

Benchmarking and tuning NFV, using Yardstick/NSB, OVS, prox, perf, Intel PCM

Alexander Komarov

Application Engineer, Intel Software and Services Group

3.11.2017

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2017, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

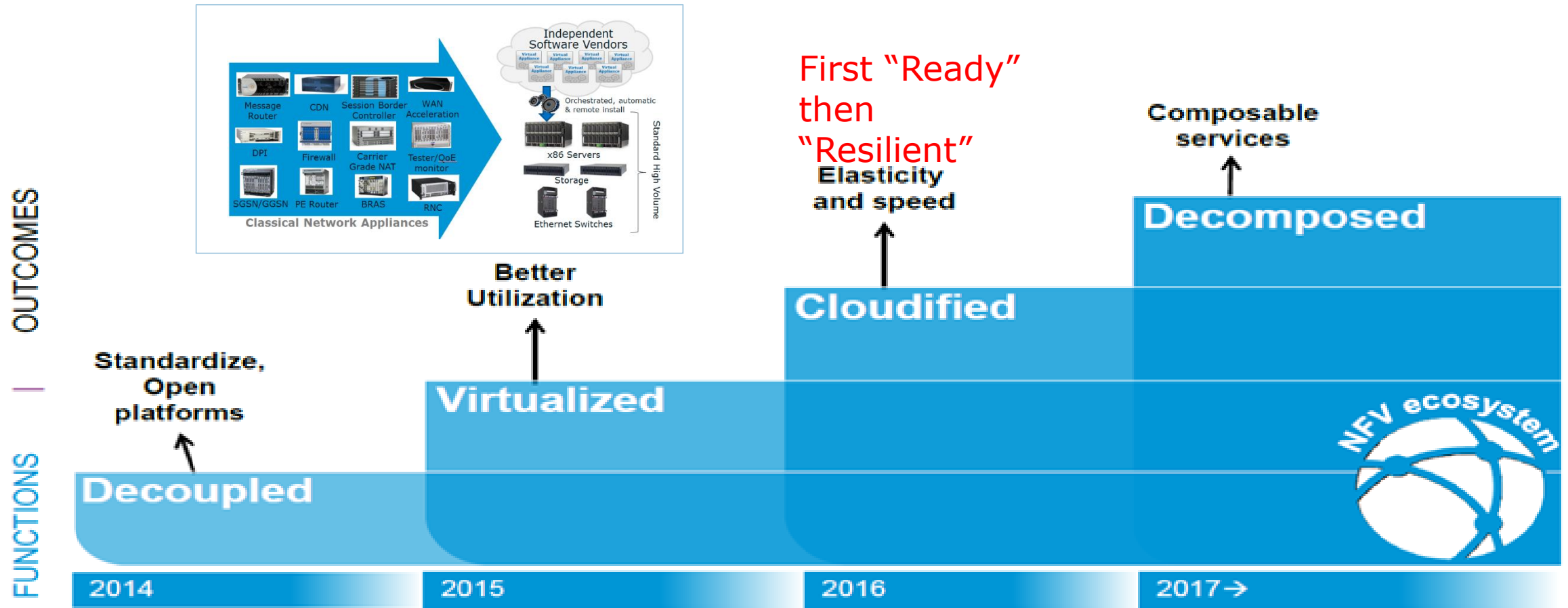
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

