Technology Guide

High Density Scalable Load Balancer - A VPP-Based Layer 4 Load Balancer

Authors

Houxiang Cai

Tao Zhu

Hongjun Ni

Mrittika Ganguli

Xiang Wang

Pan Zhang

Yichao Ma

Serlina Yu

Yiming Jing

Zhijun Tang

1 Introduction

The effort to build a software-based Layer 4 load balancer (LB) running on x86 cores has been in play since 2006. These LBs are deployed as virtual machines and as software on bare metal implementations. Hyperscale Cloud Service Providers (CSPs) have developed solutions on bare metal that are lower in cost and easy to deploy and scale. CSPs use these LBs to optimize their internal infrastructure. They also sell these LBs to subscribers to use in rented instances. One of these solutions was developed and open-sourced by Google* called MAGLEV, which is a cloud network LB. MAGLEV is a generic LB that is made platform-independent for hyperscale deployment and uses unique acceleration techniques for performance. Yahoo* Japan developed an optimized LB based on FD.io VPP and added features to scale the solution for Load Balancer as a Service (LBaaS). This implementation achieved line rate at 10 Gbps using 4 cores. The existing open-source software LBs have performance and scalability limitations for current users and typically are limited to 1 million concurrent connections and approximately 2 Mpps of throughput per core. End users have invested substantially to try and overcome the following limitations but have not reached a desired performance profile:

- High Performance: Achieve a throughput of 150 Mpps with 100 M sessions per node
- Scalability: Throughput can increase linearly as CPU cores increase

In this paper, we describe the FD.io/VPP-based (Vector Packet Processor) L4 LB solution that fully leverages Intel® architecture and Intel® Ethernet adapter hero features. The paper illustrates the architecture of the solution, the four key optimization approaches made (using the flow director of the network card, optimizing graph node path, vector processing, and data structure optimization to improve cache misses) to improve performance and scalability, and the results after using these optimization methods. Compared to an industry software LB, the IA-oriented optimized load balancer can achieve three times more packets per second, ten times more connections per second, and almost linear throughput increases as CPU cores scale up.

The document also provides a detailed explanation of how the High Density Scalable Load Balancer (HDSLB)-VPP performs throughput and new connection performance testing. Finally, the test results are analyzed and summarized. The solution HDSLB-VPP with improved leading throughput performance and recordable concurrent connections provides ISV/CoSP/ODM companies a good choice to optimize their own business or reduce their get-to-market effort (resources and time).

This document is part of the [Network Transformation Experience Kits]().

Table of Contents

Figures

Tables

Document Revision History

Revision	Date	Description
001	November 2023	Initial release.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
AVX-512	Intel® Advanced Vector Extensions 512 (Intel® AVX-512)
CPS	Connections Per Second
CPU	Central Processing Unit
DLB	Intel® Dynamic Load Balancer
DPDK	Data Plane Development Kit
DUT	Device under test
FNAT	Full Network Address Translation
HDSLB-VPP	VPP-based High Density Scalable Load Balancer
Intel® Ethernet FD	Intel® Ethernet Flow Director
LB	Load Balancer
MCNAT	Multiple Cloud Networking Automation
NAT	Network Address Translation
NSIT	Network System Integration Testing Framework
RFC2544	Benchmarking Methodology for Network Interconnect Devices
RSS	Receive Side Scaling
SDK	Software Development Kit
SNAT	Source Network Address Translation
SR-IOV	Single Root I/O Virtualization
VPP	Vector Packet Processing
VRRP	Virtual Router Redundancy Protocol

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
Intel® Xeon® Scalable Platform Built for Most Sensitive Workloads	https://www.intc.com/news-events/press-releases/detail/1423/intel-xeon-scalable-platform-built-for-most-sensitive
Intel® Ethernet Flow Director (Intel® Ethernet FD)	https://www.intel.com/content/www/us/en/developer/articles/training/setting-up-intel-ethernet-flow-director.html
Intel® AVX-512 Instructions	https://www.intel.com/content/www/us/en/developer/articles/technical/intel-avx-512-instructions.html?wapkw=avx-512
RFC2544 - Benchmarking Methodology for Network Interconnect Devices	https://www.rfc-editor.org/rfc/rfc2544
What is VPP?	https://wiki.fd.io/view/VPP/What_is_VPP%3F
Introduction to DPVS	https://github.com/iqiyi/dpvs
Keepalived User Guide	https://keepalived.org/doc/

2 Overview

HDSLB-VPP is a software L4 Load Balancer Project under an Intel Proprietary License (IPL). It aims to build a performance-leading load balancer in industry, reaching 150 Mpps level throughput, 1000 M level concurrent connections, 10 M level CPS (Connection Per Second) per node, and outstanding linear scalability. HDSLB-VPP customizes a light infrastructure from Fd.io/VPP, which fully utilizes VPP's advantages such as feasibility, extensibility, rich network stack, and software license independence, and develops rich load balancing features based on advanced IA features. Key features are deeply optimized on Intel mainstream platforms (i.e., 3rd Gen Intel® Xeon® Scalable processor, 4th Gen. Intel® Xeon® Scalable processor, Intel® Xeon® D processor, and more), covering Intel® Advanced Vector Extensions 2 (Intel® AVX2 , Intel® Advanced Vector Extensions 512 (Intel® AVX-512) , Intel® Ethernet Flow Director (Intel® Ethernet FD), and more.

It is delivered as a licensed product with the best practice for deploying load balancer solutions on IA platforms. HDSLB-VPP is a powerful solution to extend from the existing load balancer customers (top tier CSPs, CoSPs) to edge customers such as network security, telecom, and enterprise vendors. Figure 1 shows the functional support and architecture of HDSLB.

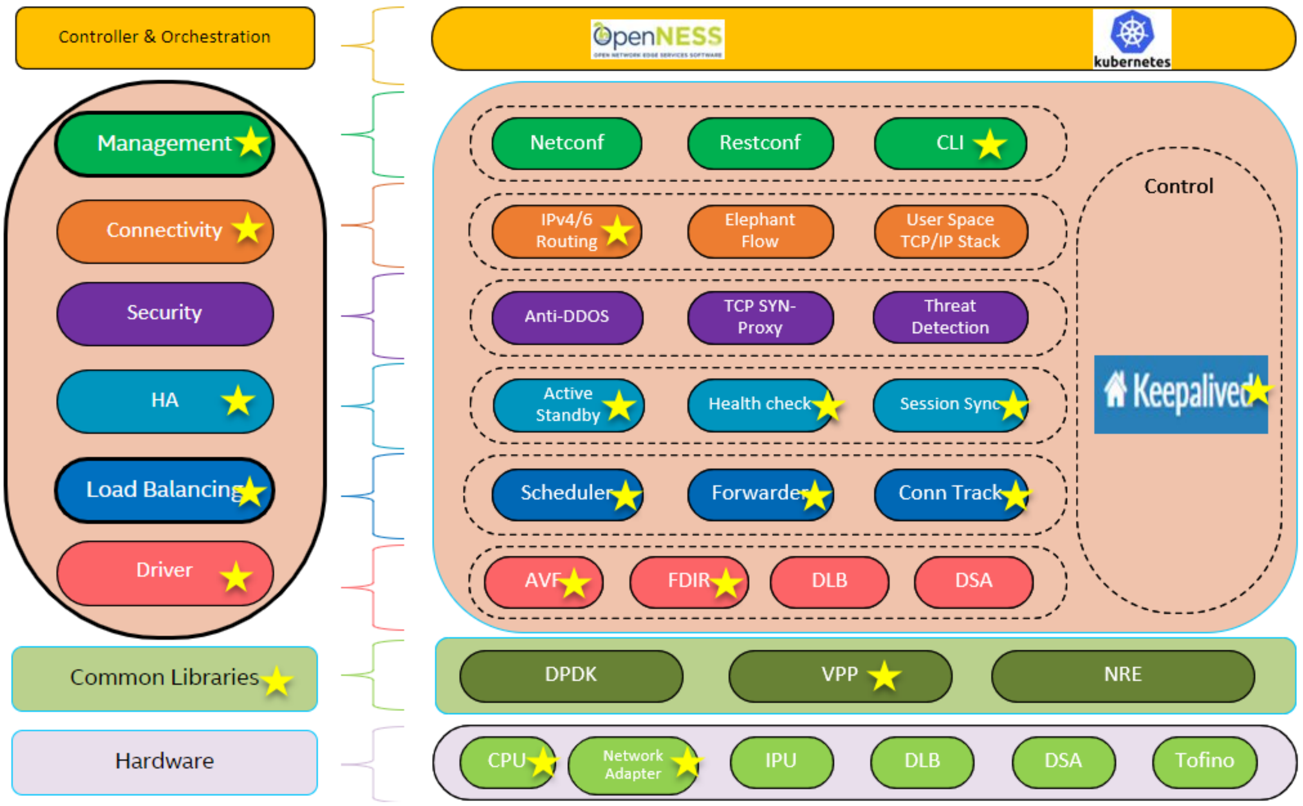


Figure 1. HDSLB-VPP Overview

3 HDSLB-VPP Overall Architecture

Vector Packet Processing (VPP) is a framework that builds the foundation for network data plane processes. Furthermore, Intel has dedicated efforts to accelerate VPP with Intel technologies. HDSLB-VPP is based on the VPP framework, which has high performance, modularity, and rich functions. VPP provides a comprehensive ipv4/ipv6 dual stack protocol that allows us to quickly implement customized L3 functions. Layer 4 load balancer as a key feature of HDSLB-VPP can be easily integrated. HDSLB-VPP conducts some key performance optimizations for LB scenarios using Intel's hardware to achieve industry-leading performance.

