices, I/O performance is severely impacted. This performance degradation directly affects the encryption per-

formance of SR-IOV cryptographic devices, contradicting the requirements of high-performance cloud computing.

This paper uses the Intel Corporation DH895XCC Series QAT cryptographic device as the experimental object, employs SR-IOV technology to build a virtual cipher machine (VCM) cluster on the KVM platform, and proposes a control scheme that automatically adjusts the VF interrupt frequency upper limit by monitoring encryption speed changes. This scheme overcomes the excessive performance overhead caused by numerous virtual interrupts and leverages the configurable characteristics of VF drivers to implement SR-IOV cryptographic devices in KVM, optimizing resource waste caused by excessive interrupts and fully exploiting the performance of SR-IOV cryptographic devices.

1 Related Work and Technology

To address the challenge of excessive interrupt frequency in virtualized environments degrading the cryptographic computation performance of SR-IOV cryptographic devices, this paper completes the following work: (a) Model establishment: Using the AES algorithm as an example, it analyzes the relationship between encryption speed, encryption byte blocks, and interrupt frequency, establishing a corresponding mathematical model and theoretically deriving the relationship among the three factors. Results show that the primary factor affecting VCM encryption performance is interrupt frequency magnitude. (b) Interrupt frequency optimization: It proposes a control scheme that automatically adjusts the virtual function (VF) interrupt frequency upper limit by monitoring encryption speed changes. This scheme incorporates a speed monitoring module in the VF driver layer to monitor encryption speed changes in real time, adjusting the VF interrupt frequency upper limit only once during the entire monitoring process to maximize VCM encryption performance. (c) Experimental verification: Experimental results demonstrate that the optimized model maximally maintains VCM encryption performance.

1.1 Related Work

Academic research on SR-IOV device interrupts has primarily approached the problem from two perspectives: first, directly targeting interrupt frequency to achieve performance optimization by reducing interrupt frequency; second, targeting interrupt mechanisms to achieve performance optimization by improving interrupt paths.

Regarding methods that directly reduce excessive interrupt frequency to improve I/O performance, Dong et al. identified the performance degradation issue caused by SR-IOV interrupts and proposed a coarse-grained interrupt frequency control method to optimize virtual interrupt handling in XEN. However, this method requires predetermined optimal interrupt frequency upper limits for VFs, representing a static configuration without universality. Guan et al. proposed an event-based polling model that eliminates interrupts in critical I/O

processing paths, but this model requires APIC emulation in the host kernel, adding extra host overhead. Li et al. built upon the coarse-grained interrupt control method to propose an adaptive interrupt frequency control approach. This method incorporates an interrupt frequency control module to control interrupt frequency and improve VM performance by judging the relationship between the interrupt frequency received by SR-IOV devices and the optimal interrupt frequency. However, this method increases I/O latency and is unsuitable for I/O-intensive workloads, having only been implemented for network card virtualization.

For interrupt mechanism optimization, academia has focused on reducing excessive CPU resource consumption when large numbers of interrupts arrive, thereby improving I/O transmission performance. Gordon et al. proposed a method with fewer interrupt exits (ELI) that employs software simulation to establish a shadow interrupt descriptor table (IDT) within the guest VM to directly handle interrupts. However, in the ELI mechanism, the shadow IDT can only receive interrupts already allocated to the guest, while other unallocated interrupts still require host involvement. Moreover, when numerous virtual interrupts need processing, significant performance overhead still occurs. Tu et al. proposed a direct interrupt delivery (DID) method that completely eliminates VM-exits during VM interrupt processing, optimizing the interrupt handling path without requiring software simulation. However, this mechanism disables VM-exits by unloading operations related to root mode, posing potential security issues that may lead to incorrect handling or non-handling of virtual interrupts. Hu et al. proposed an efficient and responsive event system for I/O virtualization (ES2) that improves bidirectional I/O transmission between VMs and hardware devices, optimizing the I/O transmission path and enabling interrupt transmission without VM-exits. However, this method is unsuitable for SR-IOV devices, as interrupt handling in SR-IOV environments still involves the host, and frequent context switching when processing large numbers of virtual interrupt requests severely impacts system performance. Wang et al. proposed a multi-root I/O resource pooling method based on SR-IOV that enables multiple servers to share and reuse the same I/O device, reducing wiring redundancy in single servers and improving resource utilization efficiency. However, this method causes I/O device resource competition due to interrupts, and when numerous interrupt requests exist in the system, server I/O performance degrades severely.