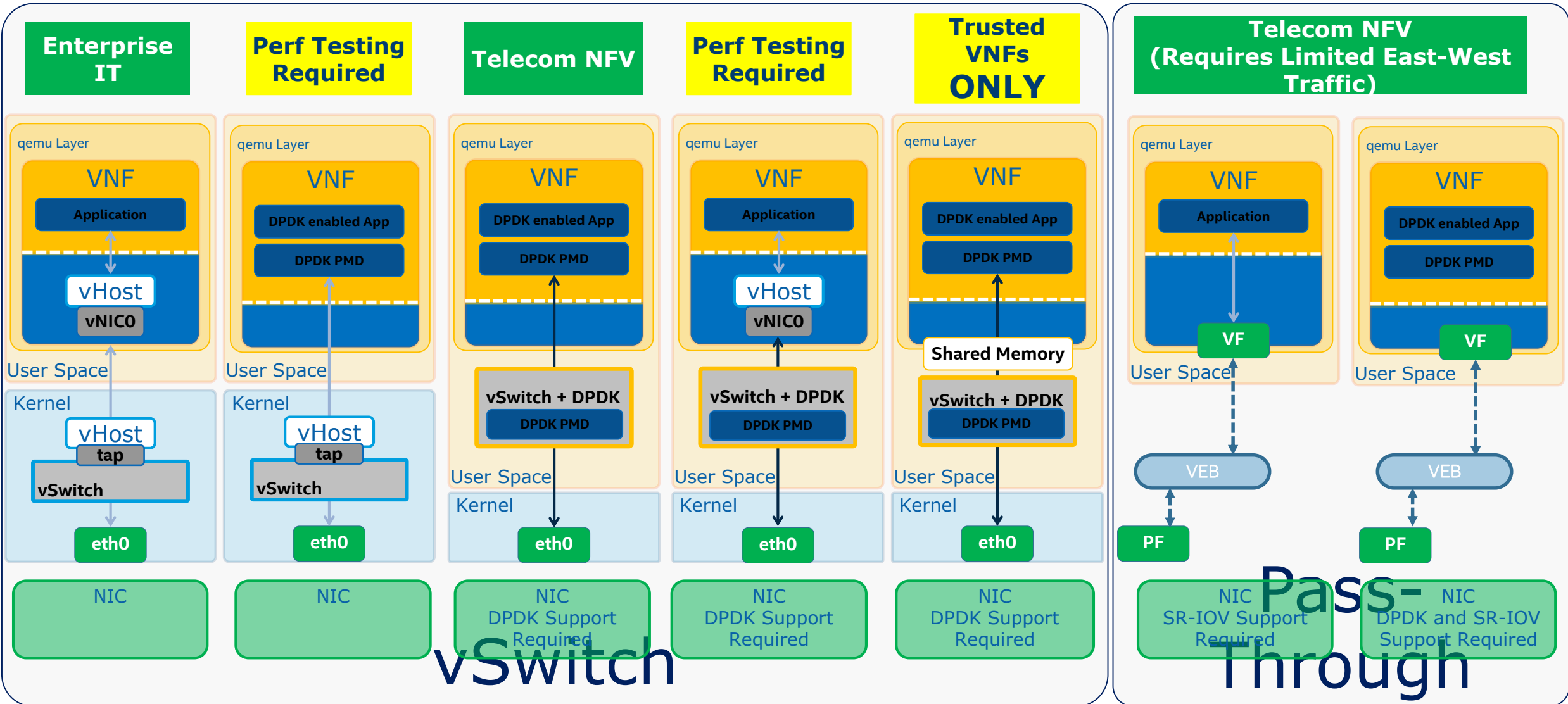
Agenda

- NFV intro
- NSB (Prox, Yardstick)
 - Benchmarking
 - Examples (mini case studies)
- Virtualized performance profiling
 - Profiling guest
 - Profiling in host
- Utilizing best instruction sets in private clouds (EPA or bottom up approach)
- VNFs in containers vs VMs.

NFV software transformation



NFV config options for Networking Data Path



NSB Methodology – vnf performance benchmarking

VNF performance benchmarking

Native Linux environment

Standalone Virtualized environment

Managed virtualized environment (e.g. OpenStack)

Evaluate both scale-up and scale-out performance data

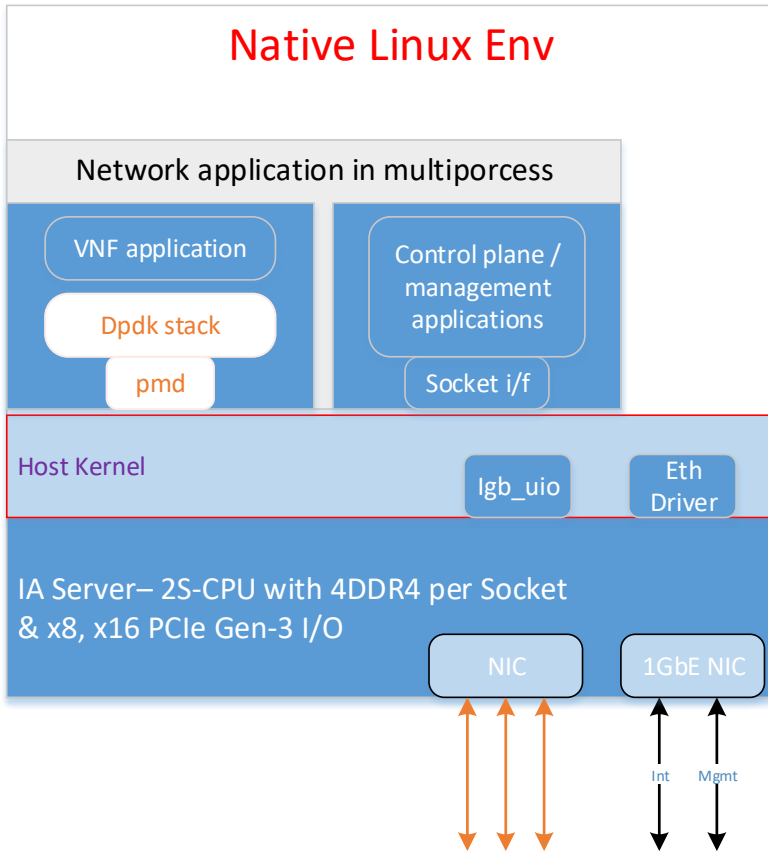
VNFs performance graphs for both scale-up and scale-out in all three environments

Collect KPIs: Network KPIs, VNF KPIs and NFVi KPIs

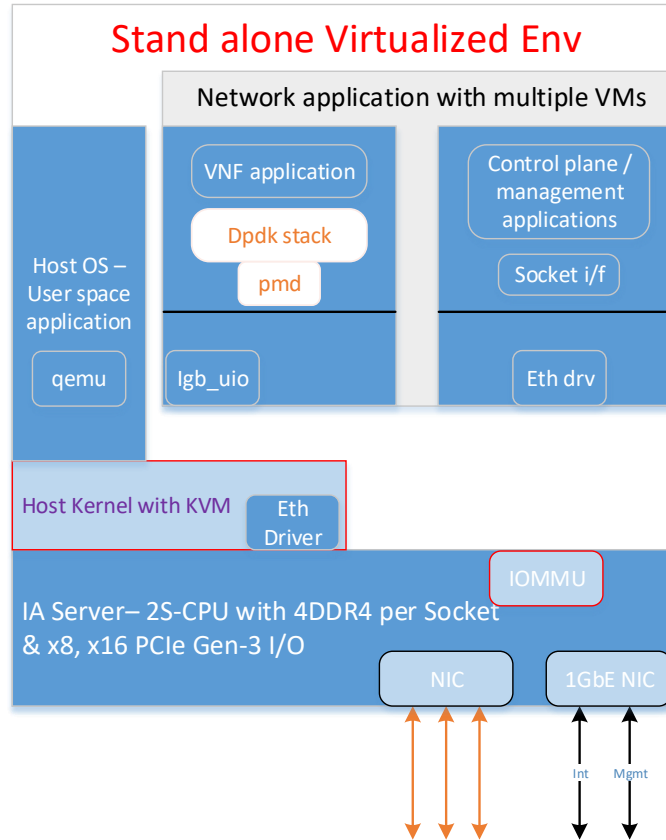
Test Infrastructure: Standard test framework for all 3 environments

