SmartSilc
Solutions by Silicom Ltd.

QuickAssist Technology to Accelerate

Feb. 2015

ABSTRACT

NGINX (pronounced, "engine-x") is considered to be the fastest growing web server worldwide [1]. It is an open source project backed by commercial source branch, with pretty good software design (asynchronous single thread) and features (server, reverse proxy, load balancer, etc.). NGINX is both feature rich and powerful. It exhibits ease of use to the administrator, along with a better than average performance figures. In fact, there are many "switches", "levers", "slides" and "button" one can push and pull, to tune up and optimize server's performance [2] to best fit a given platform. However, tuning NGINX operation it only half of optimization that can be implemented, for SSL crypto computation will always wait patiently around the corner to come and consume considerable CPU power; especially when it comes to asymmetric encryption, crypto key exchange or key signing. Looking further ahead, things will only get more serious, when SPDY will evolve to be the de-facto standard web protocol. Considerably larger portion of data will traversed encrypted through the net, and SSL handshakes and re-handshakes will be more and more ubiquitous.

Target audience of the test described herein are engineering teams, dealing with SSL accelerations for their implementation, or will soon get there. Lawful interception, intrusion detection, application delivery controllers, and most notably, Firewalls.

The reason NGINX was the server software of choice for the purpose of this test, is twofold. First, this is a user space application, of which operation on one hand, and constrains on the other, are relatively known and easy to convey. Second, it is native asynchronous design, with OpenSSL interface for HTTPS service. Thus, a quick integration to hardware crypto engine seems a very logical step to do. In short, we chose to demonstrate a user space application – exhibiting close to real life web traffic handling.

Keywords

NGINX; SSL Offload, Acceleration, SPDY, man-in-the-middle

1. INTRODUCTION

The primary goal of the benchmarks and tests described herein is to exhibit and demonstrate how NGINX HTTPS service is **increased** and **optimized** in performance, through the use of Intel® Coleto Creek 8955 acceleration chip set. The specific areas where an acceleration is expected are demonstrated. Finally, an analysis is reached regarding the measured capacity improvement, against the **cost of acceleration and offload**.

A "near real life" data of SSL acceleration options is required more and more, mainly among application vendors whose application is expected to massively deal with encryption, facing upcoming HTTP/2.0. Formerly considered an attack, "man-in-the-middle" increasingly becomes the mode of operation for more and more lawful interception operators.

2. TESTS CONSIDERATIONS

2.1 Apples and Oranges or Types of Tests

Most commonly, benchmark results of look-aside offload acceleration engines, are showing impressive results, under nominal and ideal tests scheme. For instance, testing close looped RSA primitive operations can give a clue of the nominal capability of the offload engine, but does not address other major concerns when opting for the use of encryption offload.

Silicom ltd., 8 Hanagar St., Kefar Sava 4442537, ISRAEL
sales@silicom.co.il www.silicom-usa.com www.smartsilc.com

When choosing a car, it is not enough to ask what is the engine's torque, but it is important to understand in advance how many seats are in the car, what is the expected MPG, etc. In other words, other important concerns may be:

- Power consumptions and mechanical envelope
- Host memory constraints and consumption
- Maximal number of SSL servers' certificates that can be used
- Maximal number of SSL client instances
- Time and efforts for application integration
- User space / Kernel space
- How far is offload implementation from software-only implementation
- Future scalability and virtualization support
- PCIe buss utilization
- What is the cost of offload
- How many RSA handshakes can be achieved, **for real**. What bandwidth can be achieved, **for real**.

This information is valuable in preliminary stages of system design, with offload integration in mind. In this current work we are rackling the last three concerns, of which the last item may be the key element for deciding to opt for an offload engine.

So, instead of running on tight loops as close as possible to the acceleration engine, we've attempted to install a full HTTPS service setup, on standard server. The high level method of the tests described herein was as follows:

1) Set up NGINX in an optimal setup
2) Benchmark software SSL performance, and check CPU utilization
3) Benchmark hardware SSL performance, and check CPU utilization
4) Compare results of software encryption and hardware encryption
5) Compare results to "nominal" close tight loop of hardware acceleration engine performance
6) Re-optimize NGINX and/or operating system setup

However synthetic and sterile they may be, the nominal benchmark results are the first reference for assessment of how far can we further go to optimize real life like test setup. These nominal results for the current Intel® Coleto Creek SKUs (the 8950 and 8955) are presented in table 1.

In near real life scenario, however, the figures described in Table 1, translate to more abstract gauges. The RSA operations rate translates to **SSL handshakes or connections per second**, while the bulk crypto figure translates to **SSL or HTTPS bandwidth**.

