

QAT: Evaluation of a Dedicated Hardware Accelerator for High Performance Web Service

Xue Shuai*, Liu Yao*, Zhang Wang*

*Shanghai Key Lab of Scalable Computing and System, Shanghai Jiao Tong University, China

mattxue_sjtu@sjtu.edu.cn, liuyao@sjtu.edu.cn, Miwa1997@sjtu.edu.cn

Abstract—Security is increasingly becoming a vital necessity for web service. Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols that are used to provide confidentiality and data integrity between two applications. SSL/TLS has been widely integrated into web service applications. However, the heavy overhead that is brought by frequent encryption and decryption operations in SSL/TLS has terribly burden the CPU. To address this issue, various vendors and researchers focused on developing new alternatives, such as GPU-based and FPGA-based accelerators. Unfortunately, expensive development cost and time cost seriously restrict their deployment in web servers. In this paper, we present and evaluate an emerging dedicated hardware accelerator named Quick Assist Technology (QAT). We analyse and explain the QAT's asynchronous working mode and how it cooperates with OpenSSL and Nginx server. The experimental results show that the dedicated hardware accelerator can speed up the number of connections by 10x comparing to the traditional software ways.

Keywords—Cryptography, Hardware Accelerator, Nginx, QAT, Web Service

I. INTRODUCTION

As the disclosure of personal privacy, such as bank accounts, identity information, seriously put a threat to user's personal and property safety, security is gradually becoming a fundamental necessity in web service. A growing number of web-based service providers eager to ensure the privacy while providing normal service to their users. Therefore, SSL/TLS has been widely adopted and integrated into their service.

SSL/TLS aims to provide privacy and data integrity between two applications. Nowadays, SSL/TLS is used in every personal computer or smartphone's browser worldwide to provide HTTPS functionality. OpenSSL has implemented these two protocols and open-sourced to its users and developers. However, CPU is heavily occupied and frequently processes complex cryptographic operations, extremely affects server's performance and users' experience.

To address this issue, various alternatives have been proposed and optimized to improve the poor performance that incurs by SSL/TLS. Among these alternatives, QAT is a hardware-assisted alternative that dramatically accelerates the cryptographic operations and mitigates CPU's burden. QAT works as a co-processor and is inserted into a PCIe x8 slot. QAT also provides compression and decompression functionalities to save storage in data centre.

In this paper, we present and explain the overall architecture of Nginx web server that using QAT to accelerate the HTTPS connections. Then, we first evaluate the performance of 1024-bit RSA encryption and 2048-bit RSA encryption. We then evaluate the performance of HTTPS connections that QAT can bring comparing to the software way. The Nginx's performance that gains by CPU is taken as a baseline, the synchronous and asynchronous modes that QAT supports are evaluated and presented to reveal the performance improvements using QAT accelerator.

The paper is organized as follows. Section II introduces the related works about hardware accelerators. Section III presents the detailed architecture of Nginx web server that integrated with QAT accelerator. Section IV evaluates the system. Finally, Section V concludes this paper.

II. RELATED WORK

A. FPGA-Based Accelerators

Thiruneelakandan [1] proposed an FPGA-based VLSI encryption system that accelerates the encryption algorithms used in the SSL/TLS protocols. The SSLv3 and TLSv1 protocols have been implemented in the proposed system using Nios-2 soft-core processors. These algorithms include Extensible Encryption Algorithm (SEA), Message Digest Algorithm (MD5) and Secure Hash Algorithm (SHA2). The experimental results show that these algorithms are somewhat accelerated in the VLSI encryption system on the Altera Cyclone III FPGA DE2 development board. Mohamed [2] also implemented these algorithms and deployed them onto Nios-2 soft-core processors. The results show that AES-128 is sped up by 9x, AES-256 is 22x faster than the CPU ways, SHA256 can be accelerated by 22x, RSA2048 private key encryption is sped up by 57x. Nazrulislammondal [3] proposed a method based on FDFM (a small amount of DSP slices and a few memory blocks) that supports flexible multi-bit arithmetic and implemented in FPGAs. In order to show the potential of the designed processor, 128-bit RSA encryption/decryption is implemented and compared to the soft processor "MicroBlaze" in the FPGA. The resulting processor uses only one DSP48E1 slice and four block RAMs (BRAMs), with RSA encryption running at 0.42ms on it. However, MicroBlaze uses three DSP48E1 slices and 170 Block RAMs (BRAMs) running at 152.28ms. The main contribution of Nakano et al. [4] is to propose an efficient modular exponentiation algorithm for RSA encryption and