Methodology: Vnf comparison through performance benchmarking

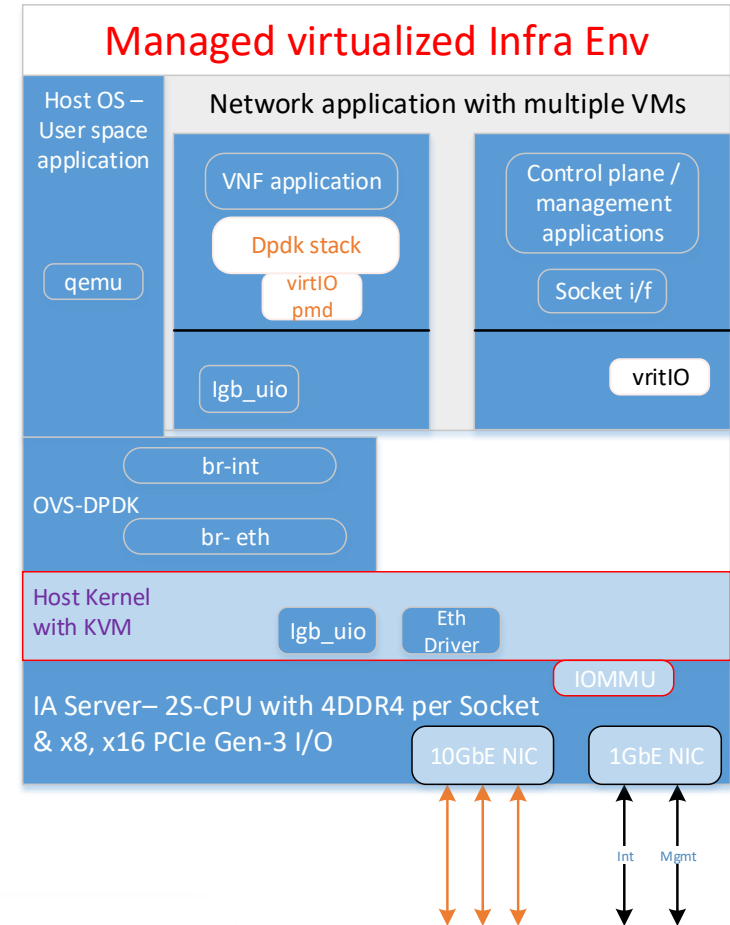
Native Linux Env



Stand alone Virtualized Env



Managed virtualized Infra Env



EVALUATE BOTH SCALE-UP AND SCALE-OUT PERFORMANCE DATA

Physical Network Topology and SW BOM

FOR SW BASED TRAFFIC GENERATORS

NETWORK TOPOLOGY



- Supported traffic generators**
- TREN
 - Pktgen
 - Ping
 - Prox gen
 - Benchmark/NGiNX

SW BOM

NSB 16.04 based on YardStick	
TREX 2.05	2.4.18-2ubuntu3.1
NGiNX 1.1.1.10	Pktgen 3.0.14
OVS 2.7	
DPDK 17.04	
QEMU 2.5	
Ubuntu 16.04 LTS	



NFVI Characterization Tools - PROX

Workloads: developed in collaboration with SP partners

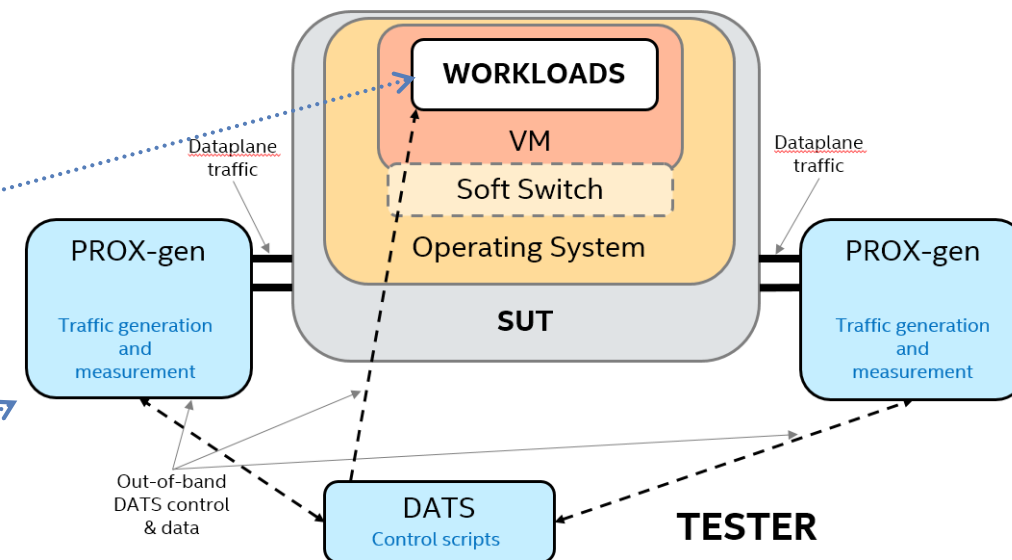
- Workloads exercise NFVI HW and SW features identified as performance-critical for VNFs and SFCs
- Workloads implement “kernels” of VNF/SFC functionality
- Run in PROX using different configuration files

Traffic generation and measurement (PROX-gen)

- Generates traffic that is specific to each workload
- Packet size and arrival distribution is configurable
- Automation interface provided for DATS

DATS: Dataplane Automated Testing System

- Scripts to automatically drive, measure, report multiple dataplane test cases.
- Measurements with 0 packet loss (according to RFC2544)
- Workload-specific Key Performance Indicator (KPI) used as summary statistic
- Output is an automatically generated report.
- Enables faster, more reliable test case execution



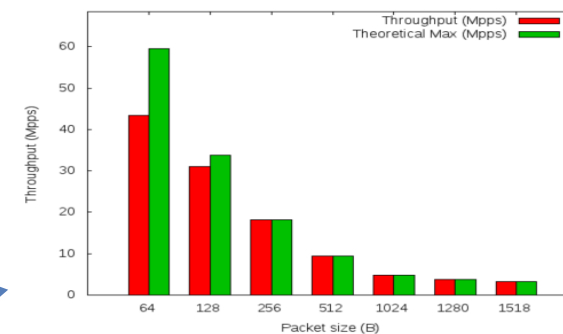
Port forwarding without touching packets

Description

The application will take packets in from one port and forward them unmodified to another port. This use case is not representing any real use case but it is a good start to do a sanity check of the environment.

