WHITE PAPER

Intel Corporation

Intel Platform Service Assurance - Platform Policy Enabling Resource Management

Authors

Wojciech Andralojc

John Browne

Tomasz Kantecki

1 Introduction

Network Function Virtualization (NFV) enables the consolidation of a wide variety of communication appliances. To meet NFV input/output performance requirements, production Virtual Network Functions (VNFs) must perform to a high standard. However, these production VNFs can be impacted by a third-party VNF that is using an unfair amount of system resources, causing a performance decrease in production VNFs. These third-party VNFs are generally known as *Noisy Neighbors*. Specifically, competition for and contention over processor cache resources are a well-established cause of noisy neighbor conditions.

To help alleviate the impacts of noisy neighbors in such scenarios, Intel provides Intel® Resource Director Technology (Intel® RDT) and a tool called Platform Quality of Service (PQoS). One of the main parts of Intel® RDT is Cache Allocation Technology (CAT), a component that allows system administrators and developers to gain control of cache memory.

The capabilities that Intel processors integrate to deal with the potential resource contention issues associated with NFV are described and discussed in an earlier paper titled *Performance Evaluation of Cache Allocation Technology for NFV Noisy Neighbor Mitigation*. This paper describes dynamic partitioning encompassing both guaranteed partitions and fixed partitions.

To get optimal performance in a CAT-enabled environment, it is critical to allocate an adequate number of Cache Ways (CWs) for each VNF. If the number of CWs is too small, the VNF performance and overall system performance may degrade. If the number of CWs is too large, that can be considered a waste of resources. It is important to keep in mind that a VNF's cache demand may differ depending on the load placed on it. Therefore, static cache allocation may be suboptimal in cases where the load is changing over time. On the other hand, if load changes are sudden (for example, going from no load to fully loaded) dynamic cache allocation may lead to transient packet loss. This is more likely to happen if a large amount of a production VNF's cache was reallocated to a third-party VNF. There is a tradeoff to make between a production VNF's transient packet loss and a third-party VNF's performance gain. The user must decide if transient packet loss is acceptable and how high it can be and configure the system accordingly.

The purpose of this paper is to explore the dynamic resource partitioning of Last Level Cache (LLC) using the CAT component of Intel® RDT and verify if there are currently available statistics that allow the detection of a VNF's current cache demand and implement the Platform Policy Agent that would allow fully-automated Dynamic Resource Partitioning (DRP).

This document is part of the Network Transformation Experience Kit, which is available at: https://networkbuilders.intel.com/

Table of Contents

Figures

Tables

1.1 Intended Audience

This white paper is for architects and developers who want to implement Dynamic Resource Partitioning (DRP) between high-priority production VNFs and best effort VNFs (exhibiting noisy neighbor behavior) without affecting the performance of the production VNFs.

1.2 Terminology

Table 1 describes the acronyms used in this white paper.

Table 1. Terminology

ABBREVIATION	DESCRIPTION
ACL	Access Control List
CAT	Cache Allocation Technology
CBM	Contention Bit Mask
CPU	Central Processing Unit
CW	Cache Way
DDR	Double Data Rate
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
DRP	Dynamic Resource Partitioning
ESP	Encapsulating Security Payload
HTTP	Hyper Text Transmission Protocol
I/O	Input/output
IPSec	Internet Protocol Security
KVM	Kernel Virtual Machine
LLC	Last Level Cache
LPM	Longest Prefix Match
NFV	Network Function Virtualization
OS	Operating System
OvS*	Open vSwitch*
PID	Process identifier
PMD	Pole Mode Driver
PQoS	Platform Quality of Service
QEMU	Quick Emulator
SA	Security Association
SD-WAN	Software Defined Wide Area Network
SFP	Small Form-factor Pluggable
SP	Security Policy
SUT	System Under Test
TCP	Transmission Control Protocol
VNF	Virtual Network Function
VT-d	Intel® Virtualization Technology for Directed I/O
WAN	Wide Area Network
XML	Extensible Markup Language

1.3 Reference Documents

Table 2 provides links to documentation related to the concepts in this white paper.

Table 2. Reference Documents

REFERENCE	SOURCE
AESN-NI Multi Buffer Crypto Poll Mode Driver	https://doc.dpdk.org/guides/cryptodevs/aesni_mb.html
Data Plane Development Kit (DPDK)	http://dpdk.org/
DPDK IPSec Security Gateway Sample Application	https://doc.dpdk.org/guides-16.04/sample_app_ug/ipsec_secgw.html
Intel® Multi-Buffer Crypto for IPsec Library	https://github.com/intel/intel-ipsec-mb
Intel® RDT	https://github.com/01org/intel-cmt-cat
Open vSwitch (OvS)	http://openvswitch.org/
Performance Evaluation of Cache Allocation Technology for NFV Noisy Neighbor Mitigation	https://ieeexplore.ieee.org/abstract/document/8004214
Quick Emulator (QEMU)	http://www.qemu-project.org/
Processor Counter Monitor (PCM) tool	https://github.com/opcm/pcm

1.4 Technologies

1.4.1 Intel® Resource Director Technology (RDT)

Intel® Resource Director Technology (Intel® RDT) brings new levels of visibility and control over how shared resources such as last-level cache (LLC) and memory bandwidth are used by applications, virtual machines (VMs), and containers. It is the next evolutionary leap in workload consolidation density, performance consistency, and dynamic service delivery, helping to drive efficiency and flexibility across the data center while reducing overall total cost of ownership (TCO). As software-defined infrastructure and advanced resource-aware orchestration technologies increasingly transform the industry, Intel® RDT is a key feature set to optimize application performance and enhance the capabilities of orchestration and virtualization management server systems using Intel® Xeon® processors.

Intel® RDT provides a framework with several component features for cache and memory monitoring and allocation capabilities, including CMT, CAT, CDP, MBM, and MBA. These technologies enable tracking and control of shared resources, such as the Last Level Cache (LLC) and main memory (DRAM) bandwidth, in use by many applications, containers or VMs running on the platform concurrently. Intel® RDT may aid "noisy neighbor" detection and help to reduce performance interference, ensuring the performance of key workloads in complex environments.

More information can be found at https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html

1.4.1.1 Cache Allocation Technology (CAT)

Software-guided redistribution of cache capacity is enabled by CAT, enabling important data center VMs, containers or applications to benefit from improved cache capacity and reduced cache contention. CAT may be used to enhance runtime determinism and prioritize important applications such as virtual switches or Data Plane Development Kit (DPDK) packet processing apps from resource contention across various priority classes of workloads.

1.4.1.2 Memory Bandwidth Monitoring (MBM)

Multiple VMs or applications can be tracked independently via Memory Bandwidth Monitoring (MBM), which provides memory bandwidth monitoring for each running thread simultaneously. Benefits include detection of noisy neighbors, characterization and debugging of performance for bandwidth-sensitive applications, and more effective non-uniform memory access (NUMA)-aware scheduling.

