CHELSIO T6 UNIFIED WIRE DELIVERS LEADERSHIP PERFORMANCE WITH F5 NGINX OFFLOAD

Using proven TCP/IP Offload Engine (TOE) and Inline TLS/SSL Acceleration

Overview

Chelsio T6 Unified Wire adapters are designed specifically to offload computationally intensive network and cryptographic operations from general purpose CPUs. The T6 Unified Wire adapters support TCP Offload Engine (TOE) for hardware-based full offload of the TCP/IP protocol and Inline TLS/SSL offload capability up to 100Gb line rate performance. Servers with system load comprised of network and cryptographic operations can utilize T6 adapters for solid performance and efficiency improvements through offloading of these CPU-intensive operations.

NGINX Plus brings everything you love about NGINX Open Source, adding enterprise-grade features like high availability, active health checks, DNS system discovery, session persistence, and a RESTful API. NGINX Plus is a cloud-native, easy-to-use reverse proxy, load balancer, and API gateway. Whether you need to integrate advanced monitoring, strengthen security controls, or orchestrate Kubernetes containers, NGINX Plus delivers with the five-star support you expect from NGINX.

This paper presents a comparison of Chelsio 25G dual-port performance for the modes listed below for the NGINX Plus web server application running on Linux.

- NIC: Using T6 for host CPU-based processing of TCP/IP and TLS/SSL operations.
- TOE: Using T6 for TCP/IP protocol offload and host CPU-based processing for TLS/SSL operations.
- Inline TLS/SSL offload: Using T6 for both TCP/IP and TLS/SSL offloads.

It establishes the fact that using T6 to offload both the network and cryptographic operations, the NGINX Plus web server consumes significantly less CPU with consistent performance.

Test Results

The below graph presents the %CPU and throughput results of NGINX Plus web server, using Chelsio T6 in plain NIC, TOE and Inline TLS/SSL offload (TOE-TLS) modes.

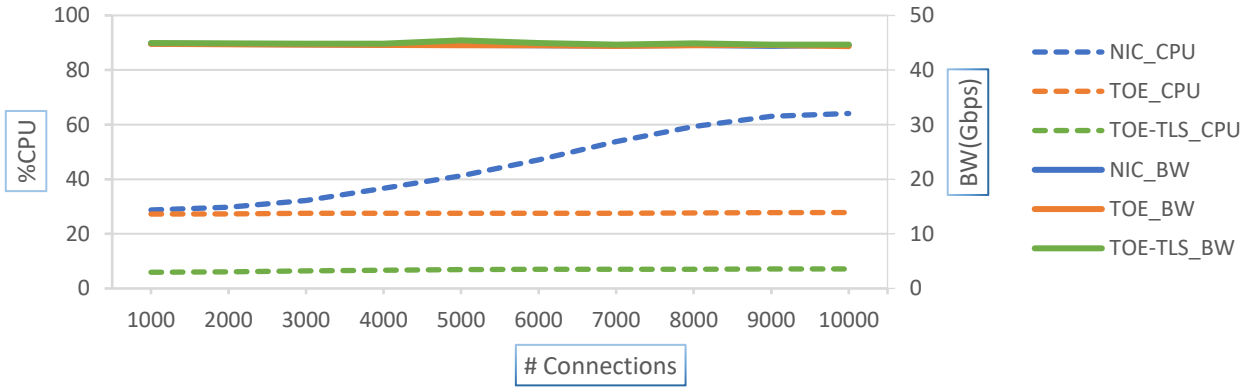

Figure 1 - Throughput & CPU Usage vs. # Connections

With the Chelsio Inline TLS/SSL offload solution, only 7% CPU was used on the server even with 10000 connections, while the TOE mode used 28% CPU and plain NIC mode used 64% CPU.

