SmartSilc
Solutions by Silicom Ltd.

QuickAssist Technology Acceleration Quadruples Squid Web Cache Server Performance

Apr. 2016

ABSTRACT

Web cache proxy servers are quite unique software creations. Developing, optimizing and even operating and integrating web cache servers require high level of expertise in two distinct fields, rarely found in the same profile of developers and integrators.

Firstly, as a web proxy, cache server in general features high degree of network engineering hallmark. Form the implementation of lower layers of networking and TCP connections handling, up to layer 7 HTTP intricacies, moving up to different web applications aspects that should be taken into consideration, it is a high profile networking work for quality web caching. One of the prominent aspects of proxy cache configuration is the dynamic content provisioning. While static content is provisioned from cache, and dynamic content would be brought from source servers, successful and optimal combination of the two is a key factor for web cache added value.

Second, good caching is all about proper storage handling. Starting from the choice of storage structure and down to the proper selection and configuration of the file system underneath, storage plays yet another key factor when it comes to cache service. Storage is required to serve as quickly as possible, and as stable as possible, without losing integrity. There are implementations (e.g., Varnish) that have taken this requirement to the extreme, and have focused mainly in storage optimization. Having gone to a great length to that direction, those implementation ended up without the capability to scale into other direction, such as SSL connectivity, which is the subject of this paper.

So all in all, it is not all simple at all to implement a good web cache proxy server. One have to be specialized in many aspects of engineering (networking, storage, etc.), and to strike the right balance as it comes to deciding what really matters. Actually, everything matters.

1. SQUID AND SSL

Squid web cache proxy server [1] is a well maintained open source project that has been out there for quite a while, and has a well-established install base. From software architecture point of view, squid codebase sustains a subtle balance between maintainable code on one hand, and well performing code. This can be said on the storage implementation in Squid and also on the networking part of it. In this philosophy, the SSL part is implemented as an elegant overlay on the communications layer, using OpenSSL libraries and API thereof [2]. This is the place to comment on the Squid software design tactics, as it comes to the communications layer, and say that architect took a very smart choice in implementing this layer with non-blocking I/O in mind. This design approach enabled Silicom to successfully integrate the hardware acceleration engine.

2. ADD INTEL® QUICKASSIST

2.1 Why Offload Engine

Intel® QuickAssist Technology (QAT) [3] is a modern software suite, driving Intel® acceleration chipset family [4], around which, Silicom has built a wide range of PCIe adapters [5] in various form factors, and compositions. From single chipset adapter, to dual, and up to quad chipset adapter, Silicom offer support for various PCIe bus profile (8 lanes, 16 lanes, PCIe v2 and PCI v3). Luckily, Intel® has gone to a great length spreading the support for QAT in a lot of popular open source projects [6], and not only that, but they've also added QAT as a crypto engine for the asynchronous version of OpenSSL, [7] soon to be part of the official release.

So, to cut a long story short, replacing Squid's native OpenSSL version with QAT enabled asynchronous OpenSSL required small and straight forward modifications to `<squid>/src/ssl_support.c` SSL overlay source module, and from this point onward, all crypto work had moved from software to Silicom adapter.

* Non-Blocking I/O

THE MOST OPTIMAL WAY TO BENEFIT FROM A HARDWARE OFFLOAD ENGINE, SUCH AS INTEL® QUICKASSIST TECHNOLOGY ADAPTER, WOULD BE BY DRIVING IT IN NON-BLOCKING I/O MODE (ASYNCHRONOUSLY).

There are two fronts where offload engine for SSL add value. Firstly, number of SSL handshakes per second is greatly improved by the offload engine, simply because the CPU is relieved off the heavy lifting of complex asymmetric encryption computations, and a purpose built dedicated engine is used instead. Bulk crypto, which is performed on the connection's payload is also performed on the crypto engine, enhancing the offload effect.

2.2 Tests

The test scheme that was used was simple. Same version of Squid was operated as software only, and then with the assistance of Intel® based Silicom SSL acceleration adapter, and the results were compared to one another. Table 1 summarizes the test environment.

Server	Dell Inc. PowerEdge R420
CPU	Intel(R) Xeon(R) CPU E5-2440 v2 @ 1.90GHz
NIC	Silicom PE210G2SPi9 2 x 10Gb
SSL card	Silicom PE3iSCO2
Squid	v2.6.9 v3.5.13
OpenSSL	1.0.1m

Table 1 - Test Setup

Apache benchmark [8] tool was used as a stress tool.

2.3 Results

At a glance, the addition of hardware SSL acceleration vastly and improves total handshakes per second.

Squid	Accelerated	Conns/Sec	CPU%
v2.6.9	No	354	35%
v2.6.9	Yes	1,276	17%
v3.5.13	No	298	40%
v3.5.13	Yes	1,176	25%

Table 2 - Tests Results

It is immediately visible that SSL offload both eased up on CPU (leaving much more breath for business logic processing) and more importantly, allowed for four times as many connections to serve. The setup remain quite a naïve setup throughout the tests, meaning that the Squid process was running as single process on single CPU core, without stretching to parallel processing, that would have exhibited true linear performance scalability with connections handling.

SUMMARY

We've seen how easy it is to empower Squid with more muscles without compromising any other aspects of its operation, and through a simple addition of crypto engine and quietly replacing OpenSSL version with another. Next, as mentioned above, test would be performed to allow for parallel processing and linear scalability.

REFERENCES

[1] See http://www.squid-cache.org/

[2] See https://www.openssl.org/

[3] See: http://www.intel.com/content/www/us/en/embedded/technology/quickassist/overview.html

[4] See: http://ark.intel.com/products/codename/60172/Coleto-Creek

[5] See: http://www.silicom-usa.com/article.aspx?item=946&amp%3Bln=en

[6] See: https://01.org/packet-processing/intel%C2%AE-quickassist-technology-drivers-and-patches

[7] See: http://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/accelerating-openssl-brief.pdf

[8] See: https://httpd.apache.org/docs/2.4/programs/ab.html