1.4.2 Platform Quality of Service (PQoS) Toolkit

The PQoS toolkit contains software library package and command line tools that allow easy access to Intel® RDT on Linux* and FreeBSD* operating systems. This software is available through Linux packaging systems like 'dnf' or 'apt-get' and directly from the GitHub* project web site: https://github.com/intel/intel-cmt-cat. Part of the package is a "pqos" tool that allows you to monitor and manage resources at run time and it works across wide range of Linux kernel versions including those that don't have native Intel® RDT support.

1.4.3 Platform Policy Agent

The Platform Policy Agent is a software component that allows Dynamic Resource Partitioning (DRP) between a high-priority production VNFs' group and a Best Effort VNFs' group. The high-priority group should be optimized for best performance with no impact from cache and memory hungry applications (noisy neighbor type) in the best effort group. In the case of a low load on a high-priority group, unused cache resources should be reallocated to the best effort group. Examples of such agents are Open Day Light Honey Comb or Nova Compute in OpenStack*.

2 Platform Policy Enabling Use Case

To demonstrate the usage of Intel® RDT technologies in this document, we used an SD-WAN application to simulate a real-world use case. SD-WAN is an acronym for software-defined networking in a wide area network (WAN). SD-WAN simplifies the management and operation of a WAN by decoupling (separating) the networking hardware from its control mechanism.

The SD-WAN used in our example is composed of an IPSEC gateway combined with a Deep packet inspection component, representing an SD-WAN solution composed of a firewall and routing functions. This function combination would be typically used at the edge of customer premises.

2.1 SD-WAN and Best Effort VNFs Overview

An SD-WAN is a specific application of software-defined networking technology applied to a Wide Area Network (WAN). The SD-WAN connects enterprise networks that include branch offices and data centers over large geographic distances as shown in Figure 1.

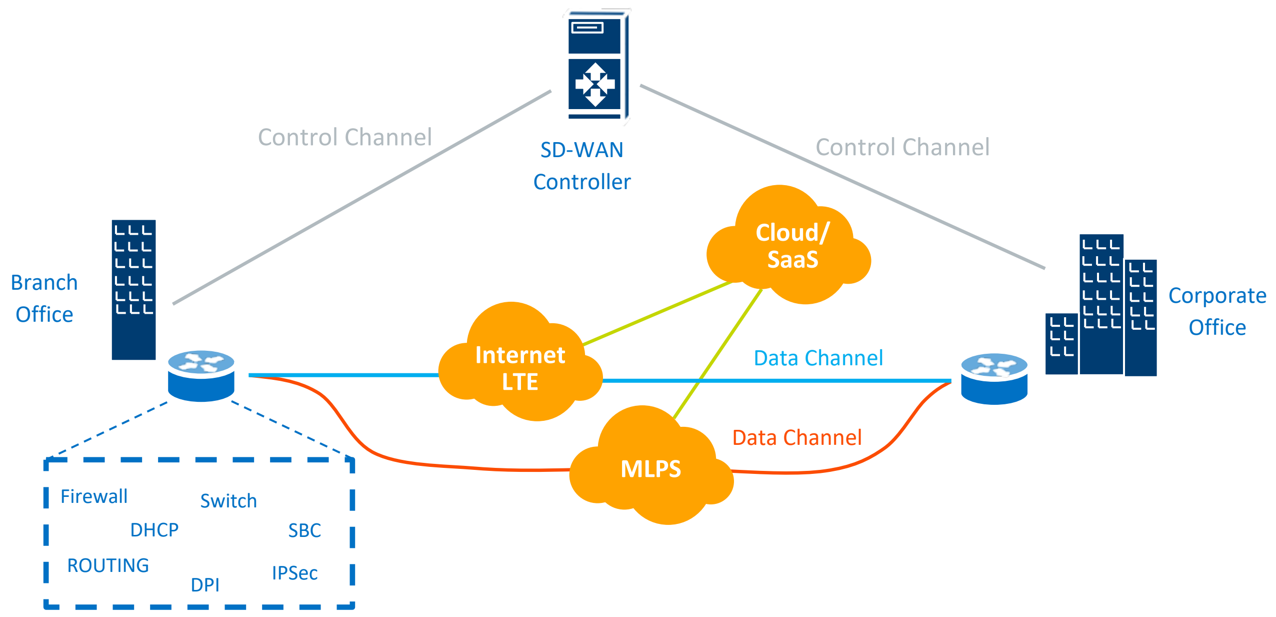


Figure 1. SD-WAN High-Level Overview

2.1.1 System Under Test (SUT) for SD-WAN Replication

Figure 2 shows a high-level view of the System Under Test (SUT) described in this white paper that replicates an SD-WAN. There are two instances of the platform, each on their own physical and identical machines. Each instance is a mirror of the other. Traffic flow is as follows:

- On the uplink, viewing from left to right, the pipeline includes a Deep Packet Inspection (DPI) Virtual Machine (VM) and an Internet Protocol Security (IPSec) VM. In this pipeline, packets are inspected in the DPI VM and passed to the IPSec VM for encryption.
- On the downlink, packets only pass through the IPSec VM for decryption. There is no need for DPI because the packets have already been inspected in the DPI VM on the uplink.

A virtual switch, called an Open Virtual Switch (OvS), connects the VMs and the IXIA Traffic Generator provides traffic for the tests that we describe later.

