

Lanner

Network Computing

Innovative Platforms for Secure Networking Infrastructure

**Network Security Performance
Enhancement using
Hyperscan & DPDK**

Version: 0.1

Date of Release: 2019-03-04

Overview

DPI (Deep Packet Inspection), IDS (intrusion detection system) and IPS (intrusion prevention system) are a few of the critical infrastructures used in cyber security technology. They are frequently used in addition to other deployed hardware firewalls. Given the pace at which new applications are emerging and the data is growing, there is an increased demand for securing oneself from cyber threats. Hence, to achieve success without compromising on application serviceability and QoS (quality of service), a highly flexible system architecture based on software is required to build performant platforms for DPI, IDS and IPS.

In a conventional approach to enhance performance, IT managers consult with purpose-built hardware vendors to come up with a set of line-rated, dedicated devices. Due to its vendor lock-in nature, maintenance cost tends to pile up operational expenditure and negatively impact profitability. With the ever increasing threats of cyber-attack, for the system defense to stay up to date, frequent software and signature updates become highly critical.

Software pattern matching, compute intensive module, multi-core implementation, and hyper-threaded x86 architectures are an optimal and cost-efficient combination to boost network security performance. This paper showcases how Lanner's all-new high performance 1U network appliance [NCA-5710](#), powered by dual Intel® Xeon® Scalable processors, can improve its DPI performance with integrated accelerators like DPDK and Hyperscan technology.

Hyperscan Technology

Hyperscan is a software based regular expression (regex) matching library. It is an open source project that uses a highly optimized algorithm and takes advantage of Intel® architecture. It supports PCRE (Perl Compatible Regular Expression) syntax. It is very flexible to use in real-world networking deployments as it comes with pre-built C APIs. Its pattern and string matching capability offers low latency, high parallelism and easy portability (due to its OS independent nature). It is designed to offer high performance and has the ability to match multiple expressions simultaneously.

For Hyperscan documentation, please refer to:

<https://software.intel.com/en-us/articles/introduction-to-hyperscan>

This white paper demonstrates test results with default search engine for Snort®, Snort® patched with Hyperscan and Snort® running with DPDK Kni. Snort-2.x versions need a separate patch to take advantage of Hyperscan. Hyperscan can be enabled or disabled by changing the configuration files.

Software - Snort®

Snort® is an open source network intrusion prevention system, capable of performing real-time traffic analysis and packet monitoring on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes. These attacks include buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts and other complicated threats. In this test, Snort® is running in Linux* kernel space.

Quote from <https://snort.org/>

DPDK KNI

The Kernel NIC Interface (KNI) is a DPDK kernel module that allows userspace applications to share packets with the kernel networking stack. To accomplish this, DPDK userspace application uses an IOCTL call to request the creation of a KNI virtual device in the Linux* kernel. The IOCTL call provides interface information and the DPDK's physical address space. This is re-mapped into the kernel address space by the KNI kernel loadable module that saves the information to a virtual device context. The DPDK creates FIFO queues for packet ingress and egress to the kernel module for each device allocated.

The KNI kernel loadable module is a standard net driver, which upon receiving the IOCTL call access the DPDK's FIFO queue to receive/transmit packets from/to the DPDK userspace application. The FIFO queues contain pointers to data packets in the DPDK. This:

- Provides a faster mechanism to interface with the kernel net stack and eliminates system calls
- Facilitates the DPDK using standard Linux* userspace net tools (tcpdump, ftp, and so on)
- Eliminate the copy_to_user and copy_from_user operations on packets.

For more information please refer to:

https://doc.dpdk.org/guides/sample_app_ug/kernel_nic_interface.html

Traffic Profile

For the purpose of simulating a real-world network, a pcap file is loaded into the traffic generator. Its packet size varies anywhere from 42B to 1514B. This pcap is captured using a web crawling script that collects traffic from top 200 sites.

Below command line shows how Snort® can be used to read a pcap.

- Snort® -r <pcap file>

Below is an example on the different fields of a pcap:

05/03-07:03:40.913946 192.168.15.101:53 -> 192.168.15.12:51806

UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:156 DF

Len: 128

Packet Field Information	
Date	05/03-07:03:40.913946
Source IP	192.168.15.101
Source Port	53
Destination IP	192.168.15.12
Destination Port	51806
IP Header Length	IpLen
Total Packet Length	DgmLen
Protocol	UDP

Table 1: Pcap Information

Test Methodology

The following test methodology demonstrates performance of Snort® when default aho-corasick search algorithm is replaced with Hyperscan.

