2012 International Conference on Computer Technology and Science (ICCTS 2012)
IPCSIT vol. 47 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V47.14

Measurement of Performance benefits using Hardware Crypto Acceleration

Rajesh Meena [+]

Performance Engineering Research Center, Tata Consultancy Services Ltd., Mumbai, India

Abstract. This paper describes the advantages of hardware cryptography acceleration. Intel has introduced a set of new instructions, Intel Advanced Encryption Standard New Instructions (AES-NI) in its E5600 CPU series, which offloads the additional overhead of encryption and decryption on the hardware. These instructions are designed to perform some complex steps involved in the Advanced Encryption Standard (AES) algorithm using hardware, thereby improving the overall execution of the AES algorithm. It is important to understand how performance benefits can a standard web application effect from this feature. In the course of testing the author found that it is not straight forward to do this measurement given the available resources for test. This paper describes, how certain challenges are overcome in order to derive the desired metric.

Keywords: cryptography, AES, AES-NI, Westmere-EP, Intel, performance, evaluation, measurement, J2EE.

1. Introduction

With the increasing use of e-commerce, the rate of occurrence of a security threat or breach has also increased. It is necessary to secure the online interaction between clients and servers with the latest cryptographic solution, in order to ensure the safety and security of the interactions. There are a number of key-based cryptography standards [4], [5] available in the market. AES is a new cryptography standard which has so far proven to be more secure as compared to the other standards [1], [2]. It is a specification for the encryption of electronic data. It uses a symmetric-key algorithm in which the same key is used to encrypt and decrypt the data. AES requires more processing as compared to the other cryptography standards like DES [4] and Blowfish [6]. The process of encryption involves a series of complex computational steps implemented in software, increasing the workload on the CPU and resulting in a dip in application performance. Application users may thus experience longer response times when AES is used to secure the communication channels.

In this paper, the throughput and CPU utilization of a J2EE web application [7] using the AES algorithm for secure client-server communication is used as a benchmark to evaluate the advantages of using the AES-NI feature. A comparison is done between the software and hardware crypto implementation and the results are graphically illustrated. The Intel Xeon E5600 (Westmere-EP) series of CPU has the AES-NI feature to perform hardware encryption and decryption [3], [12]. The instructions are designed to implement some of the complex and CPU intensive steps involved in the AES algorithm using the underlying hardware in order to accelerate the execution of AES algorithm. A load test was done by gradually increasing the number of users. Initially there was no performance improvement on using the AES-NI feature. With 8 CPU cores available, the application was not able to utilize the CPUs to their maximum capacity. Due to the limitation of hardware resources required to generate more user load, it was decided to assign the application to a single CPU and then test whether the utilization and throughput would improve on using the AES-NI feature. After

[+] Rajesh Meena.
E-mail address: rajesh.meena@tcs.com

limiting the application to use only one core, the utilization for core increases and a noticeable improvement in performance is observed.

2. Problem Statement

The complex computational steps of the AES algorithm are implemented entirely in software. It requires additional computational power to perform the algorithm steps, thus increasing the utilization of the processor. It limits the average throughput of the application. Our main objective is to see whether there is an improvement in the performance of the web application using the AES-NI feature.

3. Setup

A J2EE Pet Store web application [7] is used to perform the evaluation. The underlying hardware is Intel Xeon E5620 2 Quad-Core CPUs running at 2.4GHz on x86_64 platform with 16GB RAM. Westmere-EP is the codename of the server. Local storage of 300 GB SAS was present. This load test is performed on a 1 Gbps LAN. Centos 6.0 is the operating system built in with JDK version v1.6.0_23, Apache tomcat server v6.0.33 and NSS v3.12.10. A sequence of 13 web pages are accessed concurrently by users, which involves login, choosing the pet to buy from an available list, updating the cart, payment details and checking out the order. The database used by the application was Derby.

4. Methodology

In a typical JDK installation on Linux, the Java runtime environment is not configured to make use of hardware crypto accelerator. To configure java in such a way that it identifies Cryptographic Token Interface Standard (PKCS#11) [10] as the default provider, the Java security properties file located in $JAVA_HOME/jre/lib/security/java.security needs to be configured to use native PKCS#11 as the default provider. Network Security Services (NSS) [9], [10] is a set of open source security libraries; it fills the bridge between native PKCS#11 implementation and Java crypto APIs.