1.2 Related Technology

The SR-IOV protocol is an extension of the PCIe bus interconnect protocol released by the PCI-SIG organization. It enables a single physical device to present itself as one physical function (PF) and multiple virtual functions through hardware virtualization of the I/O device itself. Each PF has standard PCIe capabilities and can be fully configured and managed. Users can configure or control PCIe devices through PF and manage data input and output. Compared to PF,

VF has lightweight PCIe capabilities and can perform data transmission. During data transmission, since each VF corresponds to a unique resource identifier (RID) and each RID can be used to index IOMMU page tables, each VF can independently receive and transmit data packets. Moreover, each VF possesses performance-related resources such as transmission and reception descriptors while sharing other major device resources, achieving efficient resource sharing.

Currently, Intel Corporation's DH895XCC Series QAT cryptographic device represents a representative hardware cryptographic device technology. The QAT cryptographic device leverages Intel QuickAssist Technology to enhance security and compression performance for dynamic and static data in cloud, network, big data, and storage applications. It accelerates computation-intensive operations, providing a software-based foundation for security, authentication, and compression, significantly improving the performance and efficiency of standard platform solutions. This device integrates symmetric encryption and authentication, asymmetric encryption, digital signatures, public key encryption, and DH technologies. Users can call corresponding cryptographic algorithms through appropriate interfaces to meet encryption requirements. To achieve high-performance cloud services, the QAT cryptographic device natively supports SR-IOV technology, enabling efficient resource sharing in cloud environments through SR-IOV implementation. The basic architecture of the QAT cryptographic device based on SR-IOV technology is shown in Figure 1 [Figure 1: see original paper]. In this architecture, VCM communicates directly with VF through the VFIO driver without VMM intervention.

2 Key Problem Analysis

To protect data security in cloud resource center server clusters, hardware cryptographic devices are used to provide corresponding cryptographic services to upper-layer applications. SR-IOV technology virtualizes a single hardware cryptographic device into multiple virtual cryptographic devices, with each virtual device assigned to a VM as a virtual cipher machine, thereby achieving efficient utilization of single hardware physical resources. Users obtain corresponding cryptographic services by renting VCMs. When a VCM processes cryptographic operation requests from users, task interrupts severely degrade the current cryptographic operation speed. Moreover, when numerous interrupt requests exist in the cloud environment, VCM performance will be severely degraded.

Figure 2 [Figure 2: see original paper] shows the test of VCM processing a single cryptographic task, illustrating how the AES128 algorithm encryption speed changes with different encryption byte block sizes. Experimental results demonstrate that the VCM encryption speed continuously increases until the encryption byte block reaches 16,384 bytes, at which point the VCM encryption speed reaches its maximum. When the encryption byte block continues to increase beyond this point, the VCM encryption speed begins to decrease. Throughout this process, the interrupt frequency continuously increases as the encryption byte block grows. This indicates that when the encryption byte block reaches

16,384 bytes, the SR-IOV device reaches its maximum capacity for processing interrupt requests and cryptographic operations. After this point, increasing interrupt frequency only leads to performance degradation because numerous interrupt requests consume substantial CPU resources, drastically reducing the time available for CPU to process cryptographic operations.