The KPI is the number of packets per second for 64 byte packets with an accepted minimal packet loss (zero, no packet loss).

Result



Packet size (B)	Throughput (Mpps)	Theoretical Max (Mpps)	Duration (s)
64	43.3945123314	59.5238095238	120.278173923
128	30.9998391896	33.7837837838	120.292181015
256	18.1156148588	18.115942029	15.035173893
512	9.39842704915	9.3984962406	15.0372900963
1024	4.78927200203	4.78927203065	15.0380678177
1280	3.84615146073	3.84615384615	15.0364170074
1518	3.25097532563	3.25097529259	15.0354731083

Prox screenshot

- Open source project at 01.org, github (under OPNFV)
- On top of DPDK and prox, developer can combine building blocks in a text config file to create DPDK performance demonstrators/custom traffic gen/test tools
- Developer can implement additional building blocks (complimenting available blocks: Gen,lat,nop,acl,ipsec,qos,qinq,classify,cgnat,gre,route,police,lb*,etc)
- Convenient NCURSES GUI for live stats monitoring / configuration

```
prox v0.19: esp 12 up
1 tasks 2 ports 3 mem 4 lat 5 ring
Host pps rx: 0 tx: 0 diff: 0 avg rx: tx: %: -nan
NICs pps rx: 0 tx: 0 err: 0 avg rx: tx: %:
Core/Task Port ID/Ring Name Statistics per second Total Statistics
Nb Name Mode RX TX Idle (%) RX (k) TX (k) Drop (k) CPP clk (GHz) RX TX
2/0 esp_enc esp_enc 1 A 60.89 0 0 0 0 0.000 0 0
3/0 esp_dec esp_dec A 0 66.15 0 0 0 0 0.000 0 0

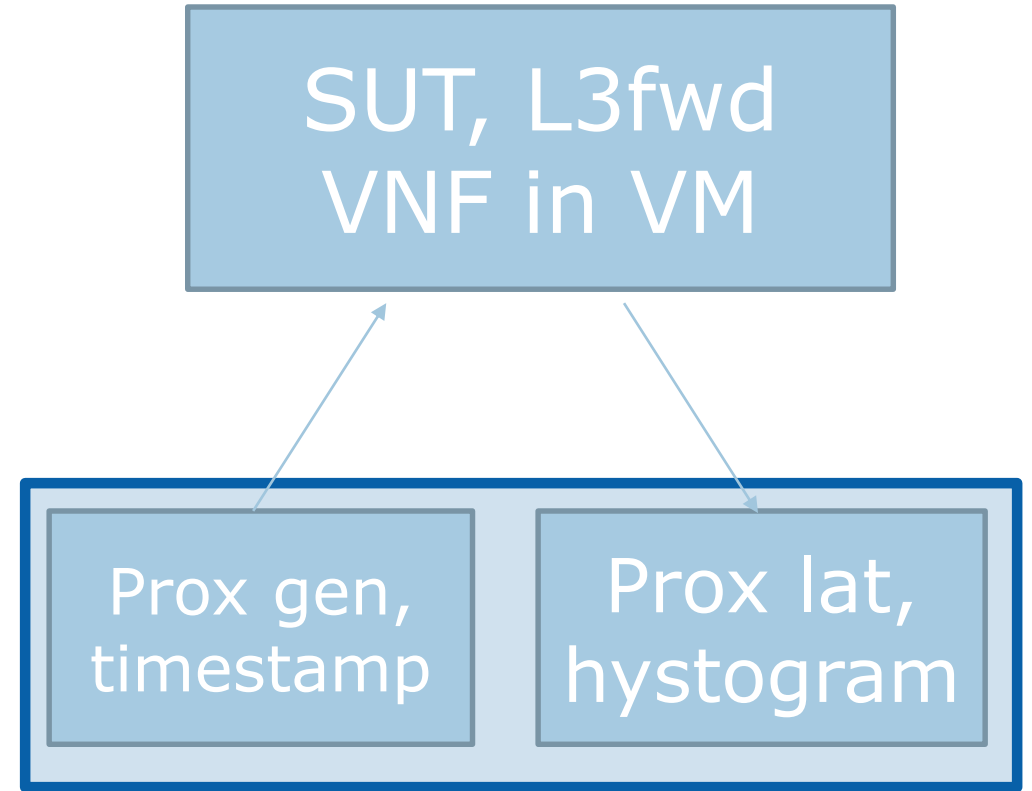
Core 2: RX port 1 (queue 0) ==> TX ring 0x7f4ab29c6080
Core 3: RX ring[0,0] 0x7f4ab29c6080 ==> TX port 0 (queue 0)
Started with 2 warnings, last 2 warnings:
warn : System did not report numa_node for device 0000:00:03.0
warn : System did not report numa_node for device 0000:00:08.0
Starting cores: 2, 3
Starting core 2 (all tasks)
Starting core 3 (all tasks)
Entering main loop on core 2
Entering main loop on core 3
Waiting for core 2 to start... OK
Waiting for core 3 to start... OK
```