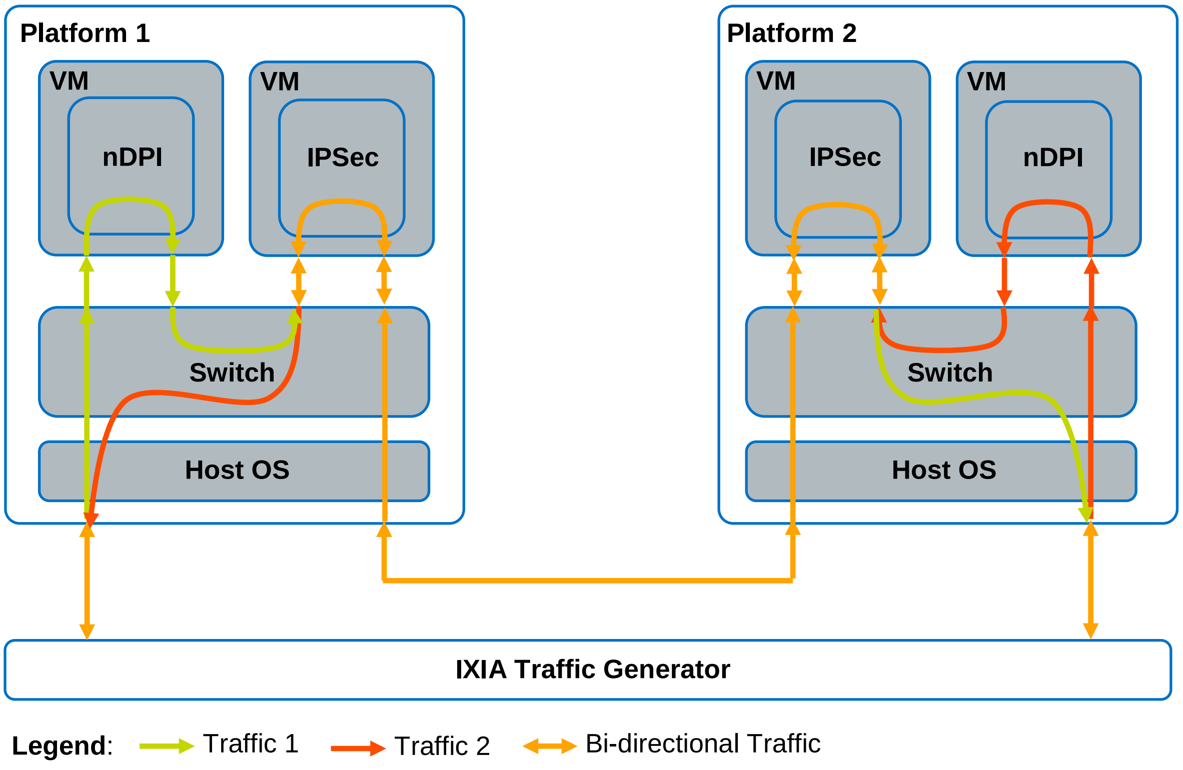


Figure 2. SUT (SD-WAN) Setup

This setup helps determine if a best effort VNF, or *noisy neighbor* (introduced in the next section), causes a performance decrease for high-priority production VNFs and if the Reliability Agent can restore or protect them.

2.1.2 SUT (SD-WAN) with Best Effort VNF Introduced

Figure 3 introduces a Best Effort (or Noisy Neighbor) VNF into the SUT setup. Running a process that is cache- and memory-intensive in the Best Effort VNF causes a performance decrease and packet loss in the overall SD-WAN. This is verifiable by examining the packet loss rate reported by the OvS and the IXIA Traffic Generator.

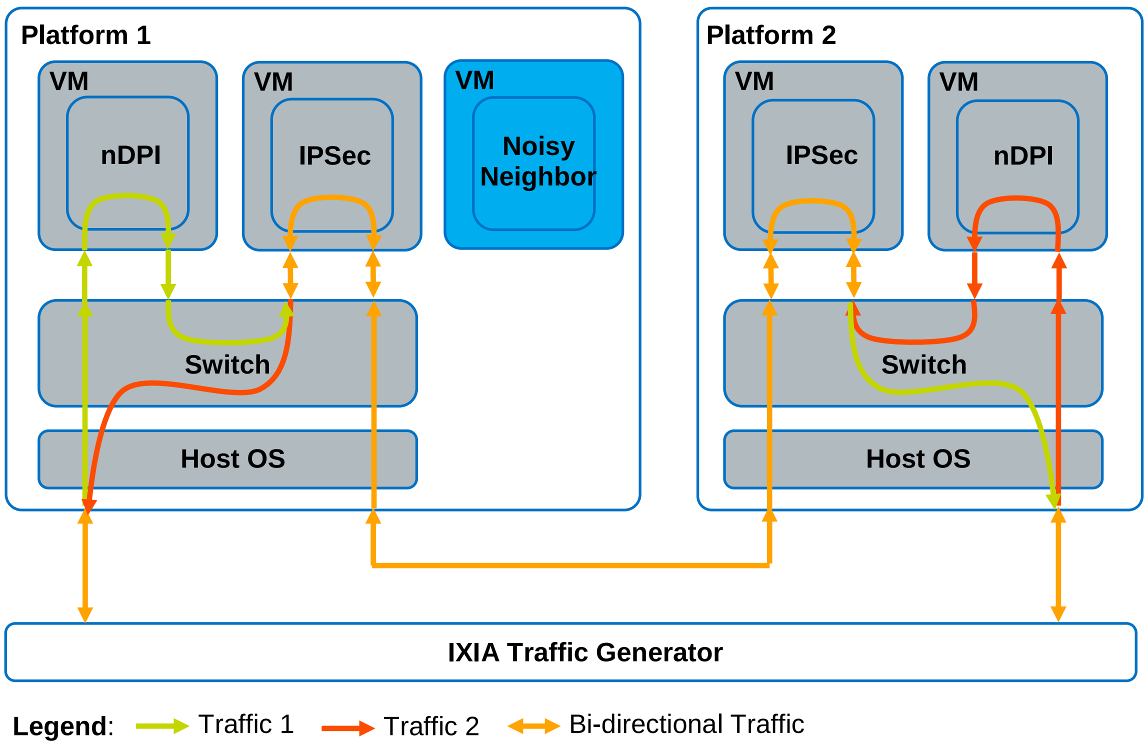


Figure 3. SUT (SD-WAN) with Best Effort (Noisy Neighbor) VNF Setup

The system configuration details can be found in Section A.6.

3 Sample Platform Policy Agent

The sample platform policy agent described here is a prototype sample software that emulates a fully functional agent.

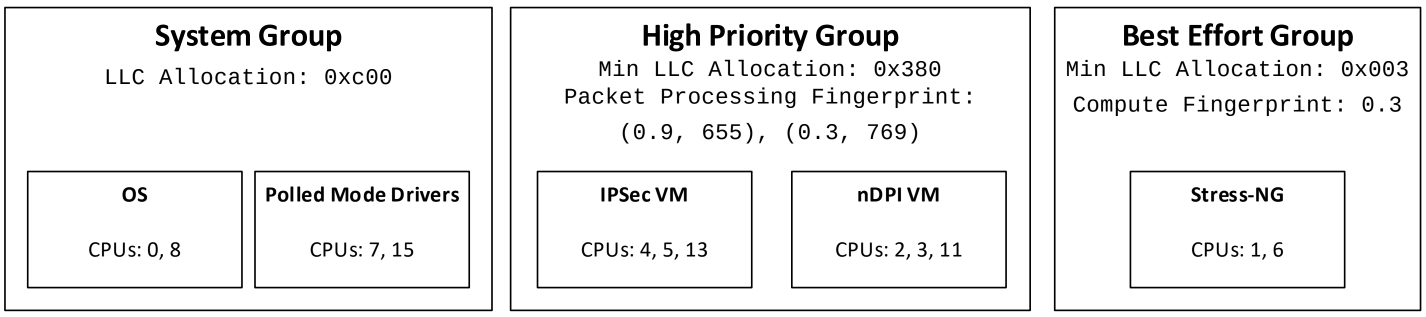
3.1 Groups of VNFs and Components

For effective use of the available cache, VNFs and system components are grouped as shown in Table 3.