decryption that implemented on FPGA board. The key idea behind the exponential modular exponentiation is to use Montgomery modular multiplication on the FPGA's redundant base 64K digital system and to make efficient use of the embedded 18*18-bit multipliers and embedded 18k-bit block RAM. The proposed modular exponentiation algorithm has been implemented on the Xilinx VirtexII Pro family of FPGAs, the XC2VP30-6. The experimental results show that the 1024-bit modular exponentiation algorithm can run within 2.521ms, expected to be 1.892ms.

B. GPU-Based Accelerators

Scott et al. [5] adopted GPUs to accelerate AES-CBC, a cryptographic operation in database applications, to replace the traditional CPU encryption and decryption operations. In database applications, encryption is increasingly used to protect sensitive data. However, encrypting and decrypting data during query processing can incur significant overhead and reduce the overall system throughput. The authors used low-cost GPUs as cryptographic co-processors, dramatically reducing the time it takes to free up CPUs to other tasks. The results show that the GPU can achieve 3~14x faster improvements comparing to the CPU implementations. Saraiva [6] used the high-performance computing capabilities of OpenCL and CUDA to speed up RSA and AES algorithms. The authors provided an engine that allows the existing applications to bypass rewriting cryptographic libraries, thus reduce the CPU load on the most commonly used algorithms. CBC decryption can be sped up by 43x, while ECB encryption can be accelerated by 31x. Hsiu-Pang et al. [7] used CUDA's GPU to accelerate DES and 3-DES. The authors chose the widely used 3-DES and implemented it on the GPU. Experimental results show that GPU encryption is 5x faster than that on the CPU. Johannes et al. [8] used CUDA and OpenCL to implement standard algorithms that integrated in OpenSSL libraries, such as AES, DES, Blowfish, IDEA, Camellia, and CAST5. In a real-world scenario, the author's GPU implementation can accelerate all block ciphers by a factor of 10 (compared to CPU implementations). The authors also implemented the OpenSSL engines based on the "engine-cuda" project. Although the original "engine-cuda" project included CUDA's powerful AES implementation, the authors' improved version of AES almost doubled.

C. Related Accelerators Researches

Changzheng et al. [9] proposed a three-phase hierarchical key management scheme called STYX to provide secure key protection and hardware-assisted service acceleration for cloud-based content delivery network (CCDN) applications. STYX is based on Intel Software Guard Extensions (SGX), Intel QuickAssist Technology (QAT) and SIGMA (SIGN-and-MAC) protocols. STYX provides tight key security for SGX-based key distribution at a lightweight overhead and can further significantly improve QAT-accelerated system performance. A comprehensive result shows that STYX not only guarantees absolute security, but also delivers up to five times the throughput of CDN comparing to the direct HTTPS server deployments without QAT, while with significant latency reductions.

III. SYSTEM ARCHITECTURE

A. QAT's Modes of Operations

There are two kinds of modes of operations that the QAT can support, the synchronous mode and asynchronous mode. In synchronous mode, the application, such as Nginx server, needs to wait for completion once it submits the request to QAT, while in asynchronous mode, the submitted request can be returned to the application once it completes rather keeping waiting.

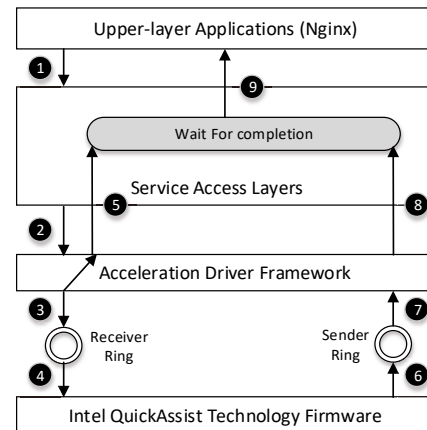


Figure 1. Synchronous operation

As shown in Figure 1, the synchronous operation will keep in waiting in step 5 until the operation is complete in step 8. After submitting a request to the QAT engine, the calling thread will be blocked or locked. Upon the completion of the request, the blocked thread will be unlocked and resume submitting more requests.