The overall software architecture of HDSLB-VPP is divided into the following four major parts:

- **Infrastructure layer**: This layer mainly includes VPP vectorization processing framework, configuration debug interface, memory management, and some high-performance fundamental libraries.
- **Network device layer**: This layer provides the input and output flow of the system. HDSLB-VPP recommends using the VPP Native AVF driver. Compared to the driver of the DPDK plugin, the native driver can achieve better performance. HDSLB-VPP integrates rich Intel hardware acceleration technologies, such as RSS, Checksum Offload, Intel® Ethernet FD, and other functions of the Intel® Ethernet 800 Series Network Adapters, and vectorization instructions of the CPU, such as Intel AVX-512.
- **Protocol stack layer:** With load balancing services as the core, this layer provides necessary protocol interactions such as IPv4, IPv6, TCP, UDP (User Datagram Protocol), and supports ICMP (Internet Control Message Protocol) to adapt to more scenarios.
- **Load balancing implementation layer:** HDSLB-VPP is a stateful 4-layer load balancer, therefore, it has a session management function. HDSLB-VPP provides rich load balancing features such as FNAT/NAT/DR and IPIP encapsulation as shown in Figure 2 as well as supports the SNAT protocol, providing RS with the ability to access

external services. The RS scheduling algorithm supports RR (Round-Robin), WRR (Weighted Round-Robin), WLC (Weighted Least Connection), and Consistent Hash.

In addition to excellent load balancing capabilities, in terms of high availability, it supports active standby mode, session synchronization, and health checks.

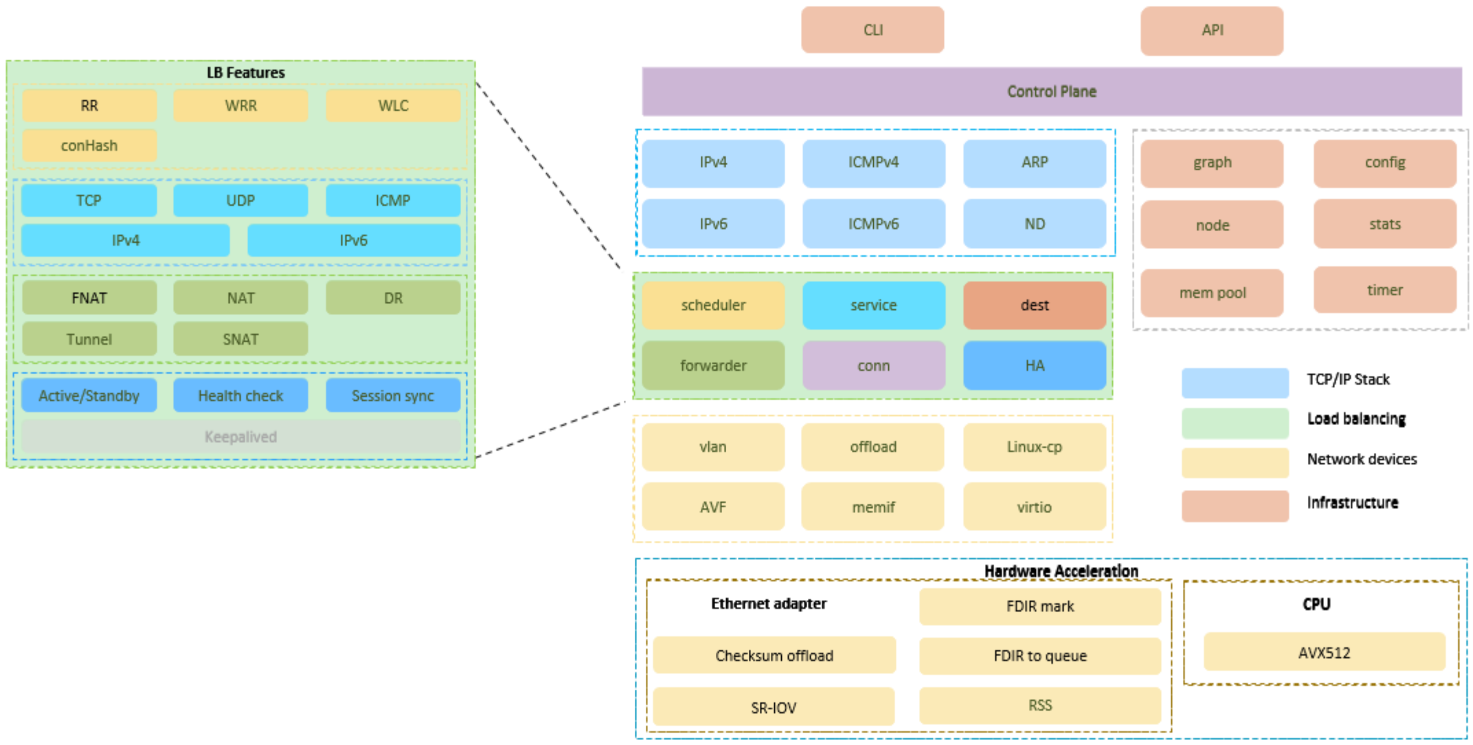


Figure 2. HDSLB-VPP Overall Architecture

3.1 Intel® Ethernet FD Configuration Processing

HDSLB-VPP uses the mark function of Intel® Ethernet FD. When users configure a LB service, some information is needed from the user: the protocol (IPv4/IPv6, TCP/UDP, etc.), IP address, and layer 4 port number. HDSLB-VPP uses this information to configure Intel® Ethernet FD . After this configuration completes, Intel® Ethernet FD hardware compares the destination IP address, destination port, and the corresponding protocol with the configuration specified by the user and marks the traffic with the same traffic. HDSLB-VPP hands over the traffic to the load balancer module as soon as the driver receives it, and the load balancer module can also save some processing steps by utilizing the already-marked traffic during the processing stage.

3.2 Data Plane Processing

HDSLB-VPP is accelerated by Intel's hardware acceleration technologies. For platforms without these hardware acceleration technologies, HDSLB-VPP can also be deployed using software to simulate the processing of Intel® Ethernet FD , though the performance is lower in this case. This part only introduces the Data Plane processing flow with Intel hardware acceleration technologies.

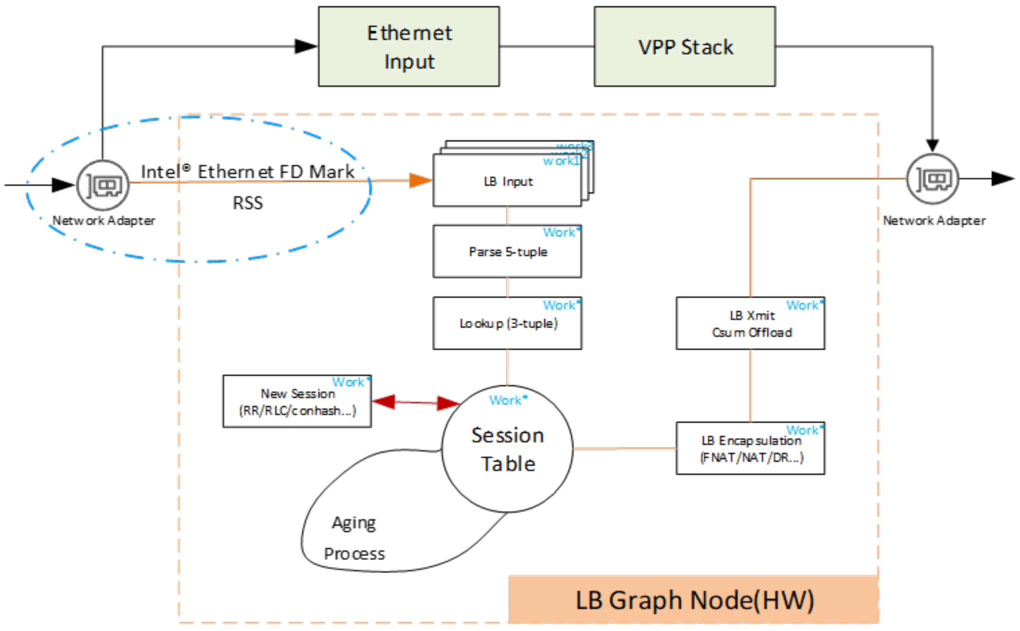


Figure 3. HDSLB-VPP Data-Plane Processing Flow

After HDSLB-VPP receives a load balancer service configured by the user, it enables Intel Ethernet FD to mark the LB traffic. The marked LB traffic is directly sent to the LB node for processing. Traffic that is not marked is sent to the 'Ethernet Input' node of VPP, such as ARP, routing protocol, or other control plane traffic. All unmarked traffic is sent to the protocol stack of VPP itself for processing.