Table 3. Groups of VNFs and Components

Group	Type
System	Static
High-priority	Dynamic
Best Effort	Dynamic

The sample policy agent does not modify system group resource allocation. However, it balances resource allocation between high priority and best effort groups depending on the load and resources required by the former group.



```
LLC Allocation: 0xc00
```
OS — CPUs: 0, 8
Polled Mode Drivers — CPUs: 7, 15

```
Min LLC Allocation: 0x380
Packet Processing Fingerprint:
(0.9, 655), (0.3, 769)
```
IPSec VM — CPUs: 4, 5, 13
nDPI VM — CPUs: 2, 3, 11

```
Min LLC Allocation: 0x003
Compute Fingerprint: 0.3
```
Stress-NG — CPUs: 1, 6

Figure 4. Group Configuration with CPU Assignment

LLC allocation is expressed as a contiguous bitmask, which is the standard way of expressing LLC allocation with Intel® RDT.

A *fingerprint* is a way of characterizing the workload, which may vary depending on its nature.

A packet processing fingerprint includes pairs of numbers that provide a linear approximation of a more complicated curve. This approach simplifies the implementation with a minimal loss of precision. The first number in a pair is the memory bandwidth to packet throughput ratio. The second number in a pair is the packet throughout.

A compute intensive group is characterized by a compute fingerprint that is expressed as memory bandwidth to retired instructions ratio.

Finding values for the fingerprints requires a series of tests and experimentation.

Note: While a low minimum LLC allocation value allows the Best Effort group to benefit more when High-priority is idle, it may cause significant packet drops on a sudden traffic rate increase. This is related to the Platform Policy Agent sampling interval. It is possible that for one full sampling interval, for example, the High-priority group is fully loaded with 12.5% traffic, but has only one CW allocated, which causes significant packet drops.

3.2 Sample Platform Policy Agent Details

The Platform Policy Agent is a software component that allows Dynamic Resource Partitioning (DRP) between a high-priority production VNFs' group and a best effort VNFs' group. The agent observes both high-priority and best effort VNFs' groups memory bandwidth utilization and external performance indicators (for example, throughput and frame loss rate, and retired instructions). Based on those observations, the agent attempts to apply a partitioning scheme that guarantees the optimal VNFs' groups performance. The agent continuously monitors performance indicators so that it can react to changing VNF cache demands.

3.2.1 Monitored Performance Indicators

A VNF's condition is evaluated periodically by analyzing statistics retrieved form the Linux kernel `perf` (per PID) or from the OvS (per OvS interface). Table 4 gives the monitored performance indicators.

Table 4. Monitored Performance Indicators

Indicator	Source	Comment
Memory Bandwidth Utilization	perf	`intel_cqm/total_bytes`
Retired Instructions	perf	`PERF_COUNT_HW_INSTRUCTIONS`
RX and TX Throughputs	ovs	Calculated from: • `tx bytes` • `rx bytes` • `duration`
Packet Drops	ovs	`tx drop`

3.2.2 Algorithm

Figure 5 shows the algorithm used by the Platform Policy Agent. The Platform Policy Agent starts by allocating the maximum possible number of CWs to the high-priority group and flushing the cache of the best effort group using `pid_cache_flush` and the PIDs provided in the configuration file. Then the agent decreases the number of CWs allocated to the high-priority group (which changes high-priority and best effort CWs ratio) until an equilibrium is reached.

When minor packet drops are observed, the number of CWs allocated to the high-priority group is increased by one and cache for the best effort group is flushed.

In the equilibrium state, the agent observes performance indicators. If one of those indicates changes by more than a defined amount or goes out of the expected range, the agent leaves the equilibrium state and starts from the beginning.

The equilibrium state is reached in the following cases:

- The high-priority group's performance indicator, based on memory bandwidth utilization and throughput, reaches the configured threshold.
- The best effort group's performance indicator, based on memory bandwidth utilization and retired instructions count, reaches the configured threshold.

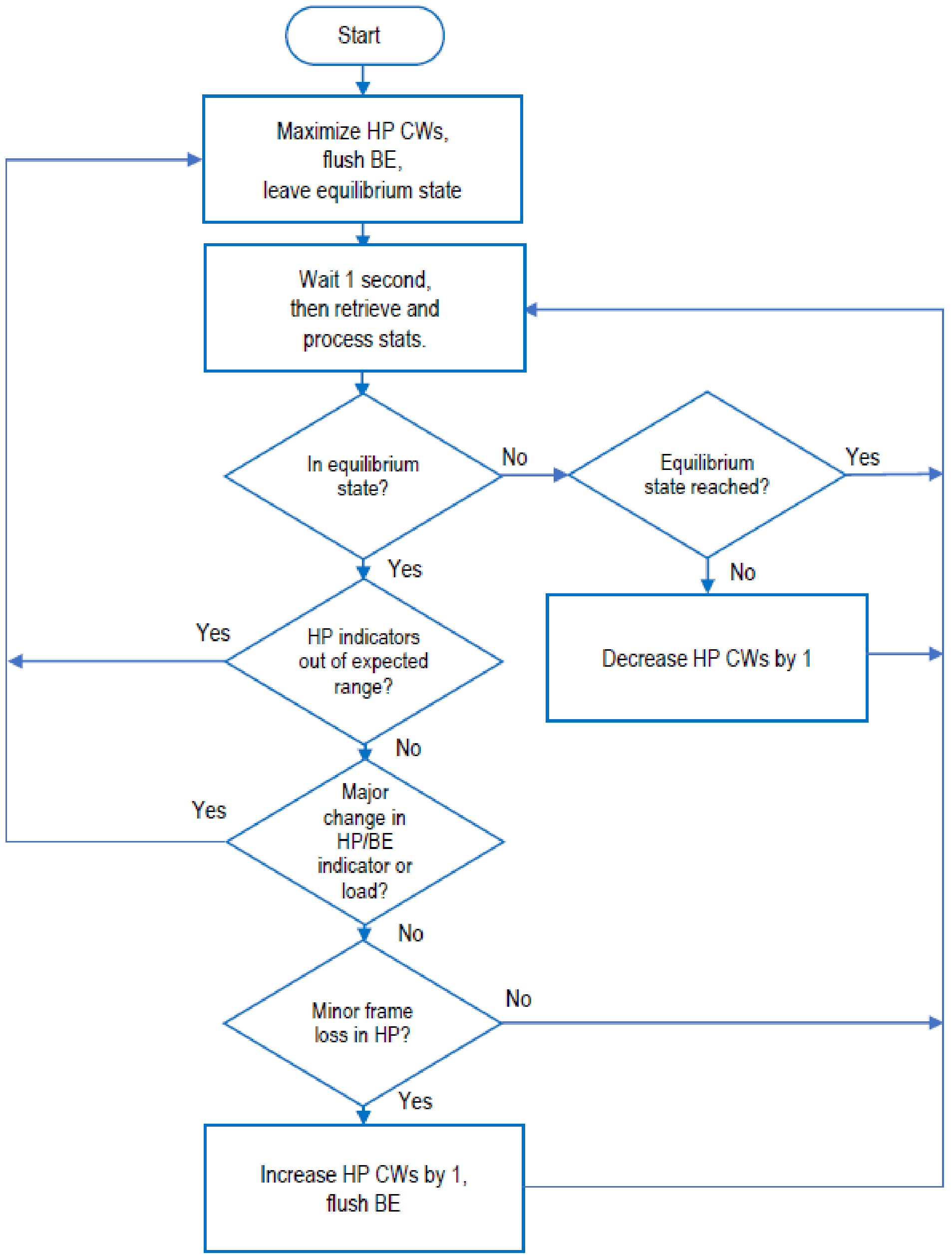

Start
Maximize HP CWs, flush BE, leave equilibrium state
Wait 1 second, then retrieve and process stats.
In equilibrium state?
No
Equilibrium state reached?
Yes
Yes
No
Decrease HP CWs by 1
Yes
HP indicators out of expected range?
No
Yes
Major change in HP/BE indicator or load?
No
Minor frame loss in HP?
No
Yes
Increase HP CWs by 1, flush BE