Great many variables become part of the test in this case, throughout the data path. Starting with the test equipment that can be either a dedicated network stress tool with SSL capabilities, a bunch of HTTPS clients' scripts shooting from several nodes, or any other tool that comes to mind. In this first test taken herein, a dedicated stress tool is used. Further down the data path, the ingress network interface is the next factor. A 10GbE interface may be good enough for SSL handshakes per second benchmarking, but may not serve as well, and even become bottleneck for SSL bandwidth testing. The OS TCP/IP stack is the next place to tune, where SSL connections per second benchmark may require different buffer setup than SSL bandwidth benchmark.

When testing SSL connections per second rate or SSL bandwidth tests, it is important not to accidentally slip into other types of test, unintentionally. Connections per second test could easily slip into a connections concurrency test, without notice, and to hit a glass ceiling, just because the process under test hits the open file descriptors limit. Or, by not carefully designing the clients operation, the test might include too many HTTP transitions (HTTP GET, probably), and before you know it, the results you get are, in fact, transactions per second, rather than connections per second.

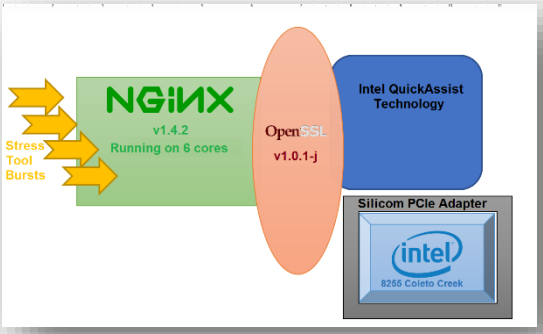
Back to the test setup, the software components integration is quite straight forward, and was implemented through the use of a will defined interface of each component.

Table 1 – Intel® Coleto Creek Nominal Performance

#	Intel® Coleto Creek SKU	RSA 1K ops/sec*	RSA 2K ops/sec*	AES128 Crypto**
1	8950	165K	35K	50Gpbs
2	8955	190K	40K	50Gbps

* Asymmetric cryptography
** Symmetric cryptography

Figure 1 - Software Components Integration



2.2 Asynchronous Operation

On the NGINX server itself, among the many "switches", "buttons", "levers", and "chokes" that can be used to adjust its performance, probably the most effective is the **number of CPU cores** to be occupied by asynchronous threads. When serving connections from multiple clients, that are established non-synchronously, then there is no question that the server side, the NGINX in this instance, should operate also in an **asynchronous fashion**. Intel® QuickAssist software suite, as the longer arm for this asynchronous mode of operation, is built just for that. Accessing the hardware offload engine is done through and API that can act either:

- Synchronously – every primitive access to the offload engine would not return, until completed;
- Asynchronously – every primitive access to the offload engine would immediately return. Completion would be signaled through a callback function.

Out of the two, the asynchronous mode was used in this test, to best suit NGINX mode of operation as a whole. OpenSSL package serves as the SSL engine for the NGINX on both software and hardware tests. On the hardware test, however, the cryptography tasks were forwarded to the Intel® Coleto Creek adapter, rather than being implemented on CPU. As a result, the asynchronous operation of the NGINX as a whole was slightly different:

- In the software-only tests – The NGINX thread operated asynchronously towards the HTTPS client;
- In the hardware assisted tests – Same as above the NGINX thread operated asynchronously towards client, and OpenSSL operated asynchronously towards NGINX thread.

However an insignificant observation it may look, orchestrating these two asynchronous operations together is a key element to squeeze best performance out of offload engine, under near real life conditions.

2.3 Get It or Not

A web HTTPS session consists of a TCP connection establishment, transporting SSL handshake, followed by clients' requests (HTTP method) and servers' responses. Depending on the specific HTTP and SSL/TLS versions, a re-handshake may be requested by either party, client or server. These session building blocks impose different burden on processing units, as it comes to cryptography. Most notably, RSA asymmetric cryptography often considered the center of gravity.

Therefore, in order to properly benchmark RSA operations capability, or more accurately, **SSL/TLS handshakes** per second, the part of the HTTP method should be a negligible as can be, and for the best, it should not be used at all.

In the test presented herein, a "GET… HTTP…" request was used, and a "200 Ok" response was expected. That was the behavior of the test equipment, and even if there was an attempt to minimize its effect by HTTP requesting a one byte size resource, still, network bandwidth was used, and bulk encryption power was invested (even for the shortest buffer), in excess.

Further test that are planned and would follow, would not send HTTP request at all. Once a session is established, it would immediately be terminated.

2.4 Cost of Offload

Operating an offload engine require transferring data to and from the accelerator. Indeed, DMA operation over PCIe bus, where the accelerator card is the DMA master, relive large part of buffer management off the host CPU; but still, CPU cycles needs to be vested to manage this path.