Traffic is distributed using RSS in multi-core systems. After network data is received, the traffic first enters the LB input node of the LB module as shown as Figure 3. LB input node uses the batch cache to accelerate the network traffic processing. The purpose of cache information is to avoid discrete memory access during the process as much as possible to reduce cache miss.

Next, a lookup is performed of the session table. If the session is not found, a new session is created. While creating a new session, the real server is selected based on the user-configured scheduling algorithm (RR, WRR, WLC, ConHash, etc.). After creating the session, the data packet is encapsulated according to the forwarding rule (NAT, FNAT, etc.). Finally, the packet is sent to the underlying TX hardware.

3.3 Performance Optimization

Compared to using DPDK plugin in VPP, HDSLB-VPP uses the AVF plugin to receive and send packets. To improve performance, the AVF plugin is not necessary to convert DPDK 'rte_mbuf_t' structure to VPP 'vlib_buffer_t' structure, which saves time in structure conversion. At the same time, by utilizing Intel® Ethernet FD function of Intel® Ethernet 800 Series Network Adapters, the traffic for LB is tagged, which can be directly used in the AVF plugin to distinguish traffic and send it directly to the node of the LB for processing the next step.

In VPP, there are already graph nodes that implement the LB function. Considering that the native processing procedure is more complex, and the traffic must pass through more nodes, it has an impact on performance and cannot fully utilize the optimization of the IA platform. In the implementation, HDSLB-VPP rearranges the graph nodes to optimize forwarding performance through a small number of graph nodes.

Inside the LB plugin, the LB node uses vectorization processing of VPP to optimize cache-miss. Through prefetch operations, it optimizes session hash tables and real server objects, and performs five steps of distributed prefetching, increasing d-cache and i-cache hits ratios, and improves stability in zero packet loss and 0.001% packet loss tests.

Finally, to achieve high performance, most LB scenarios in HDSLB-VPP are implemented as lockless designs, and each CPU core only accesses local data. Compared to other platform implementations, VPP uses an indexed data structure for indexing, which greatly contributes to memory lookup and space saving.

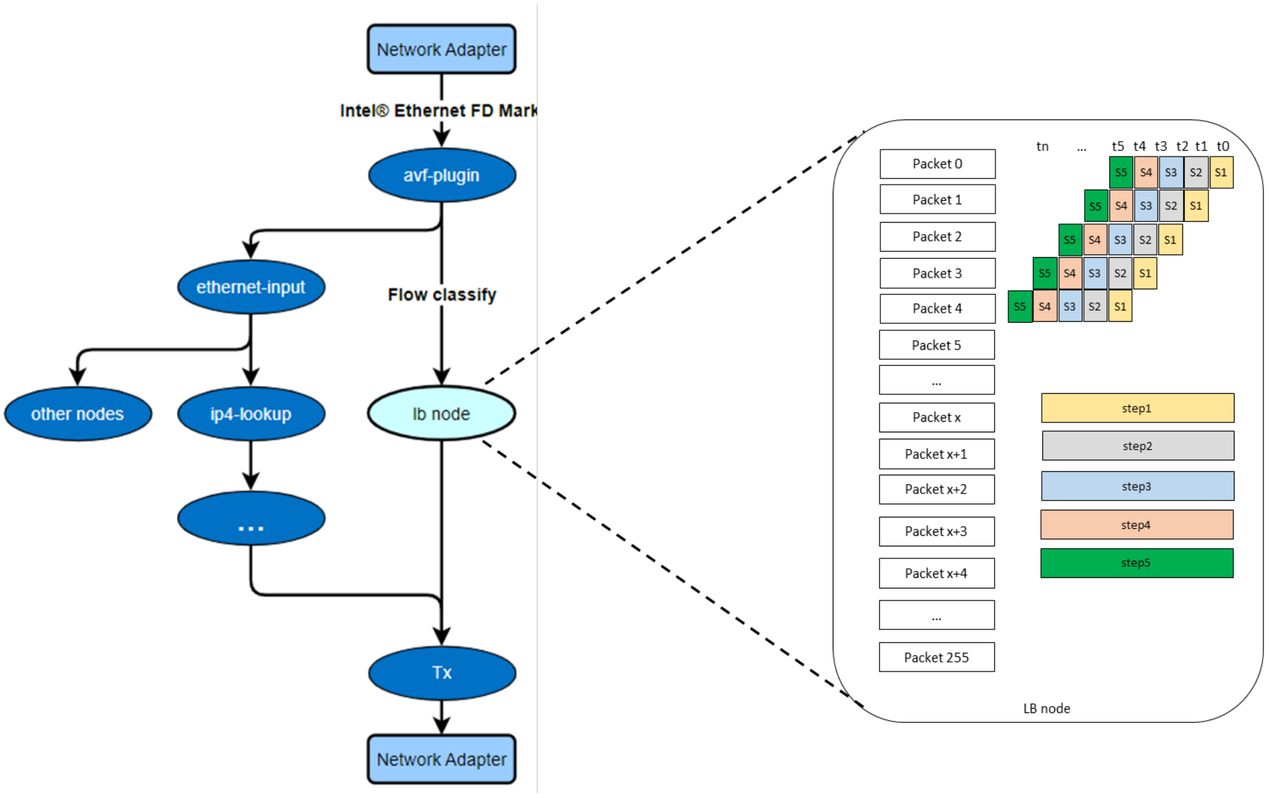


Figure 4. The HDSLB-VPP Data Plane Forwarding Process Flow

3.4 High Availability Framework

In HDSLB-VPP, Keepalived is used to implement the HA function. Keepalived is a widely used solution in industry, is implemented based on the VRRP protocol, and has complete HA function coverage.

When active/standby switching and health check processing begins, Keepalived must synchronize changes in the HDSLB-VPP forwarding engine status. For example, when the active and standby status starts to switch, the VPP and Keepalived states remain consistent. Alternatively, when the health check status of the user's server (real server) changes, the VPP and Keepalived both add or remove the real server from their respective cluster configurations. HDSLB-VPP utilizes the notify script of Keepalived to automatically call the corresponding script when the Keepalived state changes. Through the command interface reserved by HDSLB-VPP, the forwarding engine can synchronously change the state.

The session synchronization feature is due to its deep binding with LB business, which is achieved by implementing independent session synchronization nodes within the LB module. HDSLB-VPP uses UDP packets for periodic synchronization of sessions, while sensing the busy level of the LB forwarding node and automatically adjusting the number and period of synchronized sessions to maintain consistent session status between the primary and backup devices.

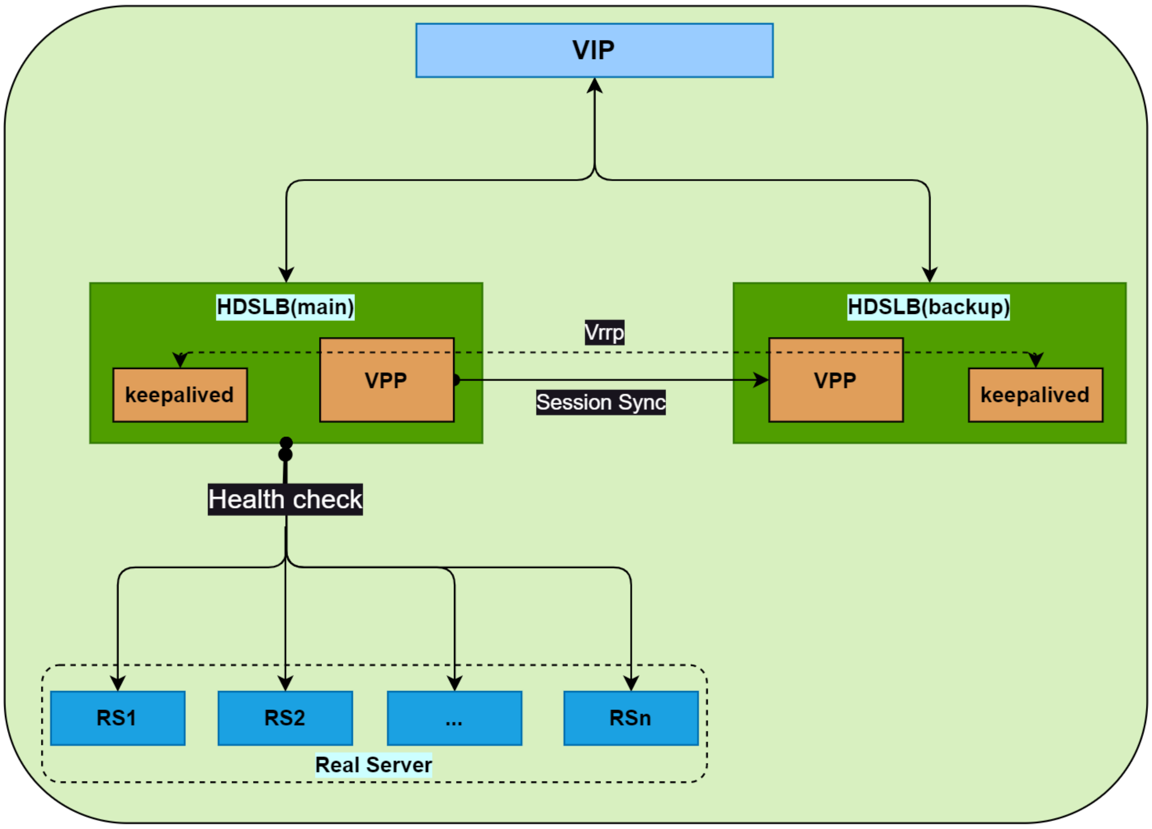


Figure 5. High Availability Framework

4 Performance Advantage

Almost all the data center ingress traffic is processed by the load balancers, so the importance of performance is self-evident. HDLSB-VPP shows an industry-leading performance in the benchmark of throughput and CPS (connections per second).