This can permit radically reduced system cost by enabling the use of a less expensive CPU or the freed-up CPU can be used for running other applications or workloads. By offloading both network and crypto operations, 55% of a typical $5k server cost is saved, which translates to $2750 savings per NGINX server. Considering that data centers may (depending on scalability requirements) typically deploy tens to hundreds to thousands of NGINX servers, such cost savings enabled by T6 Inline TLS/SSL offload can make a major TCO reduction and boosts ROI. These savings will be magnified as data centers transition to 200Gbps and 400Gbps speed.

In all the 3 modes, a consistent throughput profile is maintained delivering up to 44.8 Gbps.

Test Configuration

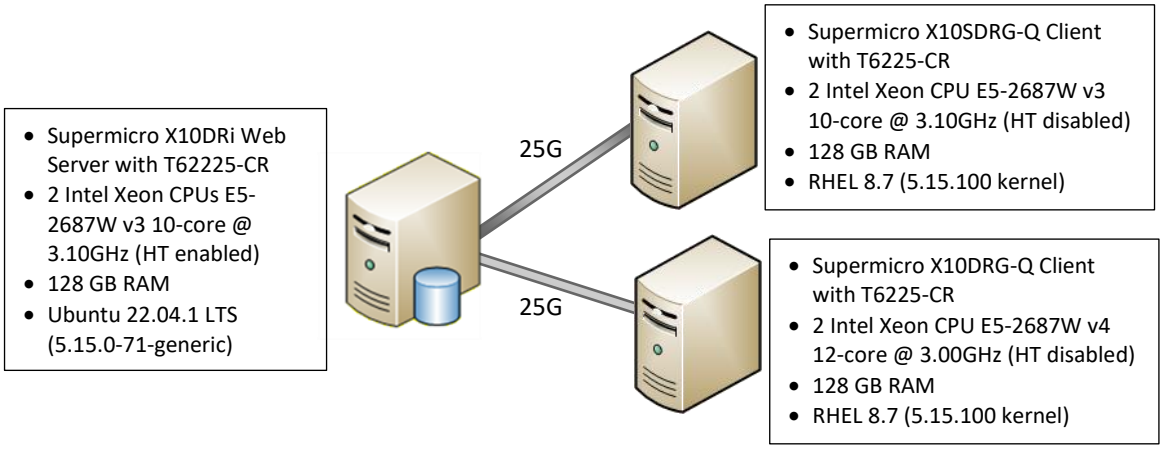

Figure 2 – Test setup for NGINX Offload

The setup consists of a server connected to 2 client machines back-to-back using both the ports on the server. The latest Chelsio Unified Wire driver for Linux v3.18.0.1 was installed on all machines. Standard MTU of 1500 was used on the ports.

Each client establishes 500 to 5000 connections using *wrk* tool to download the files from the NGINX Plus web server.

Setup Configuration

General Configuration

Execute the steps listed below on the server and all client machines.

i. Disable virtualization, c-state technology, VT-d, Intel I/O AT, SR-IOV in system BIOS.
ii. Compile and install the latest Chelsio Unified Wire package and reboot the machine.

```
[root@host~]# cd ChelsioUwire-3.18.0.1
[root@host~]# make toe_install
[root@host~]# reboot
```

iii. Set cpupower governor to performance.

```
[root@host~]# cpupower frequency-set --governor performance
```

iv. Set the below tuned-adm profile for BW/IOPs test.

```
[root@host~]# tuned-adm profile network-throughput
```

v. Load the Chelsio NIC driver (*cxgb4*) and bring up interface with IPv4 address.

```
[root@host~]# modprobe cxgb4
[root@host~]# ifconfig ethX <IPv4 address> up
```

vi. Configure the below sysctls for running higher number of connections.

```
[root@host~]# sysctl net.core.somaxconn=40960
[root@host~]# sysctl net.ipv4.tcp_tw_reuse=1
```