Prox case study 1

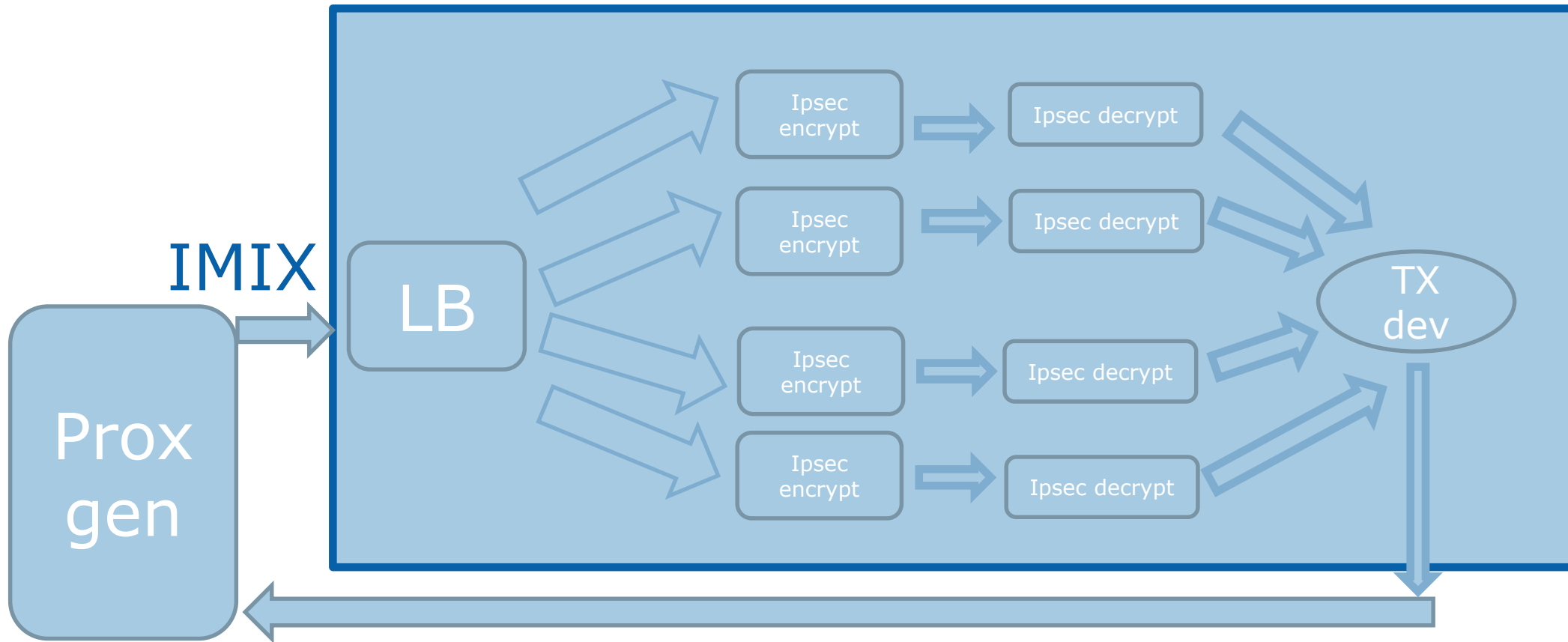
- VNF Latency PCI-Passthrough
- VNF Jitter PCI-Passthrough
- VNF Latency SR-IOV
- VNF Jitter SR-IOV

Results, μ s:

	10M small packets (64 bytes) l3 fwd VNF			
	simple fwd, native	l3fwd, native	l3fwd, PCI passthrough	l3fwd, SR-IOV
min	14	30	45	40
avg	30	64	74	69
max	90	145	160	160



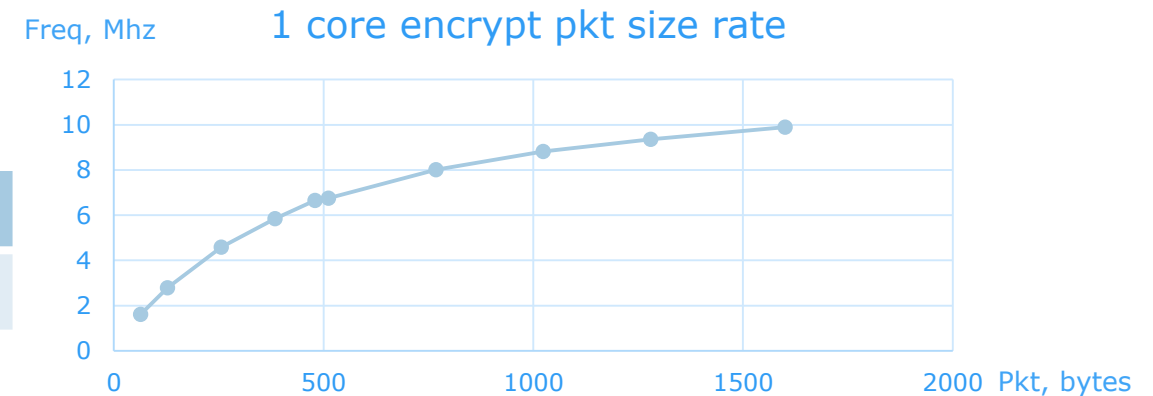
Case study 2 – Prox setup



Case study 2 – SKL-SP performance

- Xeon Platinum 8168 @2.7Ghz, RHEL 7.2, Niantic NICs.
- ~2X performance improvement over BDW-EP on IPsec workload: ~20% from IPC increase, ~80% from SIMD optimizations in DPDK-17.05 and Intel multibuffer crypto library.
- Mhz/Kpkt, @IMIX pkt rate:

BDW-EP	SKL-SP
3.4	1.5



- At IMIX full line rate (3940Kpkt), need 2 encrypter cores (+ LB core), 2 decrypter cores. Scales very well with MHZ and cores.
- When using integrated QAT, need 1 core for encrypt and 1 core for decrypt

CPU/system profiling tools

- Linux perf – de facto standard now. PMU sampling in NMIs
- Intel PCM – PMU counting mode, uncore PMU
- Intel Vtune – standalone or as perf results viewer. Best viewer for x86 PMU sampling. Custom collectors with IPT support, stack sampling support; or frontend for perf