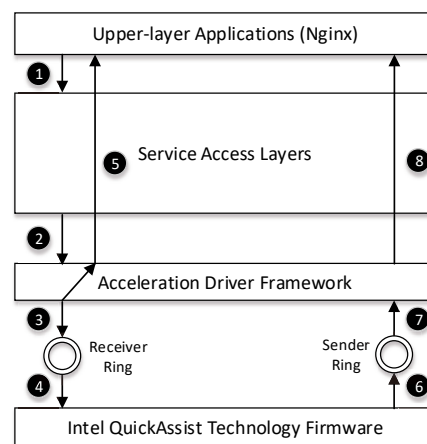


Figure 2. Asynchronous operation

Figure 2 shows the whole processing procedures of the asynchronous operation. In order to submit requests asynchronously, the application should register a callback function to continue processing the next steps once the request finishes. In QAT's implementation, it supports two types of callback functions, the interrupt mode and the polling mode. In interrupt mode, the application can be notified through interrupt for each completion of a request, this might produce excessive

interrupts when the number of requests is considerably large. In polling mode, a polling thread will periodically check for completion of a request. However, this will cause resource waste if there are sparse number of requests.

B. The Overall Architecture

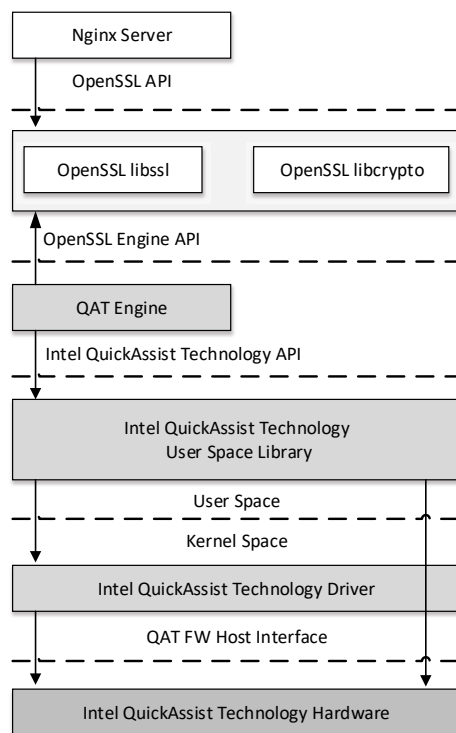


Figure 3. The overall architecture

In Figure 3, the overall fundamental components consist of QAT hardware accelerator, QAT kernel driver, QAT user space library, QAT engine, OpenSSL libraries and Nginx server. The QAT driver, that interacts with QAT hardware device, runs in kernel space. The QAT user space library is exposed under user space in order to be called by upper layer applications. It is worth mentioning that the OpenSSL supports the “engine” feature that allows users to replace the default engine with self-designed software or hardware engines. Therefore, the QAT engine that combines the OpenSSL libraries and QAT user space APIs can dramatically accelerate the performance of Nginx HTTPS connections. The Nginx server relies on OpenSSL libraries to apply the security features in HTTP connections.

As we mentioned above, the asynchronous mode can substantially improve the time in processing request. In OpenSSL-1.1.0 version, an ASYNC_JOB mechanism is introduced to enable the cryptographic operations to be performed asynchronously. As shown in Figure 3, the QAT engine can offload the cryptographic operations onto QAT device thus frees up the CPU to handle other tasks. Once the engine has been initialized, all encryption operations that are registered and executed through The EVP API will be transparently offloaded onto QAT engine. With the help of QAT accelerator, the processing time of cryptographic

operations can be significantly reduced and the performance can be dramatically enhanced.

IV. EXPERIMENTAL RESULTS

In this section, we evaluate a representative algorithm, RSA, that is an asymmetric algorithm which is widely used in security protection. The 1024-bit and 2048-bit RSA algorithms are chosen to be tested as they are currently most frequently used. We then evaluate the Requests Per Second (RPS) performance of Nginx server using Apache Bench to prove the performance improvements that QAT accelerator achieves.

A. Experimental Setup

The two identical servers are equipped with two 22-cores Intel(R) Xeon E5-2699 CPUs, 64GB RAM, connected with 40GbE Ethernet Controllers. The server is equipped with an Intel QAT DH895XCC Series PCIe acceleration card. The Apache Bench benchmark that resides in client concurrently sends 1000 requests for each connection to server, each test is configured with 100 connections. We select OpenSSL version 1.1.0f and Nginx version 1.10.0, Apache Bench version 2.4. In order to measure the performance of RPS of HTTPS requests, the requested file size is fixed with 0K size.