Server Configuration

i. Enable kTLS in openssl configuration file, */etc/ssl/openssl.cnf* by adding the below line.

```
[system_default_sect]
options = ktls
```

ii. Install NGINX plus by following the steps from NGINX Docs. Verify the nginx binary version.

```
[root@host~]# nginx -v
nginx version: nginx/1.23.4 (nginx-plus-r29)
```

iii. Configure the web server by updating required settings in */usr/local/nginx/conf/nginx.conf*

```
#user  nobody;
worker_processes  40;

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    keepalive_timeout  65;

    server {
        listen       443 ssl backlog=20480;
        server_name  0.0.0.0;

        keepalive_timeout  3600;
        keepalive_requests 100000;

        ssl_certificate      /root/server.crt;
```

```
        ssl_certificate_key  /root/server.key;

    # SSL Settings
    # ##
            ssl_protocols TLSv1.2; # Dropping TLSv1 SSLv3, ref: POODLE
            ssl_ciphers AES128-GCM-SHA256


            ssl_prefer_server_ciphers on;

            location /2MiB/ {
                 sendfile on;
                 root /nginx;
                    }
            }
    }
```

iv. Create a data file for the clients to download.

```
[root@host~]# mkdir -p /nginx/2MiB/
[root@host~]# dd if=/dev/urandom of=/nginx/2MiB/1_1.txt bs=1024 count=2000
oflag=direct iflag=fullblock status=progress
```

v. Configure the server in NIC/TOE/TLS offload modes.

Software:

```
[root@host~]# t4_perftune.sh -s -n -Q nic
```

Chelsio TOE offload:

```
[root@host~]# modprobe -v t4_tom
[root@host~]# cat nginx_toe_cop.rule
all => offload !ddp !coalesce
[root@host~]# cop -d -o nginx_toe_cop.policy nginx_toe_cop.rule
[root@host~]# cxgbtool ethX policy nginx_toe_cop.policy
[root@host~]# t4_perftune.sh -s -n -Q ofld
```

Chelsio Inline TLS/SSL offload:

```
[root@host~]# modprobe -v t4_tom
[root@host~]# cat nginx_tls_cop.rule
src or dst port 443 => offload tls mss 32 bind random !nagle
all => offload
[root@host~]# cop -d -o nginx_tls_cop.policy nginx_tls_cop.rule
[root@host~]# cxgbtool ethX policy nginx_tls_cop.policy
[root@host~]# t4_perftune.sh -s -n -Q ofld
```

vi. Start the NGINX plus server.

```
[root@host~]# nginx -c /usr/local/nginx/conf/nginx.conf
```

NOTE: Restart the web server after configuring the different modes specified in step v.

Client Configuration

i. CPU affinity was set.

```
[root@host~]# t4_perftune.sh -n -Q nic -c 0-15
```

ii. *wrk* tool (v4.2.0) was run from both the Clients to download the file.

```
$inst = 10, 20 .. 100

[root@host1~]# for i in $(seq 1 ${inst}) ; do /root/wrk-4.2.0/wrk -t 1 -c 50
-d 100s https://102.25.1.42/2MiB/1_1.txt & done
[root@host2~]# for i in $(seq 1 ${inst}) ; do /root/wrk-4.2.0/wrk -t 1 -c 50
-d 100s https://102.25.1.42/2MiB/1_1.txt & done
```

Conclusion

This paper compares the performance of Chelsio T6 TOE and Inline TLS/SSL offload solution with software-based protocol and cryptographic processing running on Linux. With consistent throughput and considerable CPU savings (even with 10000 connections), the Chelsio solution proves to be the best choice for users looking for the highest level of data security and integrity, with no compromise in performance. T6 is currently the only secure engine capable of full TCP/IP Offload at 100Gbps. The introduction of integrated encryption within a NIC price and power envelope should facilitate the migration towards secure cloud networks and storage.

Related Links

100G NGINX Offload & Encryption
NGINX Applications Offload
The True Cost of Non-Offloaded NICs
Concurrent Offload & Encryption at 100GbE
Chelsio Terminator 6 ASIC 100GE Crypto Offload
The Chelsio Terminator 6 ASIC
Learn more about NGINX Plus