Figure 3 [Figure 3: see original paper] shows the test when a VCM simultaneously processes multiple identical cryptographic tasks, displaying the encryption speed changes for each task and CPU usage. Experimental results reveal that when a VCM processes two identical cryptographic tasks simultaneously, each task' s encryption speed is approximately half of that when processing a single task. When processing three identical tasks simultaneously, each task' s speed is about one-third of the single-task speed, and so on. This demonstrates that when a VCM processes multiple cryptographic tasks concurrently, CPU load balancing limitations cause each task to receive an average allocation of available CPU resources, resulting in severe speed degradation for each task. However, the sum of encryption speeds for multiple tasks is less than the speed of a single task, and the more simultaneous tasks a VCM processes, the greater the difference between the total speed and the single-task speed. This indirectly reflects the impact of interrupt frequency on VCM encryption performance: excessive interrupt requests from multiple tasks cause VCM encryption performance degradation.

Therefore, SR-IOV cryptographic devices still face a critical challenge: excessive interrupt request frequency leads to performance degradation. To address this challenge, this paper investigates the factors affecting VCM encryption performance, establishes a corresponding mathematical model, verifies theoretical analysis results through experiments, and analyzes and experimentally validates the differences among different encryption types.

3 Model Establishment and Optimization

Currently popular encryption and digital authentication algorithms adopt block encryption methods, which divide plaintext into fixed-size data blocks and execute cryptographic algorithms to obtain ciphertext. This paper uses the AES algorithm in cipher block chaining (CBC) mode as an example to analyze factors affecting SR-IOV cryptographic device encryption performance, using encryption speed as the performance reference standard to demonstrate that excessive interrupt request frequency degrades SR-IOV cryptographic device encryption performance.

3.1 Interrupt Frequency Control Model Establishment

Assume that within 1 second, the interrupt request frequency arriving at the SR-IOV cryptographic device is I, the time required for the SR-IOV cryptographic device to process each interrupt is t_0 (where t_0 is a constant), the total time spent on cryptographic operations is T ($T < 1$), the total encrypted bytes within

1 second are divided into n groups, and the time required to encrypt each group is t_1 (where t_1 is a constant). Then we have:

$$T = n \cdot t_1$$

$$I \cdot t_0 + n \cdot t_1 \leq 1$$

$$n = \frac{B}{L}$$

where B is the encryption data packet size, L is the block size (1024 bytes), and V is the QAT cryptographic device encryption speed (measured experimentally).

From equations (1)-(3), we obtain:

$$I \cdot t_0 + \frac{B}{L} \cdot t_1 \leq 1$$

$$I \leq \frac{1 - \frac{B}{L} \cdot t_1}{t_0}$$

When equality holds in equation (4), taking the derivative yields:

$$\frac{\partial I}{\partial B} = \frac{t_1}{L \cdot t_0}$$

Equation (5) shows that $\frac{\partial I}{\partial B} > 0$, meaning the SR-IOV cryptographic device' s interrupt request frequency I increases with encryption data packet size B. The general trend of I with respect to B is shown in Figure 4 [Figure 4: see original paper]. Correspondingly, the general trend of encryption data packet B with respect to I is shown in Figure 5 [Figure 5: see original paper].

From equations (1)-(3), the relationship between encryption speed V and encryption data packet B and interrupt frequency I is:

$$V = \frac{B}{1024 \cdot (I \cdot t_0 + \frac{B}{1024} \cdot t_1)}$$

When equality holds in equation (4), taking partial derivatives with respect to B and I yields:

$$\frac{\partial V}{\partial B} = \frac{1024 - I \cdot t_0 \cdot L}{(I \cdot t_0 \cdot L + B \cdot t_1)^2}$$

$$\frac{\partial V}{\partial I} = -\frac{B \cdot t_0 \cdot L}{(I \cdot t_0 \cdot L + B \cdot t_1)^2}$$

All symbols and their units are defined in Table 1 .

Table 1 Symbol definitions and units

Symbol	Meaning	Unit
I	Interrupt request frequency	Hz
t_0	Time to process each interrupt	s
t_1	Time to encrypt each data group	s
T	Total time for cryptographic operations	s
n	Total number of blocks	-
B	Encryption data packet size	bytes
V	QAT cryptographic device encryption speed	MB/s

From the above, $\frac{\partial V}{\partial B} > 0$ and $\frac{\partial V}{\partial I} < 0$. This means encryption speed V increases with encryption data packet B and decreases with interrupt frequency I. Since $V = 0$ when $B = 0$ (no interrupt requests), encryption speed V must have a maximum value $V_{\max}$.