4.1 Throughput

The throughput benchmark of HDSLB-VPP is based on the 3rd Gen Intel Xeon Scalable processor platform. Figure 6 shows the test topology. The DUT (Device Under Test) runs HDSLB-VPP. An Ixia traffic generator is used to simulate client and real servers. In the performance benchmark test, we use a board card (including two ports) from the Ixia tester to simulate the transmission and reception of traffic. We use one Ixia port to simulate the client to generate huge flows and traffic, and use another Ixia port to simulate real servers to receive the processed traffic. The final test result is measured by Ixia. Three modes (MAX, NDR, PDR) of results are recorded from Ixia report with HDSLB-VPP worker core number 1, 2, 4, 8, and 16.

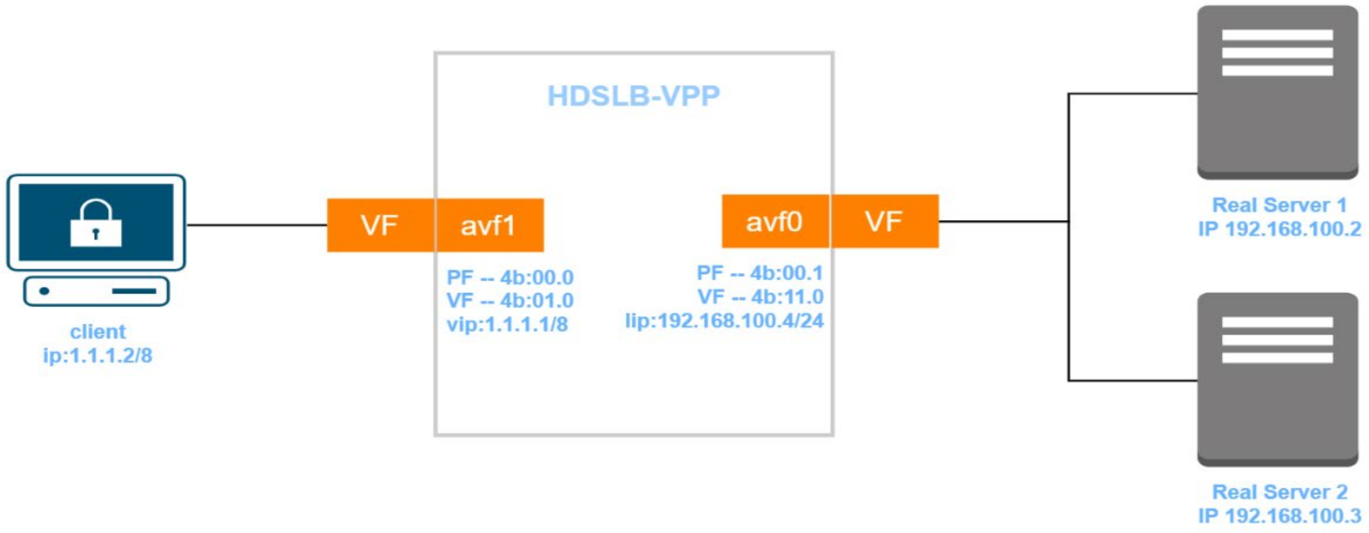


Figure 6. Two-Armed Throughput Test

4.1.1 Platform Configuration and Topology

In the throughput test, the DUT server is an Intel® Xeon® Platinum 8380 processor. HDSLB-VPP only uses one socket, which is installed with 8x32G memory. Some system configurations are recommended for the HDSLB-VPP performance benchmark, such as enabling SR-IOV and INTEL IOMMU. The configuration details of HDSLB-VPP for this test are shown in the Table 3.

Table 3. Throughput Test Platform Configuration

Item	Description
Server Platform	3rd Gen Intel® Xeon® Scalable processor
CPU	Intel® Xeon® Platinum 8380 CPU @ 2.30 GHz
Memory	512 G: 32 GB x 16 DIMMs x 2 NUMA nodes @ 3200 MHz
Network Adapter	2x Intel® Ethernet Network Adapter E810-CQDA2; PCIe 3.0/4.0 x16
Network Adapter Driver and Firmware	Driver: ice Version: 1.9.11 Firmware Version: 4.00 0x80010eb3 1.3236.0
Operating System	Ubuntu 20.04.4 LTS
Linux Kernel Version	5.4.0-148-generic
HDSLB-VPP Version	23.04 (base on VPP 20.04)
Turbo	OFF
Numa Balancing	0
Transparent Hugepage	0
CPU Power	Performance
SR-IOV	Enable
Intel IOMMU	ON

The deployment method is shown in Figure 7. Two 100 GbE Ixia ports are directly connected to two 100 GbE ports of one Intel® Ethernet Network Adapter directly.

In the throughput test, the Ixia tester generates traffic to create connections for each flow before recording the benchmark result. After all connections are created, Ixia starts to measure the packets per-second results of mode PDR, NDR, and MAX.

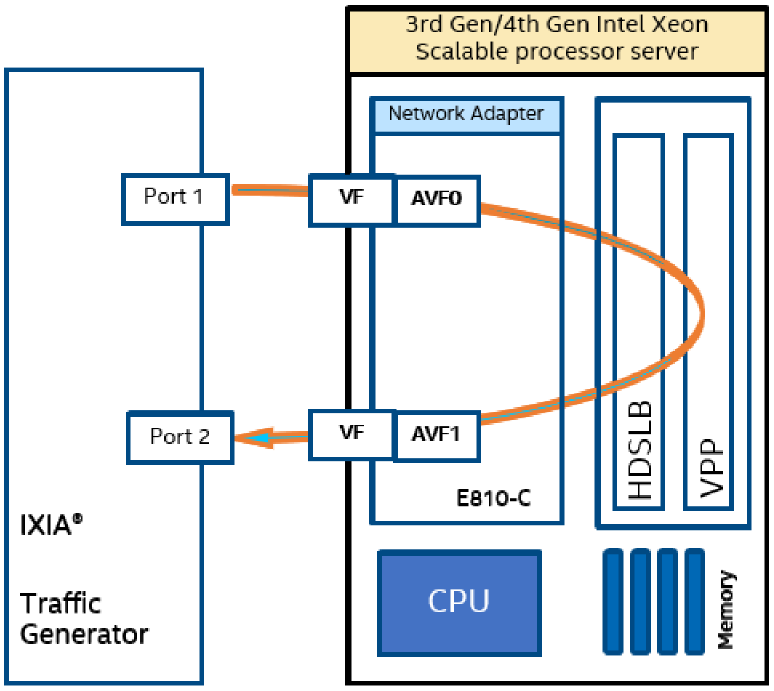


Figure 7. Throughput Test Topology

To evaluate the performance and stability of throughput, MAX, NDR and PDR throughput result are recorded:

- **MAX**: Record the maximum forwarding PPS of HDSLB-VPP without paying attention to packet loss rate.
- **PDR**: Partial packet drop rate, tested using RFC2544 with 0.001% packet loss rate.
- **NDR**: No packet drop rate, zero packet loss in RFC2544 test.

The smaller the performance difference between MAX, NDR, and PDR, the better the stability of HDSLB-VPP.

Table 4 describes the detailed test matrix.

Table 4. Test Description

Item	Test Parameter
Forward Mode	FNAT / NAT / DR / IPIP
LB Scheduling Algorithm	RR/WRR/WLC
Protocol	IPv4-UDP64/ IPv6-UDP80
Packet Loss Rate	MAX/PDR/NDR
Core	1C / 2C / 4C / 8C / 16C
Sessions	10M-Per Core

4.1.2 Performance Results

4.1.2.1 Single Core Performance

Compared to multiple modes, the single core throughput performance of HDSLB-VPP is around 10 Mpps. Moreover, compared to NDR and MAX, the performance is very stable. The DR mode has the highest performance due to its relatively simple processing flow while NAT-SW has the lowest performance. NAT-SW is a completely software-based implementation solution that does not rely on hardware acceleration.