B. RSA Performance

Figure 4 shows the performance of 1024-bit RSA encryption using OpenSSL speed benchmark. “Software” represents the whole procedures for processing a request are dealt by CPU. “SYNC” stands for synchronously submitting requests using QAT accelerator. “ASYNC” represents an asynchronous manner. As shown in Figure 4, the “ASYNC” can outperform the “SYNC” by 9.20~10.37x. The performance of the “SYNC” is nearly the same to that of the “Software”. When there are 4 cores, the Connections Per Second (CPS) of the “ASYNC” can achieve the throughput of 183.98K, while the throughput of the “SYNC” and the “Software” only reach 20.0K and 20.13K, respectively.

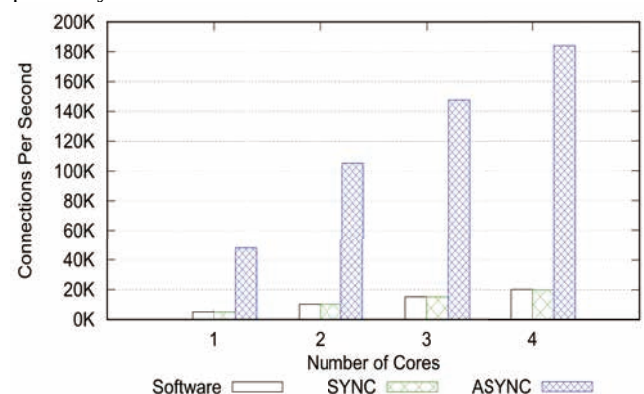


Figure 4. 1024-bit RSA encryption

The performance of 2048-bit RSA encryption is shown in Figure 5. The throughput of the “ASYNC” is approximately 40K RPS, while the throughput of the “SYNC” and the “Software” can merely get 8.50K and 8.51K RPS at best, respectively. The throughput of the “ASYNC” is 4.79~10.66x higher than that of the “SYNC”. Notice that the throughput of the “ASYNC” cannot increase as the increase of the number of

cores. When there is one core, the throughput of the “ASYNC” is 23.3K RPS, nearly half of the throughput of two cores. The reason is that the QAT reaches its upper bound limitation of processing requests, thus it can no longer process more requests.

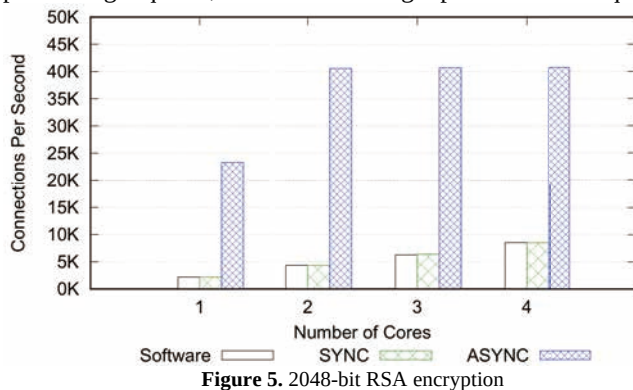


Figure 5. 2048-bit RSA encryption

C. Application-level Performance

Figure 6 presents the Nginx’s performance improvement with the help of QAT accelerator using Apache Bench benchmark. It is noted that the throughput of the “SYNC” is surprisingly lower than that of the “Software”, the “Software” is about 1.90~3.12x better than that of the “SYNC”. The reason is that overhead of the “SYNC” from Nginx application to QAT hardware device is much lengthier comparing to the “Software” way. In addition, the “SYNC” way is originally not better than the “Software” way. The throughput of the “ASYNC” can outperform that of the “Software” by 3.52~3.91x. When there are 4 Nginx instances, the throughput of the “ASYNC”, the “SYNC” and the “Software” can reach 10.48K, 2.83K and 1.20K RPS, respectively.