When encryption speed reaches its maximum, the corresponding encryption data packet B and interrupt frequency I reach ideal values. Before this ideal point, B and I are positively correlated; after this point, they are negatively correlated. This ideal value represents the optimal processing capacity of the SR-IOV cryptographic device. Denoting the ideal encryption data packet size as B_0 and the corresponding ideal interrupt frequency as I_0, we have:

$$V_{\max} = \frac{B_0}{1024 \cdot (I_0 \cdot t_0 + \frac{B_0}{1024} \cdot t_1)}$$

Considering the extreme case where B equals 0, the stationary point is $I_0 = \frac{1}{t_0}$. At this point, encryption speed is 0, and the interrupt frequency represents the maximum interrupt requests the SR-IOV device can process per second. Since the excessively large encryption data packet B leads to excessive interrupt frequency, the SR-IOV cryptographic device spends all its time processing interrupts and loses its cryptographic computation capability. However, this phenomenon does not occur in experiments because SR-IOV cryptographic devices' virtual functions support setting an interrupt frequency upper limit. To pursue higher encryption speed, the interrupt frequency upper limit should theoretically equal the ideal value I_0, which must be determined through experimental testing.

Theoretical analysis concludes that when encryption data packet B exceeds the ideal value B_0, interrupt request frequency I continues to increase, but encryption speed V decreases because the device has exceeded its cryptographic computation capacity. The general trends among encryption speed V, encryption data packet B, and interrupt frequency I are shown in Figure 6 [Figure 6: see original paper].

Therefore, it is necessary to control the size of encryption data packets processed by interrupt requests, thereby controlling the interrupt request frequency arriving at the SR-IOV device, maximizing SR-IOV device cryptographic operation performance (i.e., maximizing encryption speed) to provide users with more efficient cryptographic services.

3.2 Interrupt Frequency Optimization Method

The above analysis shows that when encryption data packet B reaches the ideal value B_0, SR-IOV cryptographic device encryption speed is maximized, and the corresponding interrupt request frequency reaches the ideal value I_0. Therefore, controlling the size of encryption data packets received by the SR-IOV cryptographic device can maintain encryption speed at a high level. The preliminary design is: when the received encryption data packet size is less than the ideal value, no changes are made; when the received encryption data packet size exceeds the ideal value, the received encryption data packet is first partitioned into blocks using the ideal value as the partitioning baseline, enabling each block to achieve maximum encryption speed. The model for controlling encryption data packets received by the SR-IOV cryptographic device is:

$$B_0 = \begin{cases} B, & B < B_0 \\ \left\lceil \frac{B}{B_0} \right\rceil, & B \geq B_0 \end{cases}$$

where $\left\lceil \frac{B}{B_0} \right\rceil$ is an integer (rounded up if decimal), representing the number of encryption data packet partitions. Correspondingly, encryption speed can be expressed as:

$$V = \begin{cases} \frac{B}{1024 \cdot (I \cdot t_0 + \frac{B}{1024} \cdot t_1)}, & B < B_0 \\ \frac{B_0}{1024 \cdot (I_0 \cdot t_0 + \frac{B_0}{1024} \cdot t_1)}, & B \geq B_0 \end{cases}$$

Since presetting the VF interrupt frequency upper limit has limitations—requiring experimental testing to obtain data for finding the optimal interrupt frequency—and considering that different cryptographic algorithms have different complexities and CPU clock cycle consumption, the interrupt frequency corresponding to maximum encryption speed varies by algorithm, making this method non-universal. To enable automatic threshold adjustment based on encryption speed changes, this paper proposes a control scheme that automatically

adjusts the VF interrupt frequency upper limit by monitoring encryption speed changes. The architecture of this scheme is shown in Figure 7 [Figure 7: see original paper].