1. Configure and start traffic on Pktgen:
 - `pktgen -l 3,4,5,6,7 -n 4 -w 02:00.0 -w 02:00.1 -- -p 0x3 -P -m "[1:3].0, [2:4].1" -s 0: snort/<pcap> -s 1: snort/<pcap>`

Pktgen Arguments:

- l** - logical cores
- n** - Number of memory channels
- w** - Add a PCI device in white list.
- P** - Enable PROMISCUOUS mode on all ports.
- m** - Matrix for mapping ports to logical cores.
- s P:file** - PCAP packet stream file, 'P' is the port number.

Bi-directional traffic is started on Pktgen at 10% rate using 2x10GbE ports by issuing the following commands on pktgen-DPDK's terminal

- Set all rate 10
- Start all

2. Configure and run Snort®

Scenario 1: Snort® running in kernel space

- Bring up ports
 - `ifconfig enp24s0f0 up`
 - `ifconfig enp24s0f1 up`
- Snort® commandline:
 - `taskset -c 2 snort -Q -i enp24s0f0:enp24s0f1 -Nb -A none -c /usr/local/etc/snort/snort_hs.conf`

Snort® Arguments:

- taskset** - Set the core needed to run snort
- Q** - Inline mode
- i <if>** - Listen on interface <if>
- N** - Turn off logging (alerts still work)
- b** - Log packets in tcpdump format (much faster!)
- c <rules>** - Use Rules File <rules>

Snort® Configurations:

For snort - unmodified:

config detection: `search-method ac-split search-optimize max-pattern-len 20`

For snort - Hyperscan MPSE:

config detection: `search-method Hyperscan split-any-any`

There are two tests in Hyperscan: PCRE-Hyperscan(default config detection setting) and MPSE-Hyperscan(config detection: search method Hyperscan).

Scenario 2: Configure Snort®(Hyperscan) with DPDK kni

- DPDK configurations and commandLine:
 - `insmod <dppdk>/kmod/igb_uio.ko`
 - `insmod <dppdk>/kmod/rte_kni.ko kthread_mode=multiple`
 - `<dppdk>/tools/dpdk-devbind.py -b igb_uio 18:00.0 18:00.1`
 - `<dppdk>/examples/kni/build/kni -c 0xf0 -n 4 -- -P -p 0x3 --config="(0,4,5,4),(1,6,7,5)"`
- Bring up Ports
 - `ifconfig vEth0_0 up`
 - `ifconfig vEth1_0 up`
- Snort® commandline
 - `taskset -c 2 snort -Q -i vEth0_0:vEth1_0 -Nb -A none -c /usr/local/etc/snort/snort_hs.conf`

DPDK kni Arguments:

-n <NUM>: Number of memory channels per processor socket

-P : Set all ports to promiscuous mode so that packets are accepted regardless of the packet's Ethernet MAC destination address.

-p <PORTMASK>: Hexadecimal bitmask of ports to configure.


-c corelist: [] The EAL options must include the lcores specified by lcore_rx and lcore_tx for each port, but does not need to include lcores specified by lcore_kthread as those cores are used to pin the kernel threads in the rte_kni kernel module.

--config="{(port,lcore_rx,lcore_tx[,lcore_kthread,...])[(port,lcore_rx,lcore_tx[,lcore_kthread,...])]" : Determines which lcores the Rx and Tx DPDK tasks, and (optionally) the KNI kernel thread(s) are bound to for each physical port.

3. Once Snort® is started, on a different terminal, application is stopped by calling pkill command.
 - sleep <time in seconds>; pkill snort
 - Time in seconds can be 30 seconds or 5 minutes. The results for both 30s and 5mins are consistent.
4. Throughput (Mbps) is noted down from pktgen-DPDK side before Snort® is killed
5. Once Snort® completes tests, packets-per-second reported by Snort® is also noted down.

DPI Performance Enhancement with Intel Hyperscan+DPDK KNI

**NCA-5710
Network Appliance**



- Support Dual Intel® Xeon® Processor Scalable Family (Codenamed Skylake-SP)
- 4x NIC Module Slots
- 12x 288pin DDR4 2666 MHz REG DIMM, Max. 384GB