Figure 5. Platform Policy Agent Algorithm

3.2.3 Cache Ways Partitioning Scheme

The Platform Policy Agent implements dynamic cache partitioning with a guaranteed minimum. A high-priority group has priority access to shared CWs, as shown in Figure 6. Groups can use shared CWs, but a groups' CWs stay isolated. The agent changes the high-priority to best effort CWs ratio. When increasing the high-priority CWs' number, it flushes the best effort cache to evict the best effort group from the CWs that are now allocated to a high-priority group.

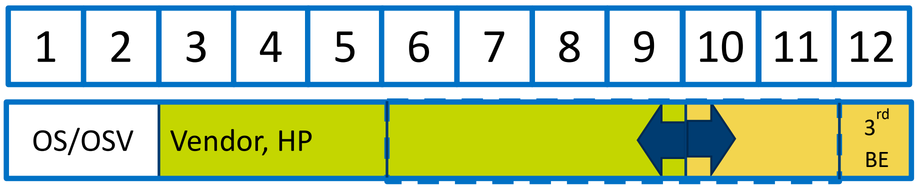


Figure 6. Cache Ways Partitioning Scheme

3.2.4 External Dependencies

Table 5 shows the minimum version of key software packages used in the setup described in this white paper.

Table 5. Minimal Version Requirements

Software Packages	Version	Comments
Linux kernel/`perf`	4.10	`perf` (monitoring) and `pqos` (CAT configuration) interoperability
`pqos`	1.0	
`pid_cache_flush`	-	Internally developed LLC flush kernel module that can flush LLC content belonging to selected PID. This is helpful to make LLC allocation changes take prompt effect.
OvS	Supporting OpenFlow1.3 or 1.4	Requires a `duration` field to be reported by the `ovs-ofctl` command per port, to calculate `throughput`.

3.2.5 CAT Configuration

CAT configuration is set and controlled by the Platform Policy Agent. VNFs and system components are categorized in three groups to use the available cache effectively. The minimum number of CWs per group are set in a configuration file. The system and static group's CAT configuration is statically configured at startup. Depending on the current dynamic groups of VNFs' demand and condition, the Platform Policy Agent changes the High-priority/Best Effort Groups' CWs ratio accordingly.

Table 6 lists possible Contention Bit Mask (CBM) values for the CAT configuration per group.

Table 6. CAT Configuration: OVS/OS, High priority VMs, and Best Effort VM

Group of Components	CBM	Cache Ways Allocated
Static, System (OS, OVS, PMD)	0xC00	2
Dynamic, High Priority VMs (nDPI, IPsec)	0x380 – 0x3FC	3 - 8
Dynamic, Best Effort VM (stress-ng)	0x003 – 0x07F	2 - 7

3.2.6 Cache Sensitivity and Changing Cache Demand

The SD-WAN setup consists of two groups of VNFs:

- High Priority with nDPI and IPSec (packet processing) VNFs
- Best Effort with (computing intensive) VM running `stress-ng`

3.2.6.1 High-Priority Packet Processing Cache Sensitivity

While investigating the packet processing groups' cache sensitivity and cache demand, we limited the number of available CWs to find the minimal number of CWs needed (without incurring frame loss). See Figure 7.

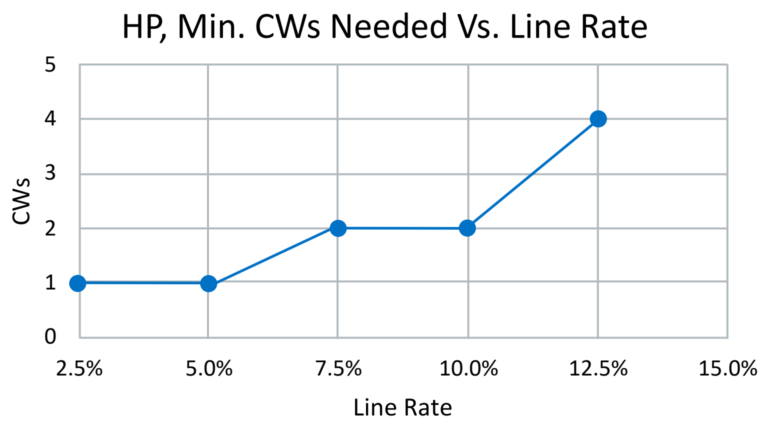


Figure 7. Minimal CWs Needed vs. Line Rate for the High-Priority VNFs Group

The lower the line rate, the smaller the number of CWs needed for the High-priority VNFs group to process packets without frame loss.

3.2.6.2 Best Effort Computing Intensive Cache Sensitivity

While investigating the best effort computing intensive groups' cache sensitivity, we limited the number of available CWs and observed the number of retired instructions.

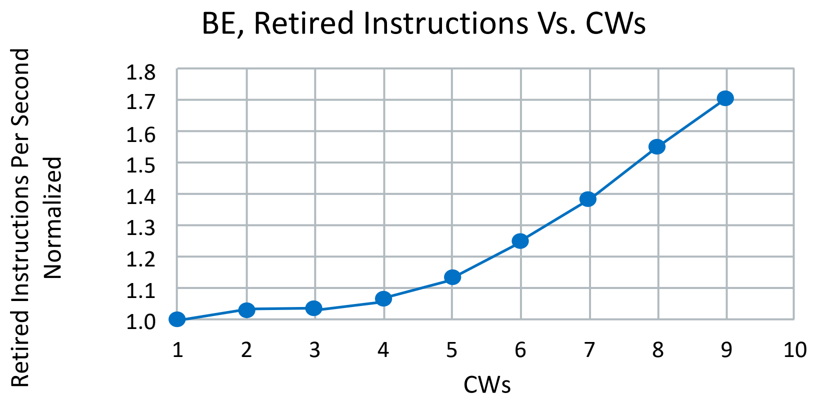


Figure 8. Retired Instruction vs. Allocated CWs for the Best Effort VMs Group

The higher the number of allocated CWs, the better the Best Effort VMs group performance, with more retired instructions reported per second.