This scheme incorporates a speed monitoring module in the VF driver layer to monitor encryption speed changes in real time. When the monitoring module detects a decreasing trend in encryption speed (i.e., the next second's encryption speed is lower than the previous second's), it automatically sets the VF interrupt frequency upper limit to the previous second' s interrupt frequency through the interrupt frequency control algorithm, with no further changes thereafter. The initial VF interrupt frequency upper limit is generally set to the ideal value I_0, because in practice, the CPU cannot dedicate all its time to processing interrupt requests, and the optimal interrupt frequency obtained in actual experiments will never exceed the ideal value I_0. Throughout the monitoring process, the speed monitoring module adjusts the VF interrupt frequency upper limit only once, and the adjusted upper limit must be less than the initial value.

The interrupt frequency control algorithm implemented in the speed monitoring module is as follows:

Algorithm: Interrupt Rate Control

```
Initialization: i = 0, j = 0, k = 0, t = 0
Interrupt Rate Handling:
    get the speed of the encryption V_i
    j ← V_i
    if j > i then
        t ← t + 1
        i ← j
    else if j < i then
        get current Interrupt Rate(I_k)
        I_0 ← I_k
    end if
```

By incorporating this interrupt frequency control algorithm into the speed monitoring module, the VF driver can monitor VCM encryption speed changes in real time when processing encryption use cases. When the monitoring module detects that VCM encryption speed has reached its maximum, it automatically adjusts the VF interrupt frequency upper limit to the interrupt frequency at maximum VCM encryption speed (i.e., the optimal interrupt frequency). This ensures that the CPU clock cycles occupied by interrupt processing do not continue to increase and that the CPU can process encryption data packets received by the VCM with maximum capability, maintaining VCM encryption speed at the highest level and fully improving VCM encryption performance.

4 Encryption Performance Testing

To demonstrate VCM encryption performance improvement after introducing the interrupt frequency control algorithm, the experiment adds a speed monitoring module to each VCM' s VF driver, using encryption speed magnitude to evaluate VCM encryption performance. The evaluation criterion is: higher encryption speed indicates better VCM encryption performance, and vice versa. The experimental environment consists of: host and VCM operating systems both running CentOS-7-x86_64-1511 with kernel version Linux 3.10.0-327.e17.x86_64, processor Intel(R) Xeon(R) CPU E5-2620 v3 @2.40 GHz, 12 CPU cores, 128 GB memory, VT-x and SR-IOV support. Each VCM is allocated 1 vCPU and 2048 MB memory, and the cryptographic device is the Intel Corporation DH895XCC Series QAT.

This chapter selects symmetric encryption algorithm AES128, hash algorithm SHA256, and public key algorithm RSA to represent different cryptographic algorithm types, investigating differences in their encryption performance improvements. The testing tool is cryptodev, a benchmark tool for invoking cryptographic devices that can test operation speeds of cryptographic algorithms including AES128, SHA256, and RSA.

4.1 AES128 Encryption Performance Testing

As described in Section 2.2, when the encryption byte block that a VCM needs to process increases, the interrupt request frequency also increases, but encryption speed begins to decrease after reaching a certain value. Chapter 3's analysis indicates that excessive interrupt request frequency consumes more CPU clock cycles, severely reducing CPU clock cycles available for cryptographic operations and thereby degrading VCM encryption speed. After adding the speed monitoring module to the VCM VF driver layer, AES128 encryption speed and interrupt frequency changes are shown in Figure 8 [Figure 8: see original paper]. The results show that when the encryption byte block reaches 16,384 bytes, AES128 encryption speed reaches its maximum, with an optimal interrupt frequency of approximately 6,450 Hz. After adding the speed monitoring module, the interrupt frequency no longer increases, and the corresponding AES128 encryption speed only slightly decreases by 0.1 GB at a 32,768-byte block before returning to maximum speed operation. This occurs because the speed monitoring module consumes some CPU resources when adjusting the VF interrupt frequency upper limit. After the interrupt frequency upper limit adjustment completes, the VCM maintains maximum encryption speed operation.