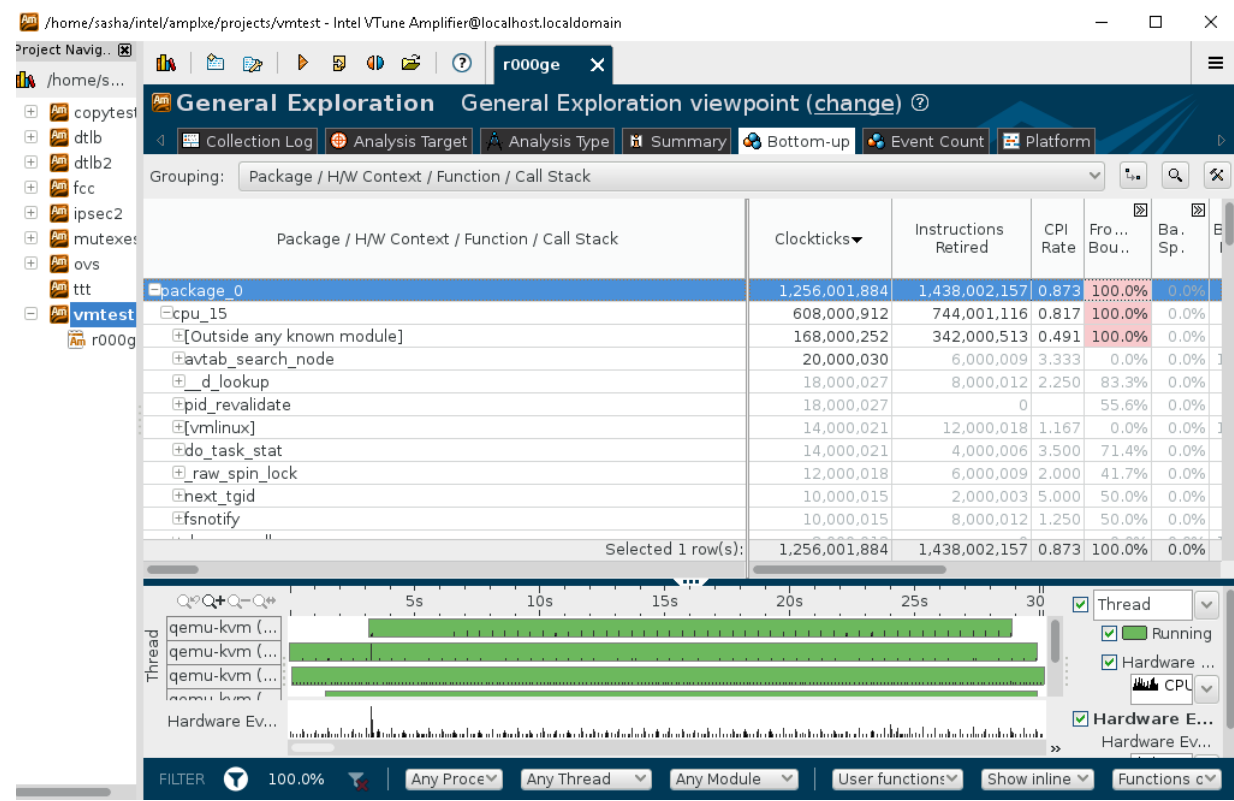
```
[root@localhost pcm-master]# ./pcm-uma.x
Processor Counter Monitor: NUMA monitoring utility
Number of physical cores: 48
Number of logical cores: 48
Number of online logical cores: 48
Threads (logical cores) per physical core: 1
Num sockets: 2
Physical cores per socket: 24
Core PMU (perfmn) version: 4
Number of core PMU generic (programmable) counters: 8
Width of generic (programmable) counters: 48 bits
Number of core PMU fixed counters: 3
Width of fixed counters: 48 bits
Nominal core frequency: 2700000000 Hz
Package thermal spec power: 285 Watt; Package minimum power: 90 Watt; Package maximum power: 413 Watt;
Socket 0: 2 memory controllers detected with total number of 6 channels. 3 QPI ports detected.
Socket 1: 2 memory controllers detected with total number of 6 channels. 3 QPI ports detected.
Socket 0
Max QPI link 0 speed: 23.4 GBytes/second (10.4 GT/second)
Max QPI link 1 speed: 23.4 GBytes/second (10.4 GT/second)
Max QPI link 2 speed: 21.5 GBytes/second (9.6 GT/second)
Socket 1
Max QPI link 0 speed: 23.4 GBytes/second (10.4 GT/second)
Max QPI link 1 speed: 23.4 GBytes/second (10.4 GT/second)
Max QPI link 2 speed: 21.6 GBytes/second (9.6 GT/second)
Detected Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz "Intel(r) microarchitecture codename Skylake-SP"
Update every 1.0 seconds
Time elapsed: 1026 ms
Core | IPC | Instructions | Cycles | Local DRAM accesses | Remote DRAM Accesses
0 | 0.30 | 24 M | 82 M | 1511 | 11 K
1 | 0.85 | 83 M | 129 M | 385 | 4176
2 | 0.22 | 846 K | 3843 K | 197 | 641
3 | 0.13 | 321 K | 2417 K | 409 | 630
4 | 0.17 | 525 K | 3039 K | 273 | 334
5 | 0.13 | 284 K | 2217 K | 149 | 272
6 | 0.57 | 74 M | 130 M | 233 | 993
7 | 0.15 | 363 K | 2880 K | 267 | 509
8 | 0.16 | 436 K | 2010 K | 349 | 834
9 | 0.16 | 389 K | 1949 K | 136 | 553
10 | 0.14 | 379 K | 2764 K | 255 | 922
47 | 0.14 | 287 K | 2006 K | 246 | 310
+ | 0.43 | 199 M | 460 M | 13 K | 57 K
```

The screenshot shows the Intel VTune Amplifier XE 2017 interface. The main window displays a 'General Exploration' viewpoint with a table of performance metrics. The table has columns for 'Package / HW Context / Function / Call Stack', 'Clockticks', 'Instructions Retired', 'CPI Rate', and 'Front-End Latency' (sub-columns: ICache Misses, ITLB Overhead, Branch Resteers, DSB Switches, Len). The table is sorted by 'Instructions Retired' in descending order. The top row is 'package_0' with 148,318,222,477 clockticks and 80,522,120,783 instructions retired. Below it are 'cpu_3', 'cpu_0', and various functions like 'main', 'ExAllocatePoolWithTag', and 'func@0x180510420'. At the bottom of the interface, there is a 'Thread' visualization showing CPU time and system bandwidth for several threads over time.