3.2.6.3 Evaluation

There seems to be a clear correlation between the line rate being processed by the high-priority VNFs group and its cache demand. Also, there is a correlation between the number of available CWs and the Best Effort VNFs group's performance. By taking both correlations into account when setting up cache partitioning, the high-priority groups' cache demand and the best effort group's cache sensitivity data can enable the protection of the high-priority groups' performance, while allowing the best effort group to gain performance when possible.

It would be convenient and resource-effective if resource partitioning could be done dynamically, depending on actual cache demand. The idea is to monitor a group's memory bandwidth utilization and external performance indicators (for example, throughput, frame loss rate, and retired instructions depending on the type of workload). Then, choose the best partitioning scheme that guarantees optimal VNF performance and cache use. It is crucial to continuously monitor performance indicators to react to changing VNFs' cache demand. This approach enables the effective distribution of available resources between VNFs to guarantee optimal performance at any given time, protecting high-priority group performance while allowing the best effort group to gain performance at the same time.

Figure 9 shows the theoretical best effort group's performance gain if cache is dynamically partitioned while considering high-priority cache demand depending on the line rate. For static partitioning, cache is partition in a fixed way, to allow the high-performance group to handle the maximum line rate of 12.5%. See Section 3.3 for the measured results.

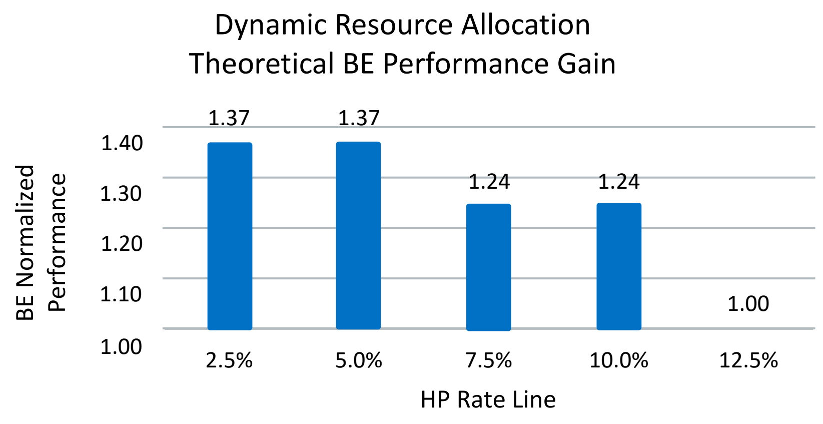


Figure 9. Dynamic Resource Allocation, Theoretical Best Effort VMs Group Performance Gain

§ Configuration: Host Kernel, 4.11.12-100.fc24.x86_64; Host OS, Fedora* Core 24; OvS, 2.7.90; QEMU, qemu-2.6.2-6.fc24; Guest Kernel, 4.4.13-200.fc22.x86_64; Guest OS, Fedora Core 22; DPDK, 16.11; IPSec, Ipsec-secgw from DPDK; nDPI, l2fwd-ndpi; Memtester, 4.3.0; Virsh, 1.3.3.2; pqos 1.1 and pid_cache_flush, https://github.com/01org/intel-cmt-cat/wiki.

3.3 Sample Policy Agent Dynamic Resource Partitioning Results

Figure 10 maps the test results to the system states.

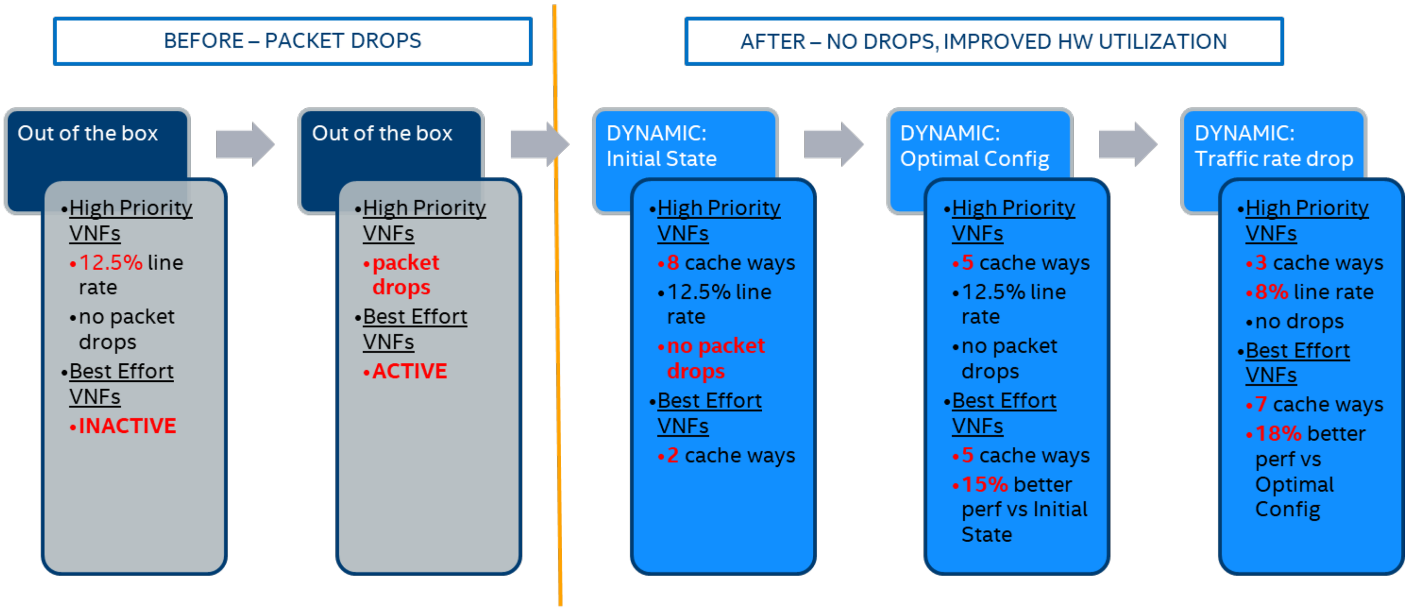


Figure 10. Sample Policy Agent Tests

Reading tested system states on Figure 10 from left to right.

Out of the Box Configuration

In the initial configuration, no resource allocation is in place and High-Priority VNFs achieve 12.5% of the line rate with no packet drops.

In the next step, 3rd party, Best Effort VNF is added, which consumes some of the resources previously used by SD-WAN VNFs and packet drops are observed.

Dynamic Resource Partitioning

Initial State

Next, the sample platform policy agent is engaged with the default configuration. After the new configuration is applied, packet drops are eliminated, High Priority VNFs are allocated 8 out of 12 LLC cache ways, and best effort group gets only 2 cache ways.

Optimal Configuration

Over time, the policy agent optimizes the configuration so that High Priority VNFs are allocated only 5 LLC cache ways and 5 ways are allocated to Best Effort group. No packet drops are observed in the production group and these VNFs achieve 12.5% of the line rate despite extra workload running next to it. Thanks to the cache ways increasing from 2 to 5, the best effort group is operating at 15% better performance level.