The implementation of the evaluation is done by compiling the NSS libraries for x86-64 architecture and integrates it with Java by copying the library files (.so) to the JDK library folder. The updated JDK is ready to use the AES-NI instructions to execute the AES algorithm steps. The load testing is done using an internal tool FASTEST. In order to see the advantage of AES-NI feature and not to load test the application, the load test is done up to 200 users. Initially the load test is done by allowing the application to use all eight cores on the server. After running the initial load test, it is observed that the application is not able to throttle all eight cores of the CPU. The measured utilization was 25-30%. Due to the low utilization of the CPU, there was no improvement seen in the throughput and response time. Since it was infeasible to further increase the user load on the server, given the limitation in hardware resources, the number of CPU cores used by the application was reduced to one by using the taskset command.

5. Evaluation of AES-NI Feature

Figure 1 show the utilization of web application server, when all the eight cores are enabled, for a maximum of 200 users the utilization was below 30%.

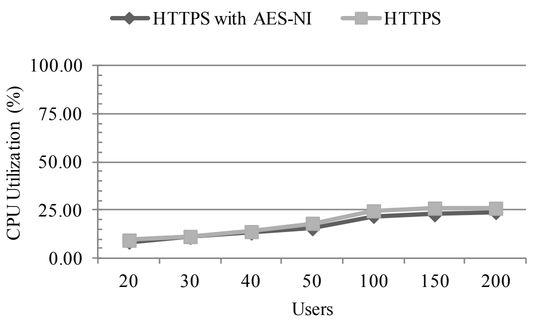

Figure 1: CPU Utilization with Eight Cores.

After binding the application to a single core, the CPU utilization increased. The load test is done with zero think time to ensure that a steady input rate is maintained at the web application. Little's Law [11] is used for validating the test results.

5.1. HTTPS Interaction without AES-NI

Initially, the load test was performed with the software based implementation of the AES algorithm. The application is set to use JDK without the NSS compiled libraries. The AES-NI feature of the Westmere-EP will not use the instruction set for the encryption and decryption steps involved in the AES algorithm. Figure 2 shows the measured throughput in pages/sec and average response time in seconds. The vertical axis starts from 1000 in order to enhance the visual difference in the throughput. Initially the throughput increases as the number of users is increased. At the load of 100 users in system, the throughput stops increasing and correspondingly the response time starts to increase at a linear rate.

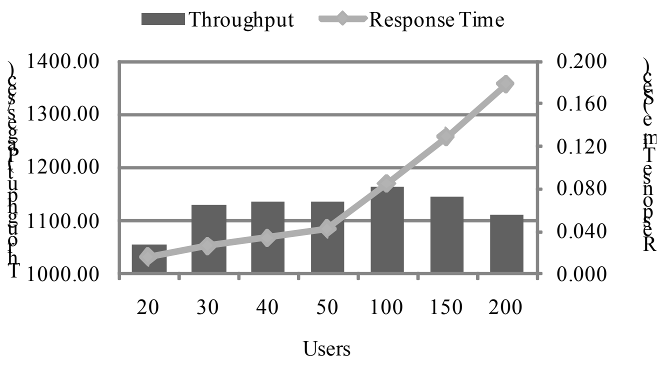

Figure 2: Throughput and response time for application without AES-NI.

Also, with the increase in number of users from 100 to 200 it can be seen that there is a drop in throughput.

5.2. HTTPS Interaction with AES-NI

To use the AES-NI feature, NSS library is compiled and put into the Java libraries. The java.security file needs to be updated in order to use the feature while encrypting/decrypting the message. A configuration file nss.cfg needs to be added to make the new provider work. There are several inbuilt parameters that need to be initialized in the file "PATH/nss.cfg" to make use of AES-NI;

```
name=NSS
nssLibraryDirectory=${java.home}/lib/amd64
nssDbMode=noDb
attributes=compatibility
```

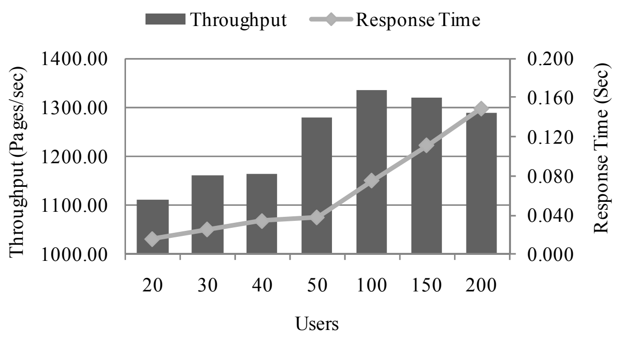

Figure 3: Throughput and response time for application with AES-NI

This file will configure the NSS provider to run in "noDb" mode and force Java to use the NSS libraries. Figure 3 shows the graph which has the AES-NI feature enabled. The behaviour of the application is the same as before, but there is an improvement observed in the throughput. The response time also decreases for the same set of users as compared to the previous test.