4.2 SHA256 Encryption Performance Testing

Unlike AES encryption, SHA256 is a hash algorithm with higher computational complexity. When a VCM encrypts using the SHA256 algorithm, its encryption speed trend is similar to AES128, but both the maximum encryption speed and optimal interrupt frequency differ. SHA256 algorithm encryption speed

changes and interrupt frequency variations are shown in Figure 9 [Figure 9: see original paper]. The results indicate that when the encryption byte block is 16,384 bytes, SHA256 algorithm encryption speed reaches its maximum, with an optimal interrupt frequency of approximately 5,109 Hz. After adding the speed monitoring module, the interrupt frequency no longer increases, and the VCM' s SHA256 algorithm encryption speed remains at 177 MB/s.

4.3 RSA Encryption Performance Testing

For the RSA public key cryptographic algorithm, its hardware implementation speed is relatively slow—approximately 1,000 times slower than DES symmetric algorithm hardware implementation. Experiments testing data blocks of 512, 1,024, 2,048, and 4,096 bytes show that larger bytes result in slower encryption speeds, while corresponding interrupt frequencies are not high. After adding the speed monitoring module, although RSA encryption speed improves somewhat, the improvement is not significant, as shown in Figure 10 [Figure 10: see original paper]. These results indicate that for the RSA public key algorithm, which has high complexity and belongs to computation-intensive operations, interrupt frequency optimization primarily targets I/O transmission optimization and does not significantly benefit computation-intensive operations. In contrast, for symmetric algorithm AES128 and hash algorithm SHA256, which have much lower computational complexity, I/O transmission optimization effects are more pronounced.

5 Conclusion

To address the performance bottleneck of SR-IOV cryptographic devices, this paper investigates the factor affecting SR-IOV cryptographic device performance—excessive interrupt request frequency causing performance degradation—and establishes a corresponding interrupt frequency control model. Theoretical analysis examines the relationship among interrupt frequency, encryption byte blocks, and encryption speed, and proposes an interrupt frequency optimization method based on this analysis. This method adds a speed monitoring module to the VF driver layer with an interrupt frequency control algorithm that monitors VCM encryption speed changes in real time. When the speed monitoring module detects decreasing VCM encryption speed, it automatically adjusts the VF interrupt frequency upper limit to the previous second' s interrupt frequency, thereby controlling interrupt request frequency received by the VCM and enabling the CPU to process cryptographic operations with maximum capability, fully improving VCM encryption performance. This paper experimentally validates the conclusions using symmetric cryptographic algorithm AES128, hash algorithm SHA256, and public key cryptographic algorithm RSA. Experimental results demonstrate that for I/O-intensive cryptographic processes (symmetric and hash algorithms), adding the speed monitoring module to the VF driver layer fully exploits VCM encryption performance. For computation-intensive cryptographic processes (RSA algorithm), performance improvement is not sig-

nificant. Therefore, this scheme effectively promotes I/O transmission optimization and substantially improves VCM encryption performance.

References

[1] Lin Chuang, Su Wenbo, Meng Kun, et al. Cloud computing security: architecture, mechanism and model evaluation [J]. Journal of Computers, 2013, 36(9): 1765-1784.

[2] Zhang Yuqing, Wang Xiaofei, Liu Xuefeng, et al. An overview of cloud computing environment security [J]. Journal of Software, 2016, 27(6): 1328-1348.

[3] Alibaba Cloud [EB/OL]. https://www.aliyun.com/product/hsm?spm=5176.8037491.395145.117.438723976J

[4] Koppel B, Neuhaus S. Analysis of a hardware security module' s high-availability setting [J]. IEEE Security & Privacy, 2013, 11(3): 77-80.

[5] Sansec Technology [EB/OL]. http://ec.com.cn/html/2014/1016/17.html.