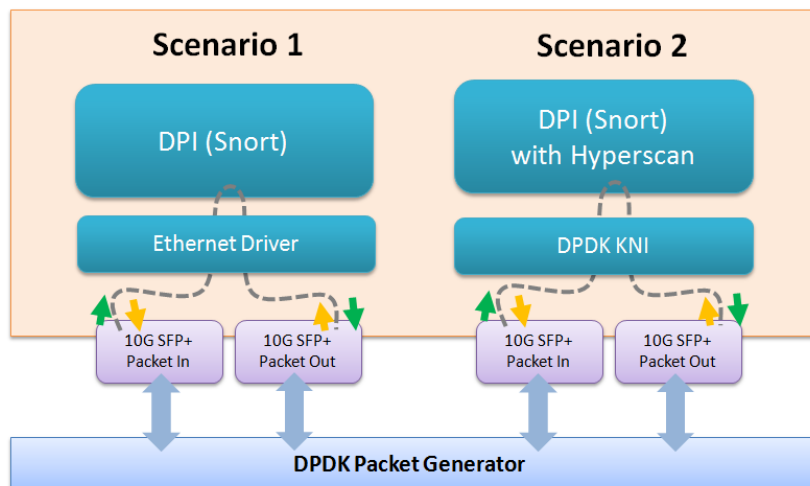


Figure 1: Test Setup

System Configuration

DPI Network Appliance- NCA-5710

Hardware	
Platform	Intel® Xeon® Scalable Processor
CPU	Intel® Xeon® Gold 6144 CPU @ 3.50GHz * 2
Memory	8 GB (4 GB * 2) DDR4
HDD	320GB WD3200LPLX
NIC	4-port 10Gbe SFP with Intel Fortville XL710 Ethernet controller (NCS2-IXM407) *2
Software	
Host OS	Ubuntu 16.04.1 with kernel 4.4.0-116-generic
NIC Driver	i40e v1.4.25-k (natural from OS kernel)
DPDK	v16.07.2
DAQ	v2.0.6
Snort	v2.9.8.2 with snort-2982-hyperscan-v3.patch
Snort Rule	snortrules-snapshot-2983
Hyperscan	v5.0.0

Table 2: DPI Network Appliance -NCA-5710 System Configuration

4.2 Packet Generator (MB-8896)

Hardware	
Platform	Intel® 5th Generation Processors
CPU Model name	Intel® Xeon® CPU E5-2640 v4 @ 2.40GHz * 2
Memory	32GB (8 GB * 4) DDR4
HDD	240GB Micron M500IT
NIC	2-port 10Gbe SFP with Intel Corporation 82599ES Ethernet controller (IXM204) 4-port 10Gbe SFP with Intel Corporation 82599ES Ethernet controller (IXM405)

Hardware	
Software	
Host OS	Ubuntu 16.04.3 with kernel 4.4.0-36
DPDK	v18.02
Pktgen-DPDK	v3.4.9

Table 3: Packet Generator System Configuration

Test Results

Test Case 1: Snort® running in kernel space vs Snort® running with DPDK kni.

(Snort v2.9.8.2 + Hyperscan v5.0 + DPDK KNI)

Scenario	Snort®+Hyperscan		Snort®+DPDK+Hyperscan		% in Difference
	Pktgen	Snort	Pktgen	Snort	
	Mbps	pkts/sec	Mbps	pkts/sec	
	190	34933	1252	202909	658%
	191	34271	1232	199646	645%
	189	34897	1257	199445	665%
	190	34262	1258	200500	662%
	190	34157	1251	199432	658%

Table 4: Test Case1 - Kernel Space vs DPDK kni

Lanner appliance NCA-5710 (powered by Intel® Xeon® Scalable Processor,) achieved around **600%+** performance gain after running Snort® with DPDK kni and hyperscan.

Test Case 2: Snort® default algorithm vs Snort® with Hyperscan patch.

(Snort v2.9.8.2 + Hyperscan v5.0 + DPDK KNI)

Scenario	Default Snort®+DPDK		Snort®+DPDK+Hyperscan		% in Difference
	Pktgen	Snort	Pktgen	Snort	
	Mbps	pkts/sec	Mbps	pkts/sec	
	550	121217	1252	202909	227%

	558	116661	1232	199646	220%
	646	120758	1257	199445	194%
	549	115556	1258	200500	229%
	643	121500	1251	199432	194%

Table 5: Hyperscan vs Aho-corasick

For Test Case 2, default Snort® that uses aho-corasick search algorithm is compared with Hyperscan search algorithm. Both tests were run with DPDK KNI. Lanner appliance NCA-5710 observed around **200%+** performance improvement with Snort® patched with Hyperscan over default Snort®.

In conclusion, with a combination of DPDK technology and an accelerated Hyperscan regex search engine, we can achieve up to 6x the base performance.



©Lanner Electronics Inc. All rights reserved.



Intel and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.