5.3. Performance Comparison between AES-NI and non AES-NI approach

In figure 4 the comparison between the throughput of the application with and without AES-NI is shown. It can be observed that there is an improvement of 10-15% in the average throughput for a load of up to 200 users.

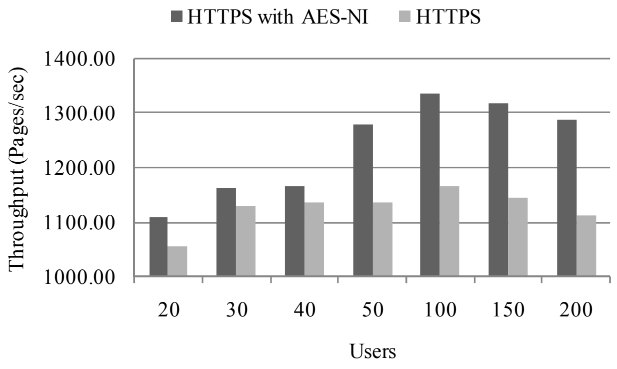

Figure 4: Throughput Comparison of application with and without AES-NI

As the number of users increased, the average response time of the application also improved with the AES-NI feature enabled. After the load of 50 users the response time started to increase linearly. A considerable difference is observed at a load of 200 users. Figure 5 shows the comparison between the average response times with and without AES-NI.

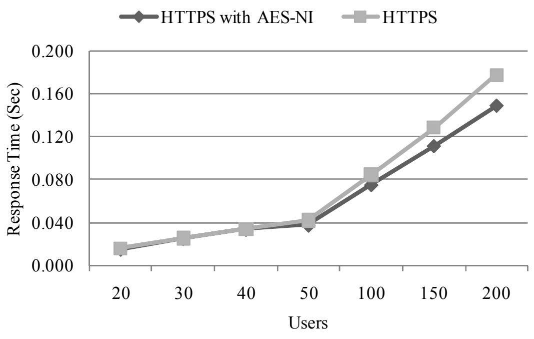

Figure 5: Average response time comparison.

In Figure 6, the CPU utilization of the application server attached to a single core is shown. For lower user load, the CPU was utilized more in the case of non AES-NI setup where software encryption/decryption overhead was involved. Gradually, the CPU starts to saturate for both cases with the increase in user test load.

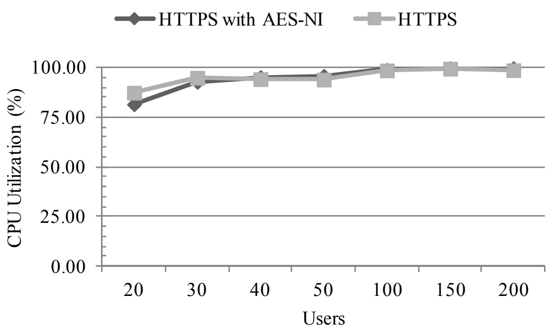

Figure 6: CPU utilization comparison with Single Core.

6. Conclusion

In our experiment we have shown how to measure the performance benefits of Intel's AES-NI feature. There is an improvement of 10-15% per CPU core observed in the average throughput of an application with AES-NI configured. The average response time is also reduced by 9%. The AES-NI feature of Intel Westmere-EP is useful for an online application performing secure client-server interactions with a sustainable load of concurrent users. A simple https application using AES will get a 10-15% performance benefit per core from the Intel AES-NI feature.

7. Acknowledgment

I am thankful to my team member Mr. Amol Khanapurkar for his able guidance during load testing.

8. References

[1] Advance Encryption Standard (http://en.wikipedia.org/wiki/Advanced_Encryption_Standard).

[2] Federal Information Processing Standards Publication (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf).

[3] Intel AES-NI Instructions (http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni).

[4] What is Data Encryption Standard (DES)? (http://searchsecurity.techtarget.com/definition/Data-Encryption-Standard).

[5] http://en.wikipedia.org/wiki/Cryptography_standards.

[6] http://en.wikipedia.org/wiki/Blowfish_(cipher).

[7] J2EE JPetStore (http://java.sun.com/developer/releases/petstore).

[8] http://www.cse.wustl.edu/~jain/cse567-06/ftp/encryption_perf/index.html.

[9] Network Security Services (http://www.mozilla.org/projects/security/pki/nss).

[10] Java PKCS#11 Reference Guide (http://docs.oracle.com/javase/6/docs/technotes/guides/security/p11guide.html).

[11] Little's Law (http://en.wikipedia.org/wiki/Little's_law).

[12] Shay Gueron, Intel Advanced Encryption Standard (AES) Instruction Set, http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set, 2010.