Traffic Rate Drop

Finally, the packet rate at traffic generator is reduced from 12.5% to 8% of the line rate. The policy agent reacted to the change and reduced the number of required cache ways for the High Priority VNFs from 5 to 3. There are no packet drops observed and the Best Effort group gets 2 extra cache ways, which further improves its performance by 18%.

4 Conclusion

In a virtualized environment, shared resource contention can have a severe impact on performance. We see this in Section 3.3 where the introduction of a busy Best Effort (Noisy Neighbor) decreases performance (for example, as a result of packet drops). By using CAT, we can statically partition the cache resources, allowing the high-priority VNFs' group to utilize as much resources as it needs to operate optimally and efficiently.

An earlier paper, *Performance evaluation of cache allocation technology for NFV noisy neighbor mitigation*, describes Static Resource Partitioning, but this solution cannot react to dynamically changing cache resource demands.

Using a Platform Policy Agent that implements the concept of Dynamic Resource Allocation, it is possible to react to changing VNFs' groups cache demand to not only to protect high-priority group performance, but also allow the best effort group to gain performance, as described in Section 3.3.

At the beginning of this white paper, we set a goal:

> Implement a Platform Policy Agent to allow Dynamic Resource partitioning between high-priority production VNFs and best effort (with noisy neighbor behavior) VNFs without affecting the performance of production VNFs.

From various tests and results gathered here, you can see that the goal has been achieved. We were able to implement a Platform Policy Agent that is monitoring two groups of VNFs, high-priority and best effort, reallocating cache resources as needed, and protecting high-priority group performance while allowing the best effort group to gain performance when possible.

Appendix A Platform and Software Details

This appendix provides test environment configuration information including hardware components, BIOS settings, kernel parameter values, and software packages.

A.1 Hardware Packages

Table 7 lists all hardware used.

Table 7. Hardware Packages

Hardware Component	Version	Links	Comments
SuperMicro	Supermicro motherboard-X10 Series		CPU Model Name: Intel® Xeon® CPU D-1540 @ 2.00GHz
Ethernet Transceivers	Intel 10 Gb Small Form-factor Pluggable (SFP) +	http://www.intel.com/content/www/us/en/ethernet-products/optics-cables/ethernet-sfp-optics-brief.html	
Memory	32 GB DDR4 2133 MHz (x2)		BankLocator: P0_Node0_Channel0_Dimm0 & P0_Node0_Channel01_Dimm0
Traffic Generator	IXIA XG12 IxNetworks V 7.50	https://www.ixiacom.com/products/ixnetwork	

A.2 BIOS Settings

Table 8 shows all BIOS settings used.

Table 8. BIOS Settings

Setting	State
Hyper Threading	ENABLED
Intel® Virtualization Technology for Directed I/O (Intel® VT-d)	ENABLED
Intel® Virtualization Technology	ENABLED
P-States	DISABLED
C-States	DISABLED
Package C-State Limit	C0/C1 State
CPU C3 Report	DISABLED
CPU C6 Report	DISABLED
Autonomous C states	DISABLED

A.3 List of Kernel Parameters

Table 9 shows Kernel Parameter values used.

Table 9. Kernel Parameters

Host OS	Guest OS
audit=0	audit=0
rhgb quiet	rd.lvm.lv=fedora/root
rcu_nocbs=1-7,9-15	rd.lvm.lv=fedora/swap
isolcpus=1-7,9-15	isolcpus=1
default_hugepagesz=1G	default_hugepagesz=2M
hugepages=8	hugepages=512
hugepagesz=1G	hugepagesz=2M
idle=poll	idle=poll
intel_pstate=disable	intel_pstate=disable
mce=ignore_ce	console=ttyS0,115200n8
nohz_full=1-7,9-15	nohz_full=1
intel_iommu=off	nohz=on
nosoftlockup	nosoftlockup
processor.max_cstate=1	rcu_nocb_poll
intel_idle.max_cstate=0	rcu_nocbs=1
selinux=0	

A.4 Software Packages

Table 10 lists all software packages that were used.

Table 10. Minimal Versions Requirements

Software Packages	Version	Links	Comments
Host Kernel	4.11.12-100.fc24.x86_64		
Host OS	Fedora* Core 24		
OvS	2.7.90	http://openvswitch.org/	
QEMU	qemu-2.6.2-6.fc24		
Guest Kernel	4.4.13-200.fc22.x86_64		
Guest OS	Fedora Core 22		
DPDK	16.11	http://dpdk.org/	
IPSec	Ipsec-secgw from DPDK	https://doc.dpdk.org/guides-16.04/sample_app_ug/ipsec_secgw.html	Patched on top of DPDK 16.11 (add support for 1 thread per port and support for 2000 flows)
nDPI	l2fwd-ndpi		Patched on top of DPDK 16.11
Memtester	4.3.0		
Virsh	1.3.3.2		
pqos	1.1	https://github.com/01org/intel-cmt-cat/wiki	PQoS tool from the Intel® RDT Software Package
pid_cache_flush	-		Internally developed Linux kernel module that can remove data from LLC belonging to selected PID. This is helpful to make LLC allocation changes take prompt effect.

A.5 Software Configuration

A.5.1 Open vSwitch

We used Open vSwitch (OvS) with the Data Plane Development Kit (DPDK) in both SUT configurations because of the greater throughput that we could achieve over the standard OvS approach.

The data plane configuration type is a userspace bridge. We used two `vhost-user` ports for nDPI and two `vhost-user` ports for IPSec. Two physical ports were added from the Ethernet controller (X710-DA4) as DPDK-type ports. We developed flows to handle the various packet flow scenarios.

A.5.2 IPSec Security Gateway

The DPDK IPSec Security Gateway is an application that uses the `cryptodev` framework. The application demonstrates the implementation of a security gateway using the DPDK based on RFC4301, RFC4303, RFC3602 and RFC2404. Internet Key Exchange (IKE) is not implemented, therefore only the manual setting of Security Policies (SPs) and Security Associations (SAs) is supported. The SPs are implemented as Access Control List (ACL) rules, and the SAs are stored in a table with routing implemented using Longest Prefix Match (LPM).

The IPSec Security Gateway application classifies the ports as Protected and Unprotected. Therefore, traffic received on an Unprotected or Protected port is consider Inbound or Outbound respectively.

The process for IPSec Inbound traffic is:
- Read packets from the port.
- Classify packets between IPv4 and Encapsulating Security Payload (ESP).
- Perform an Inbound SA lookup for ESP packets based on their SPI.
- Perform Verification/Decryption.
- Remove the ESP and outer IP header.
- Perform an Inbound SP check using the ACL of decrypted packets and any other IPv4 packets.
- Do routing.
- Write the packet to the port.

The process for the IPSec Outbound traffic is:
- Read packets from the port.
- Perform an Outbound SP check using the ACL of all IPv4 traffic.
- Perform an Outbound SA lookup for packets that need IPSec protection.
- Add an ESP and outer IP header.
- Perform Encryption/Digest.
- Do routing.
- Write the packet to port.