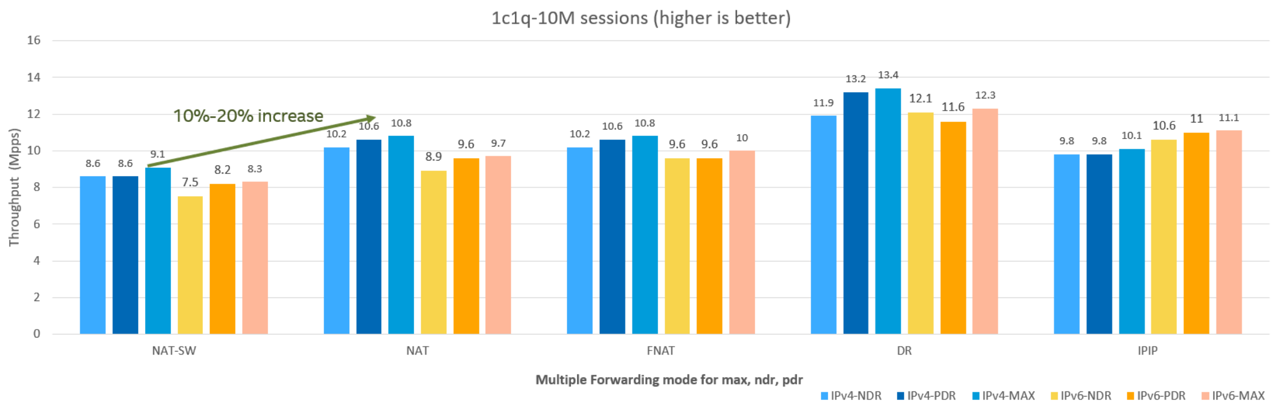
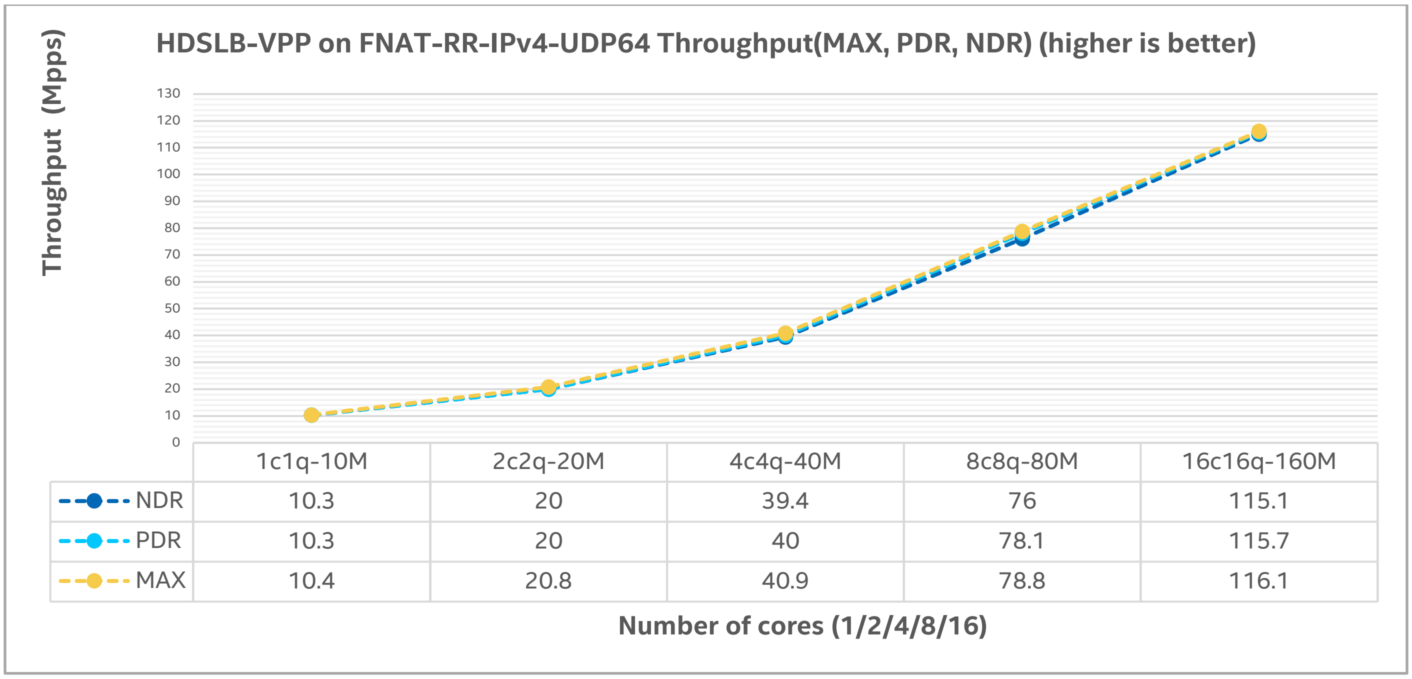


Figure 8. Multiple Forwarding Mode for MAX, NDR, PDR of Single Core

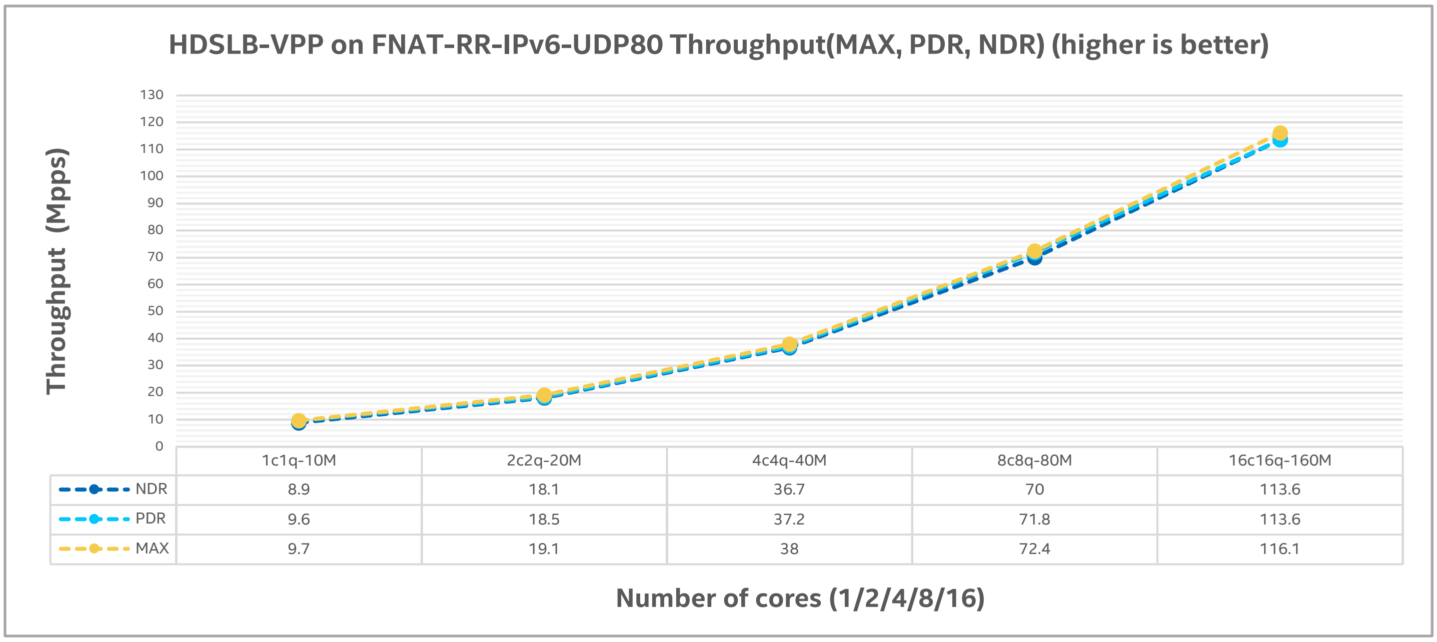
4.1.2.2 Core Scalability at 64B/80B

In a multi-core configuration, the most important concern is the maximum forwarding capability. Taking FNAT mode as an example, in a 16-core configuration, HDSLB-VPP achieves a maximum forwarding PPS performance of 116 Mpps, which is also the maximum capability of a single Intel Ethernet Network Adapter E810-CQDA2 (100 Gbps) network card. As the following figures show, the performance of HDSLB-VPP has excellent linear scaling performance as the core increasing.