Package / HW Context / Function / Call Stack	Clockticks	Instructions Retired	CPI Rate	Front-End Latency				
				ICache Misses	ITLB Overhead	Branch Resteers	DSB Switches	Len
package_0	148,318,222,477	80,522,120,783	1.842	11.8%	2.1%	3.4%	1.2%	
▶ cpu_3	37,828,066,742	21,374,032,061	1.770	11.2%	1.8%	0.9%	1.2%	
▼ cpu_0	37,732,056,598	20,334,030,501	1.856	13.3%	2.3%	4.6%	1.1%	
▶ [Outside any known module]	4,820,007,230	4,082,006,123	1.181	8.3%	2.4%	5.6%	3.3%	
▶ main	3,260,004,890	1,150,001,725	2.835	0.0%	0.0%	0.0%	0.0%	
▶ ExAllocatePoolWithTag	868,001,302	1,372,002,058	0.633	4.6%	0.2%	0.0%	0.0%	
▶ ExFreePoolWithTag	676,001,014	1,302,001,953	0.519	5.9%	0.3%	0.0%	0.0%	
▶ func@0x180510420	622,000,933	200,000,300	3.110	0.0%	0.0%	0.0%	0.0%	
▶ func@0x180730e50	360,000,540	198,000,297	1.818	0.0%	0.6%	0.0%	0.0%	

Profiling in host

- # perf kvm --host --guest ... -a record -o perf.data
 - Get /proc/kallsyms, /proc/modules from guest first.
 - Read results with perf or Vtune, or collect with Vtune (will invoke perf kvm collector).
- # Intel pcm-core, pcm-pcie, pcm-memory, pcm-numa, pcm-tsx, pcm-power, pcm-sensors.
 - Watch for excessive NUMA, memory, PCI traffic
- Any way, you only see detailed host data, kernel guest data [single guest!], and aggregated user mode guest data.
- Why bother - ?
 - Because we can read and sample all cores and uncore PMU counters!



Profiling in guest

#perf list // Run in guest

- L1-dcache-loads [Hardware cache event] GOOD
- Only cpu-clock,etc [Software event] BAD. Make sure qemu -cpu host, OpenStack config capabilities:vcpu_model.features=arch_perfmon

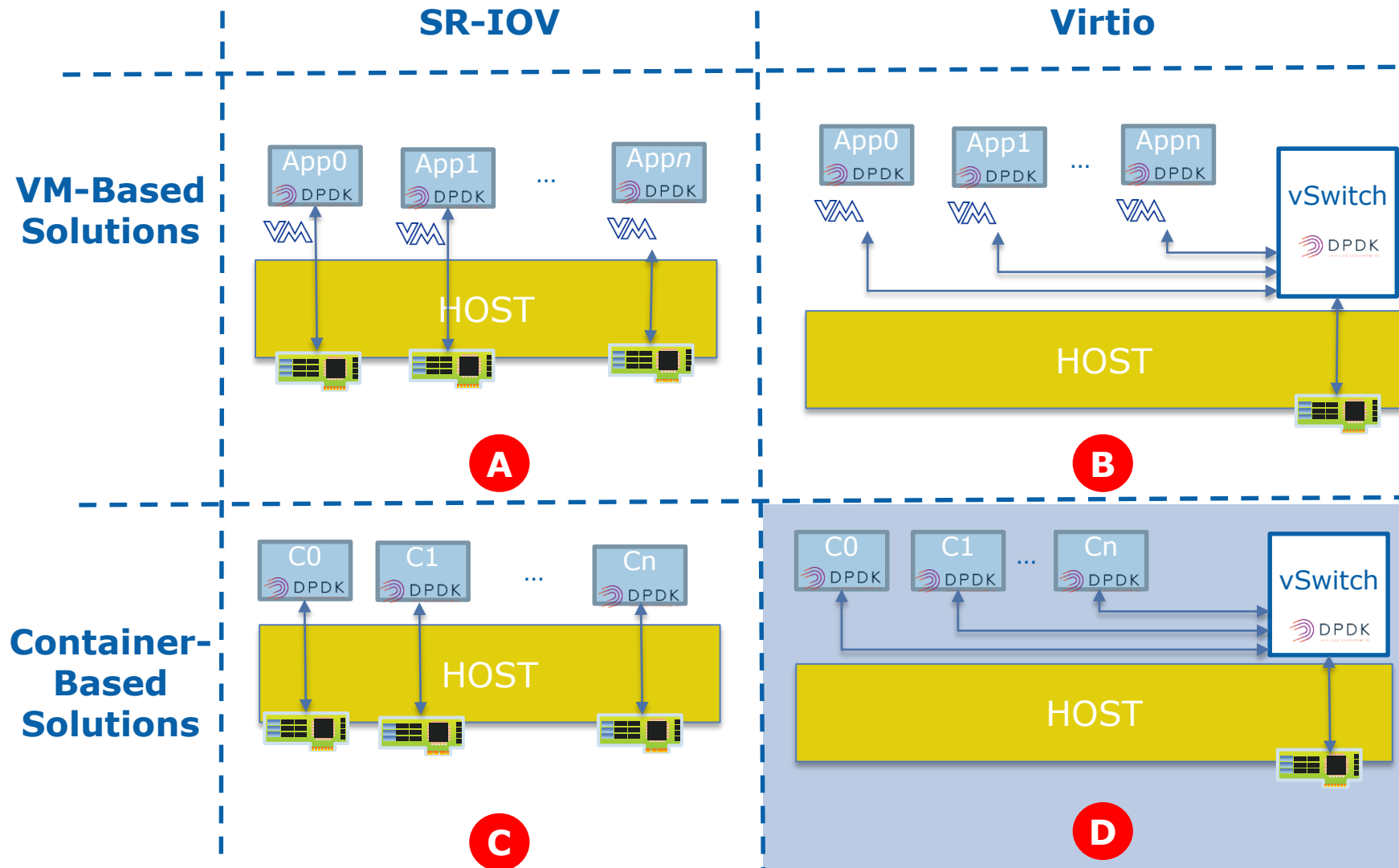
perf record -a -e instructions,ref-cycles,LLC-loads,LLC-load-misses,LLC-stores,LLC-store-misses,... sleep 60

- Open results (perf.data) in perf viewer or import to Vtune (rename to *perf).
- You don't see all PMU events – KVM only passes a dozen most important ones to guests
- All time sampling profiling methods would also work..