A.5.2.1 Source Code Modifications

To fulfill the requirements of the setup, we modified the source code of the `ipsec-secgw` application as follows:
- Added support for 2000 flows (the original maximum was 1000).
- Added support for using one thread per port (normally, one core handles both incoming and outgoing traffic, but this can lead to poor performance).

These changes are available as patches that must be applied to the DPDK.

A.5.2.2 Command Line Parameters

The `ipsec-secgw` application was configured with 2000 flows and 200 IPSec tunnels. The IPSec control-plane is not used and the tunnels are statically configured using generated configuration files.

As an encryption engine, the Intel® Multi-Buffer Crypto for IPSec library and the AESN-NI Multi Buffer Crytpo Poll Mode Driver (`crypto_aesni_mb`) are used.

The following snippet shows the `ipsec-secgw` command options used in the setup:

```
ipsec-secgw -c 0x6 -l 1,2 -n 4 \
--vdev="crypto_aesni_mb" \
-w 00:07.0 -w 00:08.0 -- \
-p 0x3 -P -u 0x2 \
--config="(0,0,1),(1,0,2)" \
-f ~/configs/con2003QATEP1.cfg
```

A.5.3 Deep Packet Inspection

For forwarding packets with Deep Packet Inspection (DPI), we used a tool called `l2fwd-ndpi`. This tool is not included in the DPDK by default. Special patches must be applied to the DPDK. The `l2fwd-ndpi` tool uses the `libndpi` library for deep packet inspection. DPI is performed on 100 packets for every 2000 packets to simulate changing flows.

DPI is done on unclassified uplink traffic only.

A.5.3.1 l2fwd-ndpi Command Line Parameters

Use the following command to start the `l2fwd-ndpi` tool:

```
l2fwd-ndpi -n 4 -c 0x6 \
-w 0000:00:06.0 -w 0000:00:07.0 -- \
-p 0x3 \
-x 100 -y 2000 \
-i -n 0x1
```

A.5.4 Best Effort VM

To simulate cache and memory bandwidth-bound workload, we used the `stress-ng` tool's standard matrix multiplication stressor. In our case, it multiplies a floating-point matrix by a scalar. We chose a matrix size of 768 by 768 as it shows the best response (the largest difference in number of retired instructions per second) to changes in the number of allocated Cache Ways (CWs).

A.5.4.1 stress-ng Command Line Parameters

Start two `stress-ng` instances, pinned to cores 0 and 1, using the following commands:

```
taskset -c 0 stress-ng --matrix 1 --matrix-method mult \
--matrix-size 768 -t 2y
taskset -c 1 stress-ng --matrix 1 --matrix-method mult \
--matrix-size 768 -t 2y
```

A.5.4.2 CPU Idle Linux Kernel Parameter

To have a number of retired instructions that is greater than 0 observed by the Reliability Agent (using `perf`) for the idle Best Effort VM, it is necessary to force the use of a polling idle loop for CPU idle (`idle=poll`).

The Best Effort VNF's kernel parameters are as follows:

```
BOOT_IMAGE=/vmlinuz-4.4.13-200.fc22.x86_64 root=/dev/mapper/fedora-root ro rd.lvm.lv=fedora/swap
rd.lvm.lv=fedora/root console=ttyS0,115200n8 hugepagesz=2M hugepages=512 default_hugepagesz=2M
audit=0 idle=poll
```

A.5.5 Cache Flush per PID Kernel Module

To flush the cache per PID, the `pid_cache_flush` Linux* kernel module was developed internally. The kernel module invalidates cache lines belonging to specific PIDs from all levels of the processor cache hierarchy. This is helpful to make LLC allocation changes take prompt effect.

A.6 System Configuration

A.6.1 Isolated Cores

To ensure that VMs, the OvS, and the host Operating System (OS) have their own cores for their own processing requirements, and to keep them outside the control of the general scheduler, they are allocated to isolated cores.

In this setup, the isolated cores are 1-7, 9-15 (HT). This is done using `grub` command line arguments and is set at each boot up.

A.6.2 Core Pinning

Table 11 shows the cores to which each VM and component are pinned.

Table 11. Core Pinning

VM/Component	Core Pinning
Host OS	0, 8
Hypervisor (KVM/QEMU)	0, 8
OvS	0, 8
DPDK PMD	7, 15
DPI VM	2, 3, 11
IPSec VM	4, 5, 13
Best Effort VM	1, 6

The virtual shell utility `virsh` is used to set CPU pinning /affinity using an Extensible Markup Language (XML) file.

The Host OS, Hypervisor, and OvS (excluding the Poll Mode Driver [PMD]) are pinned to two Intel® Hyper-Threading Technology (Intel® HT Technology) threads belonging to one physical core. Best Effort VM virtual CPUs (vCPUs) are pinned to two Intel® HT Technology threads belonging to two separate physical cores (for increased memory bandwidth utilization). The Best Effort VM is running on one machine only.

A.7 Test Environment Setup

Figure 11 shows a high-level view of the test environment setup.

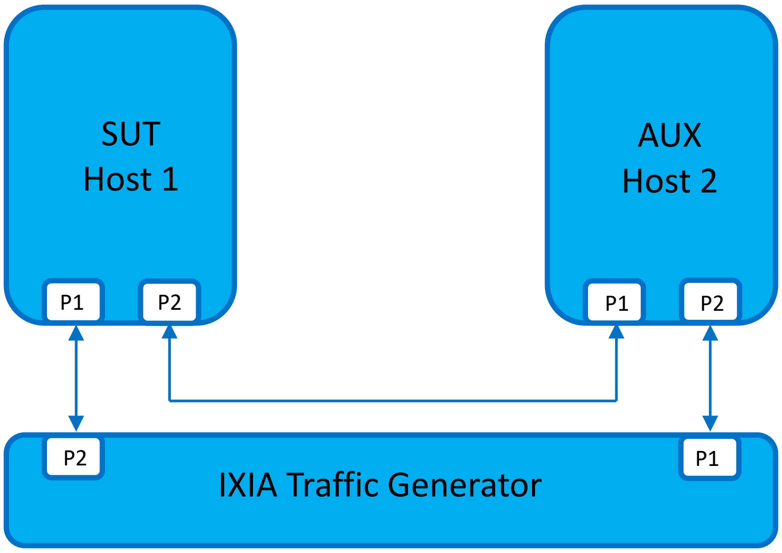


Figure 11. Test Environment Setup

A.8 IXIA Traffic Generator

Configuration settings:

- Packet Size: 256B
- Traffic Type: IPv4, Transmission Control Protocol (TCP), Hypertext Transfer Protocol (HTTP) “GET” requests, Bi-directional traffic
- Line Rate: 12.5%, 8%
- Flows: 2000 (1000 in each direction)

A.9 SUT Components

System Under Test (SUT) components include:

- Group of High-priority VMs (nDPI, IPSec)
- Best Effort VM (single VM running `stress-ng`)
- OvS
- Cache Flush per PID kernel module
- Sample Platform Policy Agent