	1c1q-10M	2c2q-20M	4c4q-40M	8c8q-80M	16c16q-160M
NDR	10.3	20	39.4	76	115.1
PDR	10.3	20	40	78.1	115.7
MAX	10.4	20.8	40.9	78.8	116.1

Figure 9. HDSLB-VPP on FNAT-RR-IPv4-UDP64 Throughput (MAX, PDR, NDR)

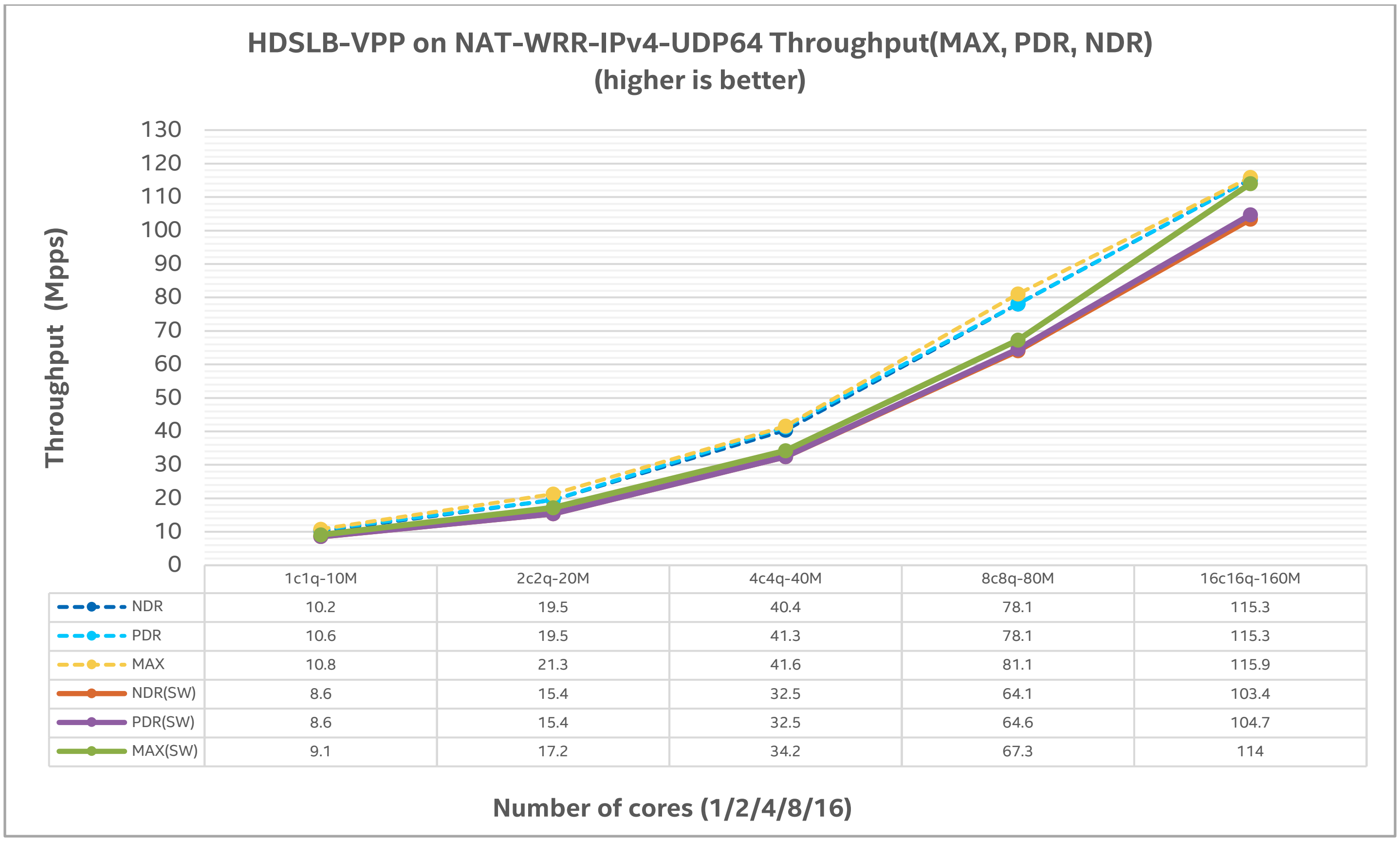


	1c1q-10M	2c2q-20M	4c4q-40M	8c8q-80M	16c16q-160M
NDR	8.9	18.1	36.7	70	113.6
PDR	9.6	18.5	37.2	71.8	113.6
MAX	9.7	19.1	38	72.4	116.1

Figure 10. HDSLB-VPP on FNAT-RR-IPv6-UDP80 Throughput (MAX, PDR, NDR)

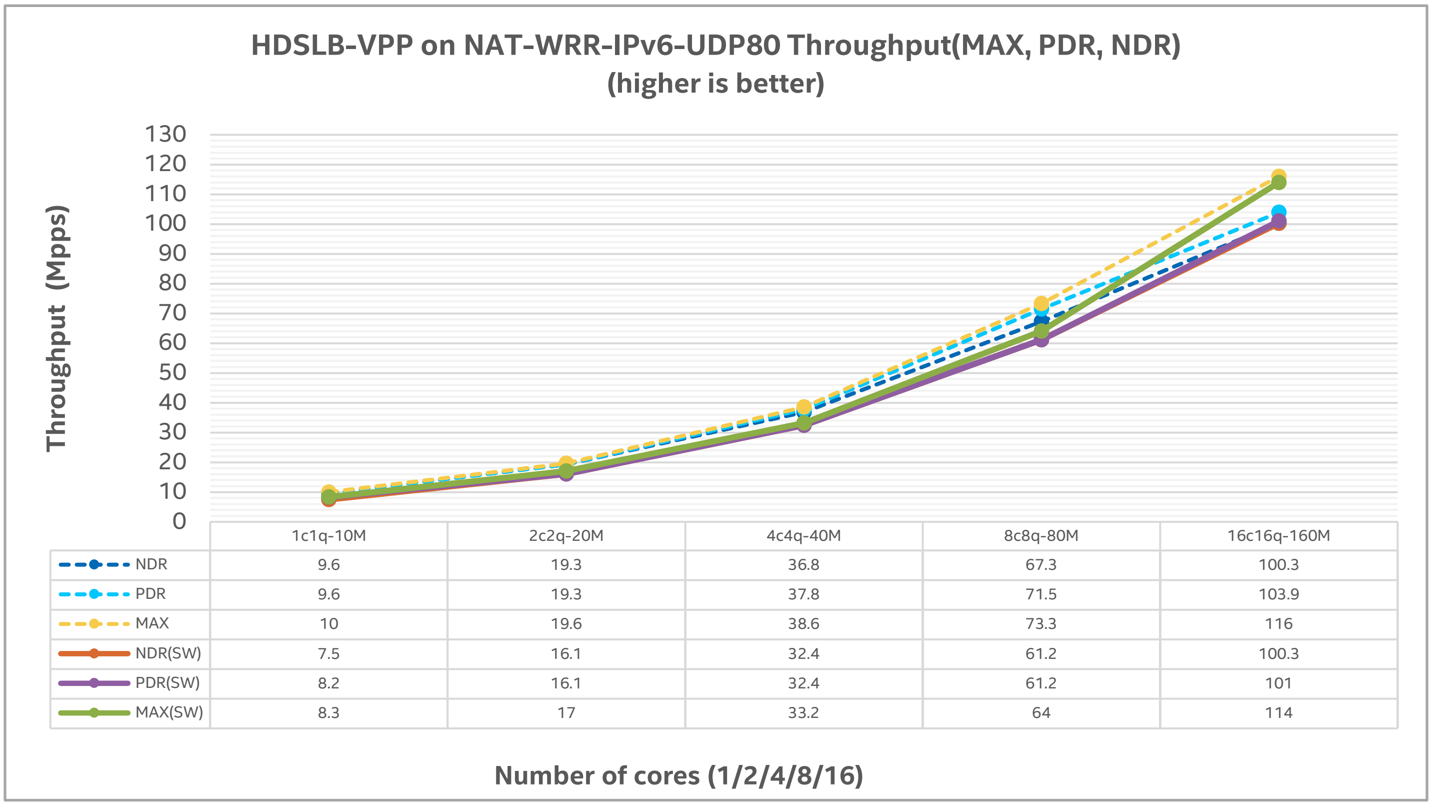
4.1.2.3 Impact of Hardware Acceleration

To illustrate the impact of hardware acceleration on HDSLB-VPP performance, we compared the performance data of hardware acceleration and software-based implementation in NAT mode. In NAT mode, the performance of a single core drops by 10-20% without physical network adapter acceleration, and as the number of cores increases, the performance approaches that of NAT (HW). The performance trends of IPv4 and IPv6 are consistent, and in the 16-core NDR test scenario, the performance of HW-NAT needs to be optimized.