[6] Dong Yaozu, Yang Xiaowei, Li Jianhui, et al. High performance network virtualization with SR-IOV [J]. Journal of Parallel and Distributed Computing, 2012, 72(11): 1471-1480.

[7] Akkinapalli K, Rao R R. A survey on encryption and improved virtualization security techniques for cloud infrastructure [J]. Global Journal of Computer Science & Technology, 2014, 14(2).

[8] Wang Guohui, Ng T S E. The impact of virtualization on network performance of Amazon EC2 data center [C]//Proc of the 29th Conference on Information Communications. Piscataway, NJ: IEEE Press, 2010.

[9] Xu Xin, B. Davda. A hypervisor approach to enable live migration with passthrough SR-IOV network devices [J]. ACM Sigops Operating Systems Review, 2017, 51(1): 15-23.

[10] Musleh M, Pai V, Walters J P, et al. Bridging the virtualization performance gap for HPC using SR-IOV for infiniBand [C]//Proc of IEEE International Conference on Cloud Computing. Washington DC: IEEE Computer Society, 2014: 627-635.

[11] Younge A J, Walters J P, Crago S P, et al. Supporting high performance molecular dynamics in virtualized clusters using IOMMU, SR-IOV, and GPUDirect [C]//Proc of ACM Sigplan//Sigops International Conference on Virtual Execution Environments. 2015: 31-38.

[12] Intel(R) quickAssist (QAT) crypto poll mode driver [EB/OL]. http://dpdk.org/doc/guides/cryptodevs/qat.

[13] Li Jian, Xue Shuai, Zhang Wang, et al. When I/O interrupt becomes system bottleneck: efficiency and scalability enhancement for SR-IOV network virtualization [J]. IEEE Trans on Cloud Computing, 2017, PP(99).

[14] Dong Yaozu, Yang Xiaowei, Li Xiaoyong, et al. High performance network virtualization with SR-IOV [C]//Proc of IEEE International Symposium on High PERFORMANCE Computer Architecture. 2010: 1471-1480.

[15] Dong Yaozu, Xu Dongxiao, Zhang Yang, et al. Optimizing network I/O virtualization with efficient interrupt coalescing and virtual receive side scaling [C]//Proc of IEEE International Conference on CLUSTER Computing. Washington DC: IEEE Computer Society, 2011: 26-34.

[16] Gordon A, Amit N, Har' El N, et al. ELI: bare-metal performance for I/O virtualization [C]//Proc of Architectural Support for Programming Languages & Operating Systems. 2012: 411-422.

[17] Guan Haibing, Dong Yaozu, Kun Tian, et al. SR-IOV based network interrupt-free virtualization with event based polling [J]. IEEE Journal on Selected Areas in Communications, 2013, 31(12): 2596-2609.

[18] Tu Chengchun, Ferdman M, Lee C T, et al. A comprehensive implementation and evaluation of direct interrupt delivery [C]//Proc of ACM Sigplan//Sigops International Conference on Virtual Execution Environments. 2015: 1-15.

[19] Hu Xiaokang, Zhang Wang, Li Jian, et al. ES2: Aiming at an optimal virtual I/O event path [C]//Proc of IEEE International Conference on Parallel Processing. 2017: 141-150.

[20] Wang Zhan, Cao Zheng, Liu Xiaoli, et al. A multiple I/O resource pooling method based on single I/O virtualization [J]. Journal of Computer Research and Development, 2015, 52(1): 83-93.

[21] Richter A, Herber C, Wallentowitz S, et al. A hardware//software approach for mitigating performance interference effects in virtualized environments using SR-IOV [C]//Proc of the 8th IEEE International Conference on Cloud Computing. 2015: 950-957.

[22] Intel® QuickAssist Technology [EB/OL]. https://www.intel.cn/content/www/cn/zh/architecture-and-technology/intel-quick-assist-technology-overview.html.

[23] Frankel S, Glenn R, Kelly S. The AES-CBC cipher algorithm and its use with IPSec [S]. Ietf RFC, 2003.

Note: Figure translations are in progress. See original paper for figures.