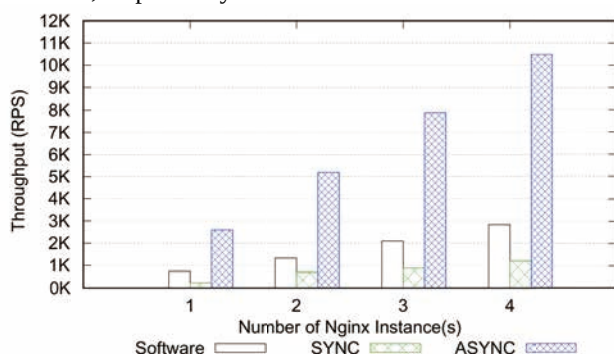


Figure 6. Nginx’s performance using Apache benchmark

V. CONCLUSIONS

In this paper, we presented and evaluated an emerging hardware accelerator called QAT. There are two modes of operations that the QAT can support, the synchronous operation and the asynchronous operation. In order to maximize the performance of the QAT accelerator, we have to elaborately arrange the Nginx server, the OpenSSL libraries, the QAT engine and the QAT driver. The experimental results show that the 1024-bit RSA encryption can be accelerated by 10x in average, the 2048-bit RSA encryption can be sped up by 10.66x at best, and the throughput of the Nginx’ HTTPS connections can be improved nearly by 4x.

REFERENCES

- [1] Thiruneelakandan, A., & Thirumurugan, T. (2012). An approach towards improved cyber security by hardware acceleration of OpenSSL cryptographic functions. *International Conference on Electronics, Communication and Computing Technologies (Vol.2, pp.13-16)*. IEEE.
- [2] Mohamed Khalil-Hani, Vishnu P. Nambiar, and M. N. Marsono. 2010. Hardware Acceleration of OpenSSL Cryptographic Functions for High-Performance Internet Security. In *Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation (ISMS '10)*. IEEE Computer Society, Washington, DC, USA, 374-379. DOI=http://dx.doi.org/10.1109/ISMS.2010.89
- [3] Nazrulislamondal M, Al Mamun M, Ahmed B. RSA Cryptography using our Designed Processor and MicroBlaze Soft Processor in FPGAs[J]. *International Journal of Computer Applications*, 2014, 104(8):25-31.
- [4] Nakano K, Kawakami K, Shigemoto K. RSA encryption and decryption using the redundant number system on the FPGA[C]// *IEEE International Symposium on Parallel & Distributed Processing*. IEEE, 2009:1-8.
- [5] Scott Fazackerley, Steven M. McAvoy, and Ramon Lawrence. 2012. GPU accelerated AES-CBC for database applications. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing(SAC '12)*. ACM, New York, NY, USA, 873-878. DOI=http://dx.doi.org/10.1145/2245276.2245446
- [6] Saraiva, Pedro Miguel Costa. "OpenSSL acceleration using Graphics Processing Units." (2014).
- [7] Hsiu-Pang Yeh, Yue-Shan Chang, Chia-Feng Lin, and Shyan-Ming Yuan. 2011. Accelerating 3-DES Performance Using GPU. In *Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CYBERC '11)*. IEEE Computer Society, Washington, DC, USA, 250-256. DOI=http://dx.doi.org/10.1109/CyberC.2011.49
- [8] Johannes Gilger, Johannes Barnickel, and Ulrike Meyer. 2012. GPU-Acceleration of block ciphers in the OpenSSL cryptographic library. In *Proceedings of the 15th international conference on Information Security (ISC'12)*, Dieter Gollmann and Felix C. Freiling (Eds.). Springer-Verlag, Berlin, Heidelberg, 338-353. DOI=http://dx.doi.org/10.1007/978-3-642-33383-5_21
- [9] Wei, Changzheng & Li, Jian & Li, Weigang & Yu, Ping & Guan, Haibing. (2017). STYX: a trusted and accelerated hierarchical SSL key management and distribution system for cloud based CDN application. 201-213. 10.1145/3127479.3127482.



Xue Shuai received the B.S. degree in software engineering from Huazhong University of Science and Technology in 2011. He currently pursues the M.S. degree in the school of software engineering in Shanghai Jiao Tong University, Shanghai, China. His research interests are mainly focused on hardware-assisted virtualization, cloud computing, hardware acceleration, Internet of things, and distributed in-memory databases.



Liu Yao received the B.S. degree in software engineering from Southeast University in 2011. She is now studying in the school of computer science and technology in Shanghai Jiao Tong University, Shanghai, China. Her research interests are mainly focused on encrypted databases, cloud computing, hardware acceleration, high availability.



Zhang Wang received the B.S. degree in software engineering from Sichuan University in 2011. She is currently studying in the school of software engineering in Shanghai Jiao Tong University, Shanghai, China. Her research interests include virtualization, Interrupt optimization.