	1c1q-10M	2c2q-20M	4c4q-40M	8c8q-80M	16c16q-160M
NDR	10.2	19.5	40.4	78.1	115.3
PDR	10.6	19.5	41.3	78.1	115.3
MAX	10.8	21.3	41.6	81.1	115.9
NDR(SW)	8.6	15.4	32.5	64.1	103.4
PDR(SW)	8.6	15.4	32.5	64.6	104.7
MAX(SW)	9.1	17.2	34.2	67.3	114

Figure 11. HDSLB-VPP on NAT-WRR-IPv4-UDP64 Throughput (MAX, PDR, NDR)



	1c1q-10M	2c2q-20M	4c4q-40M	8c8q-80M	16c16q-160M
NDR	9.6	19.3	36.8	67.3	100.3
PDR	9.6	19.3	37.8	71.5	103.9
MAX	10	19.6	38.6	73.3	116
NDR(SW)	7.5	16.1	32.4	61.2	100.3
PDR(SW)	8.2	16.1	32.4	61.2	101
MAX(SW)	8.3	17	33.2	64	114

Figure 12. HDSLB-VPP on NAT-WRR-IPv6-UDP80 Throughput (MAX, PDR, NDR)

4.1.2.4 Memory Consumption

In the throughput test, we recorded the memory consumption of HDSLB-VPP in NAT mode. The session data structure in other modes is the same as NAT mode. In the following figure, the memory consumption for different numbers of cores is recorded with 16 M sessions per core. In the case of 256 M sessions, only 70 G of memory is consumed. With a target of measuring the maximum sessions, we use 32 M and 64 M sessions per core after 16 cores. In Figure 13, the result shows that the HDSLB-VPP can achieve 1024 M sessions with 184 G of memory. These results reflect the advantages of HDSLB-VPP in terms of resource utilization.

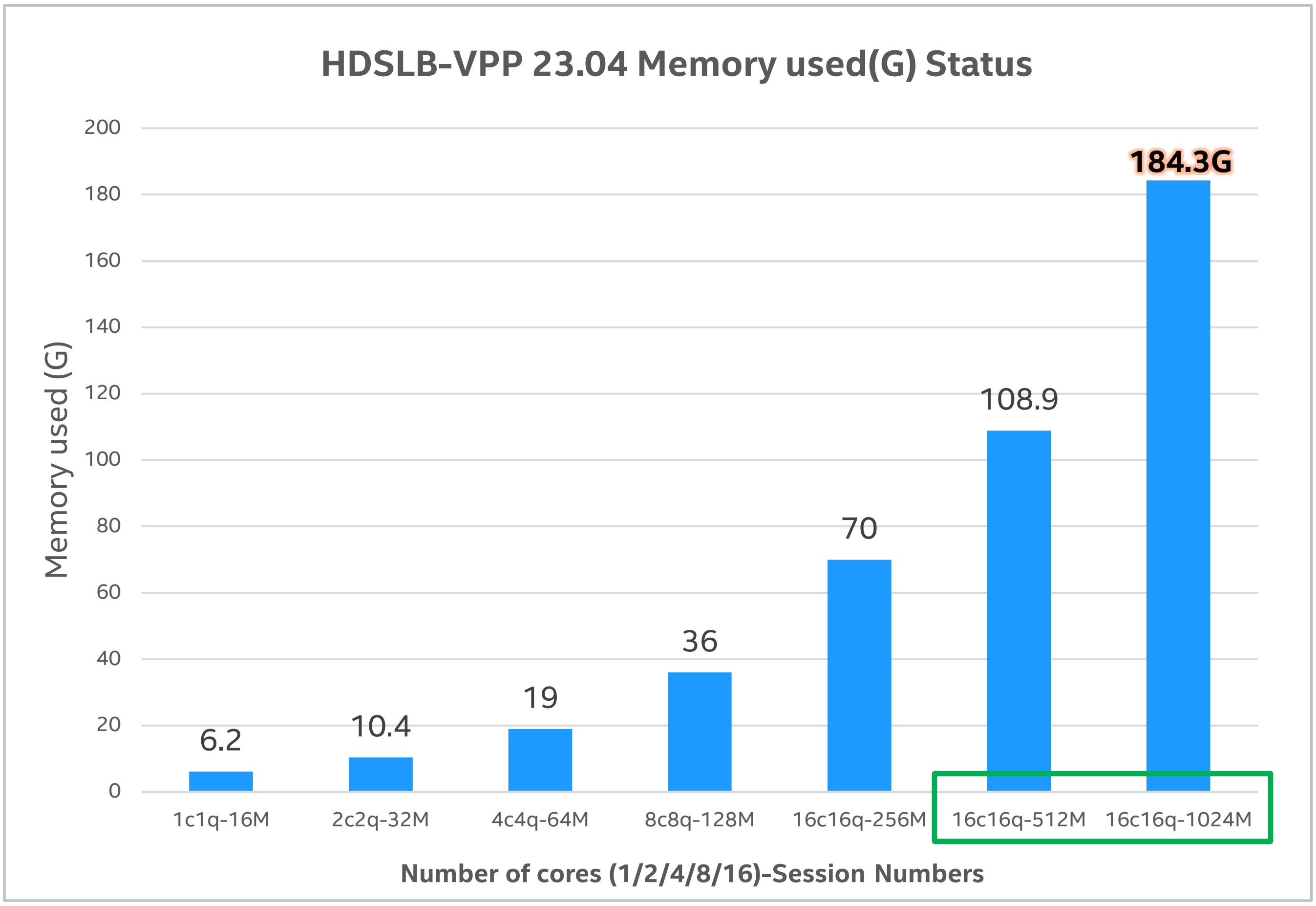


Figure 13. Memory Usage and Session Concurrency

The optimization of memory and the advantages of concurrent session volume in HDSLB-VPP make great practical value in future scenarios where IPv6 is becoming increasingly common and mature. Under the same resource configuration, HDSLB-VPP can support more session volume, effectively improving single node capacity to cope with larger address space of IPv6.

4.2 Connections Per Second

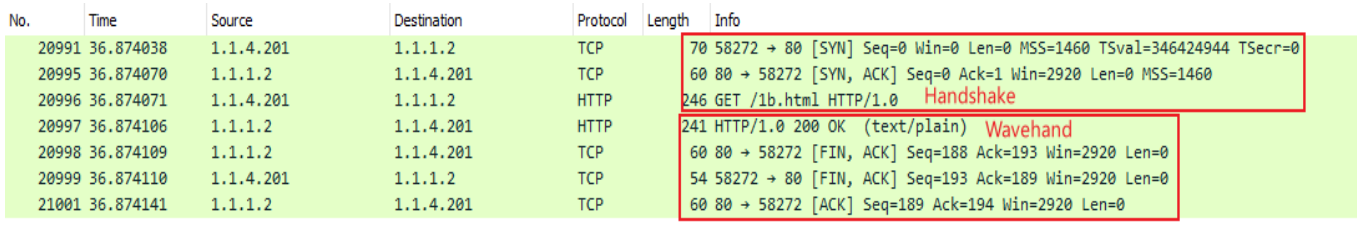
To show HDSLB CPS performance and scalability on Intel® Ethernet 800 Series Network Adapters under 1/2/4/8/16 core, Intel collaborated with Keysight to measure the recordable level TCP CPS performance of HDSLB-VPP on 3rd Gen Intel Xeon Scalable processors. HDSLB-VPP uses 1-Byte size page to reduce page request overhead and simulate a complete TCP and HTTP lifecycle with three-handshakes and four-way-handshake. Note that in this test, HDSLB-VPP deletes the connection after four-way-handshake to better match the user's actual usage scenario.

Table 5. CPS Test Description

Item	Configuration
Client	Action: GET http://DUT1:80/1b.html Protocol Level: HTTP 1.0 IP Mappings: Cycle Users Through All Source IPs IP Number: 1200 / IXload Sustain Times: 310s Ramp Down Time: 10s
Server	Protocol Level: HTTP 1.0 Connection Termination With: The last ACK from Server

	Response Size: 1 Byte

The process of creating a complete HTTP connection is shown in Figure 14.



```
No.    Time       Source     Destination  Protocol  Length  Info
 20991 36.874038  1.1.4.201  1.1.1.2      TCP       70 58272 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 TSval=346424944 TSecr=0
 20995 36.874070  1.1.1.2    1.1.4.201    TCP       60 80 → 58272 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0 MSS=1460
 20996 36.874071  1.1.4.201  1.1.1.2      HTTP     246 GET /1b.html HTTP/1.0
 20997 36.874106  1.1.1.2    1.1.4.201    HTTP     241 HTTP/1.0 200 OK  (text/plain)
 20998 36.874109  1.1.1.2    1.1.4.201    TCP       60 80 → 58272 [FIN, ACK] Seq=188 Ack=193 Win=2920 Len=0
 20999 36.874110  1.1.4.201  1.1.1.2      TCP       54 58272 → 80 [FIN, ACK] Seq=193 Ack=189 Win=2920 Len=0
 21001 36.874141  1.1.1.2    1.1.4.201    TCP       60 80 → 58272 [ACK] Seq=189 Ack=194 Win=2920 Len=0
```

Figure 14. A completely new connection contains three-way-handshake and four-way-handshake.

4.2.1 Platform Configuration and Topology

The test of CPS and throughput uses the same DUT with the same system configuration.