Utilizing best ISA in guests

- VNFs (Virtual Network Function) vendors have to deploy to private clouds, often configured to spoof CPUids (Sandy Bridge as default is common). So they have to run with “least common denominator” architecture settings, that is not very efficient.
- Enhanced Platform Awareness approach – top down, configure guests with real CPUIDs, use orchestration to start binaries compiled for the best supported instruction set.
- Bottom up approach – use a small (in LOC) tool that reveals a physical CPU model when running under a hypervisor that hides/spoofs real CPUID to guests.
- Run a tool, get a real physical CPU model, select the right binary (with AVX, AVX2, AES-NI, AVX-512 support)

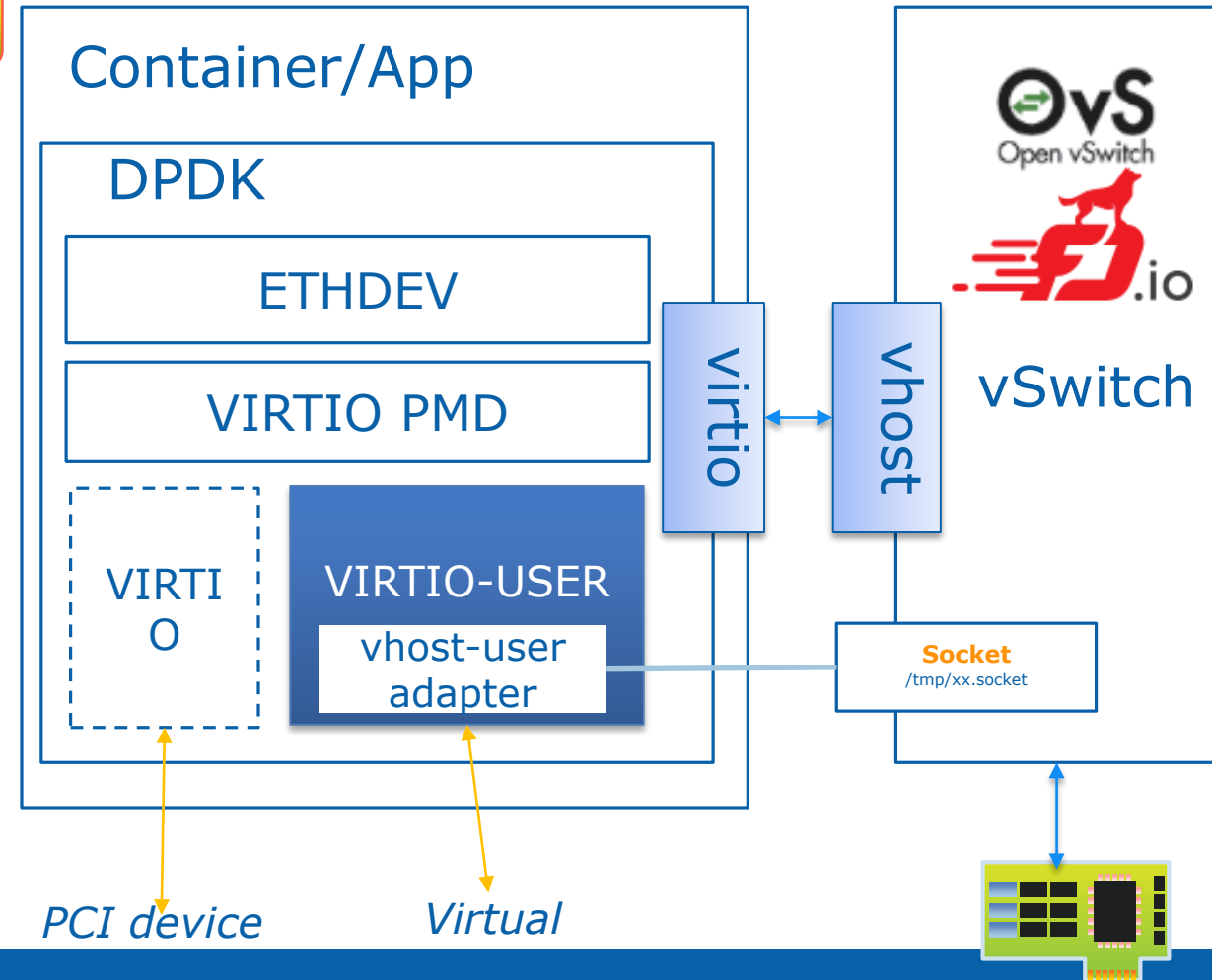
VM & Container Usage Models in NFV



Virtio in Containers

Description

- Virtio in Containers is a new approach to high-speed networking for containers.
- In a VM, QEMU helps with device emulations and interaction with the backend.
- In containers, we don't have QEMU:
 - We could introduce a kernel module to fulfil the same function as QEMU, but we're already trying to remove the existing out-of-tree kernel modules from DPDK.
 - Instead, all of the work is done in the DPDK PMD driver. We present virtio as a virtual device, just like the way that Ring, PCAP, or other virtual devices are used in DPDK. The control message are also handled through the DPDK driver.



Download links, docs

- Intel PCM - <http://www.intel.com/software/pcm>
- NSB/Yardstick - <https://wiki.opnfv.org/display/yardstick/Yardstick>
- Prox - <https://01.org/intel-data-plane-performance-demonstrators>
 - Source, as a part of OPNFV: <https://github.com/opnfv/samplevnf/tree/master/VNFs/DPPD-PROX>
- Cisco Trex - <https://trex-tgn.cisco.com/>
- Linux perf - https://perf.wiki.kernel.org/index.php/Main_Page
- Intel Vtune - <https://software.intel.com/en-us/intel-vtune-amplifier-xe/>
- CPUvirt2phys - <https://software.intel.com/en-us/articles/how-to-tell-cpu-model-when-running-under-hypervisor-that-spoofs-cpuid>