Offloading a task off CPU to a sub engine, therefore, is beneficial if such use lowers significantly the CPU cycles that are spent for the cryptography task; relieving valuable CPU cycles for business logic processing. A manual note to the GNU/Linux 'top' utility was used in this test to determine CPU usage during loads, with software or with hardware offload engine.

Figure 2 - SIlicom PE3iSC02 with Intel® ColetoCreek 8950 Chip Set

Once series of test is completed and both SSL handshake and bulk crypto is benchmarked, it would be interesting to see if both type of offload exhibit same gain in terms of gain and benefit.

3. TESTS

3.1 Outline

The following setup was used for the tests.

Server	Supermicro X9DRD-7LN4F, CPU 2xE5-2670 v2, 128Gb RAM
OS	Fedora16 x86_64 kernel 3.0.1
QuickAssist	QAT1.6.L.1.0.9-22
OpenSSL	1.0.1h
NGINX	1.4.2, patch nginx-1.4.2-005
Adapter	Silicom PE3iSCO3

Stress traffic was generated with a Spirent layer 7 packet generator.

3.2 Results – SSL/TLS Handshakes

Let's start with the bottom line. Intel® 8955 Coleto Creek chip set brings considerable added value under real life scenario. This fact, however has to be further detailed. The method of the test that was carried out herein, was to test a software only implementation, against an implementation that incorporates the Intel® 8955 Coleto Creek chip set. However, while NGINX as a software only implementation exhibited fair load balance and distribution across all incorporated CPU cores (maximum of 8 cores), same NGINX setup, but with Intel® 8955 Coleto Creek chip set as an offload crypto engine did not exhibit same fair load balance and distribution. Nevertheless, the Coleto Creek operation has brought:

a. **Significant performance improvements**
b. **Significant CPU relief**

The test still has a length to cover, but even under far from ideal conditions, the power of the Intel® 8955 could be demonstrated.

Table 2 - Interim Results for SSL/TLS Handshakes Benchmarks

1 x GET https://193.0.0.1/1.html	RSA 2K key	RSA 4K key
	AES128-SHA	
QuickAssist with Intel® 8955	21,376	4,130
	CPU usage: CPU 0-7: 80%; CPU 8-19: 0%	
Software only	4,866	2,072
	CPU usage: CPU 0-19: 100%	

We need to further check core affinity, QPI buss avoidance, no question. But even with such non optimal setup, Intel® 8955 Coleto Creek chip set has great value:

1) It literally doubles SSL/TLS handshake rate with 2k and 4k keys;
2) It relieves considerable amount of CPU cycles.

And when considering that the relieved CPU cycles are of an Ivy bridge Intel® microarchitecture running at 2.5 GHz, the added value of the accelerator immediately appear.

3.3 Results – Bandwidth

Yet to be completed next. More optimal setup would be configured for meaningful tests results.

4. Where to go from here

The target is to get **close to the nominal capability** of the acceleration engine as brought herein in Table 1. There is a lot to cover down the road. Tests will be continued with more efficient NGINX setup, to show that even with tuned software implementation, and offload engine for encryption still brings considerable value, especially for the heavy lifting tasks, of the asymmetric cryptography.

A fair estimate, based on several other tests, points at 10,000 RSA 2K SSL/TLS handshakes per second as a cap number for typical user space software implementation on a server similar to the one that was tested in this test; with all core screaming 100% utilization.

Therefore, based on the results brought herein, in Table 2, specifically from the CPU relief that was enabled by the offload engine, it is fair to estimate that in more optimal application set up, the above number of 10K handshakes, could be doubled, and perhaps even more.

To gain better results out of the setup, the following areas are to be more thoroughly observed:

1) Incoming traffic **load balancing** across CPU cores and core affinity scheme should be tightened;
2) Not using HTTP request methods at all, for instance, not sending GET request at all. Later on a 1 byte resource request would be added in controlled manner.
3) Two threads per core with and without CPU hyper threading enablement would be tested.
4) TCP tuning would be revisited.

In a more optimized setup, a bandwidth tests would be carried out as well.

Further down the road, same tests would be expanded to ECC cryptography. Moreover, dual chip adapters, as well as quad chip PCIe adapters, that are already available by Silicom, would be tested, to demonstrate linear scalability.

REFERENCES

[1] NGINX Now Powers 146 Million Websites, Launches Version 1.6 and 1.7 Of Its Web Server (Apr, 24th, 2014)
http://techcrunch.com/2014/04/24/nginx-now-powers-146-million-websites-launches-versions-1-6-and-1-7-of-its-web-server/

[2] Tuning NGINX for Performance (Oct. 10th, 2014)
http://nginx.com/blog/tuning-nginx/

[3] NGINX SSL Performance (July, 2014)
http://nginx.com/wp-content/uploads/2014/07/NGINX-SSL-Performance.pdf