Table 6. CPS Test Platform Configuration

Item	Description
Server platform	3rd Gen Intel Xeon Scalable processor
CPU	Intel® Xeon® Platinum 8380 CPU @ 2.30 GHz
MEMORY	480 G: 32 GB x 16 DIMMs x 2 NUMA nodes @ 3200 MHz
Network Adapter	2x Intel® Ethernet Network Adapter E810-CQDA2; PCIe 3.0/4.0 x16
Network Adapter Driver and Firmware	Driver: ice Version: 1.9.11 Firmware Version: 4.00 0x80010eb3 1.3236.0
Operating System	Ubuntu 20.04.4 LTS
Linux Kernel Version	5.4.0-148-generic
HDSLB-VPP Version	23.04
Turbo	OFF
Numa Balancing	0
Transparent hugepage	0
CPU Power	Performance
SR-IOV	Enable
Intel IOMMU	ON

A single network card test cannot reach the performance limit of the DUT, so we finally test scenario case of 32 cores on one socket with two network cards.

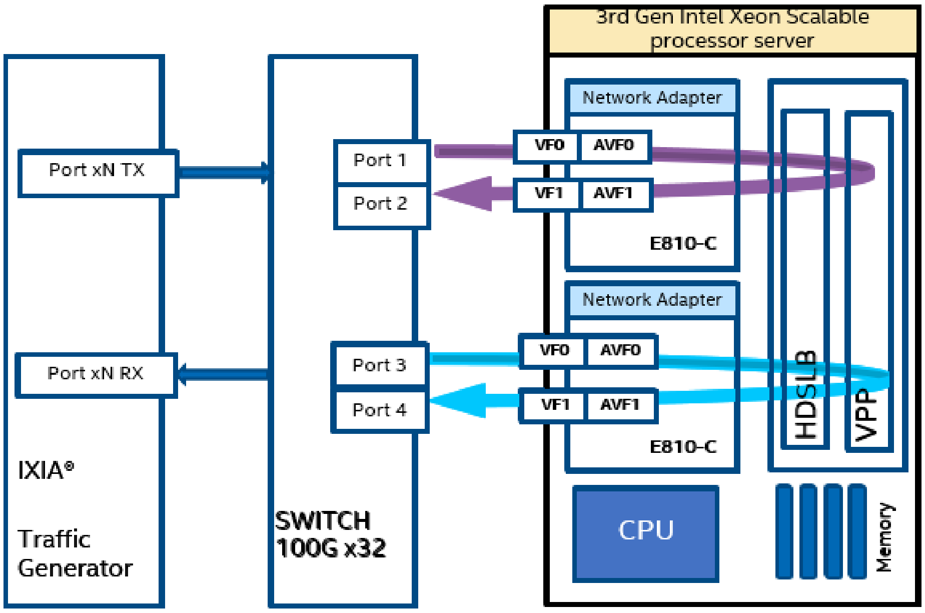


Figure 15. CPS Test Topology

The key CPS test configurations for HDSLB-VPP are as follows:

Table 7. CPS Test Description

Item	Test Parameters
Forward Mode	FNAT / NAT
LB Scheduling Algorithm	RR
Protocol	IPv4-TCP/ IPv6-TCP
Core	1C / 2C / 4C / 8C / 16C / 32C
Network Adapter	1 x Network Adapter /2x Network Adapter

4.2.2 Performance Result

During the test, we find that 16 cores nearly reach the network adapter's limitation, thus we add the scenario of 32 cores with two network adapters to get the maximum performance of HDSLB-VPP.

As shown in Figure 16, the maximum CPS can achieve up to 21.8 M with two network adapters and 32 cores in IPv4-NAT mode. On the benchmark with 32 cores on IPv4-NAT, comparing single network adapter with two network adapters, the performance increases from 14.9 M to 21.8 M. In addition, from the IPv4/IPv6 perspective, whether it is NAT or FNAT, IPv4 performance is slightly better than the performance of IPv6. Compared with the IPv4-NAT HW acceleration mode, IPv4-NAT-SW performance has dropped about 40%.

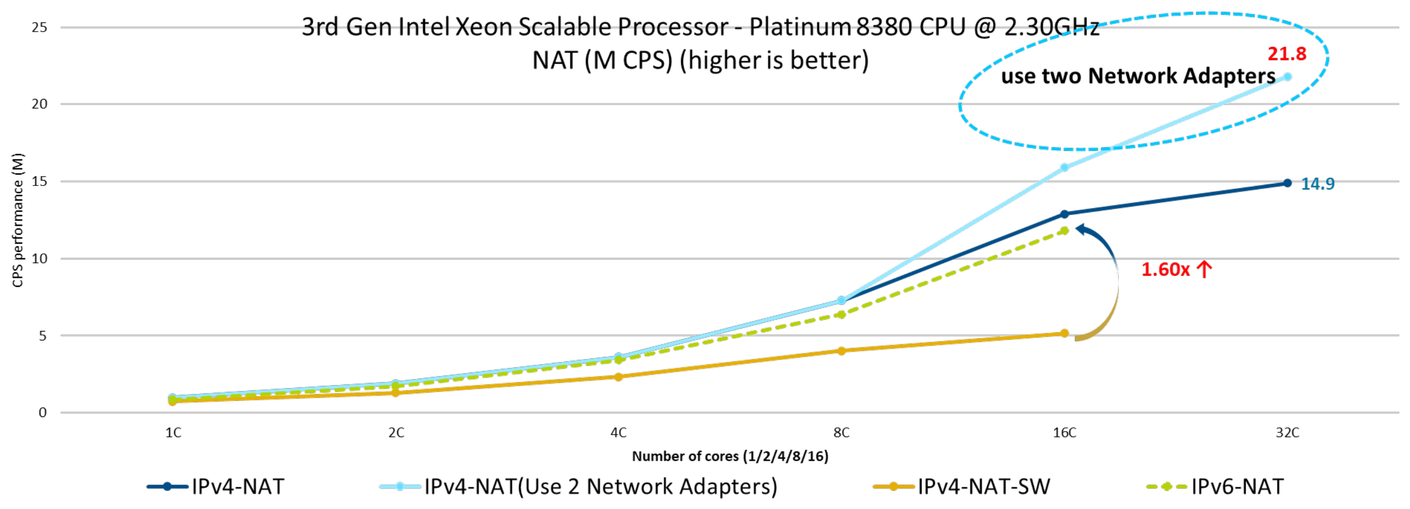


Figure 16. CPS Performance Data on 3rd Gen Intel Xeon Scalable Processor

5 Summary

As cloud and edge deployments become the norm, networking has evolved to become a key workload with load balancer as a fundamental network function. This trend evolves performance requirements such as higher throughput, CPS, and higher-level concurrent connections (at least 100 M). HDSLB-VPP is a targeted solution as a layer 4 load balancer that responds to these new challenges with high throughput on IA platforms.

In the latest HDSLB VPP L4 release 23.04, we provide a complete solution with rich load balancer features and outstanding performance on IA platform.

- Support rich load balancer features (NAT, FNAT, DR, WLC, etc.), HA, and so on
- Single core throughput can reach to 10 Mpps
- Maximum throughput can reach to network adapter line rate 116 Mpps per 16 cores with good core scaling
- Concurrent connection reaches to recordable 1000 M level with less memory consumption
- CPS reaches over 20 M(million) cps, 21 Mcps on 3rd Gen Intel® Xeon® Scalable processor platform

Appendix A Configuration

A.1 Hardware Configuration

HDSLB-VPP Config1 (3rd Gen Intel Xeon Scalable processor) -1-node, pre-production platform with 2x Intel® Xeon® Platinum 8380 with Intel AVX-512 on Intel M50CYP2SBSTD motherboard with GB (16 slots/ 32 GB/ DDR4 3200) total memory, ucode 0xd000389, HT off, Turbo off, Ubuntu 20.04.1 LTS, 5.4.0-0-146-generic, 1x 465.8G CT500MX500SSD1, 1x 447.1G INTEL_SSDSC2KW48, 2x Intel® Ethernet Controller I350, 4x Intel Ethernet Controller E810, gcc version 11.4.0, tested by Intel on 04/03/2023.

A.2 Software Configuration

Table 8. Software Configuration

Item	HDSLB-VPP
Workload and version	HDSLB-VPP 23.04
Compiler	Clang10.0.0
Libraries	N/A
Traffic Generator Type	Ixia
Traffic Generator Version	ixnetwork 9.20.2112.6 ixLoad 9.20.2112.6
Packet Loss Setting (%)	NDR/PDR/MAX
Number of IP Flows	10M Per core
Workload Config	HDSLB-VPP
Other Software Configuration	VPP 22.02
Network Adapter Firmware	3.0
Network Adapter Driver	Ice 1.9.11
Run Method: ex. cold (fresh-boot), warm (post-boot after few back to back iterations)	Warm
Iterations and result choice (median, average, min, max)	3 iterations, average
Dataset Size	64B(IPv4) / 80B(IPv6)
Operating Core Frequency	3rd Gen Intel Xeon Scalable processor 2.3GHz

1123/DN/WIT/PDF 793188-001US