FLEXRAN PLATFROM SW INTRODUCTION

REAL-TIME PLATFORM KEY REQUIREMENTS AND PARAMETERS

NPG Wireless Access Division

Legal Notices and Disclaimers

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.

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.

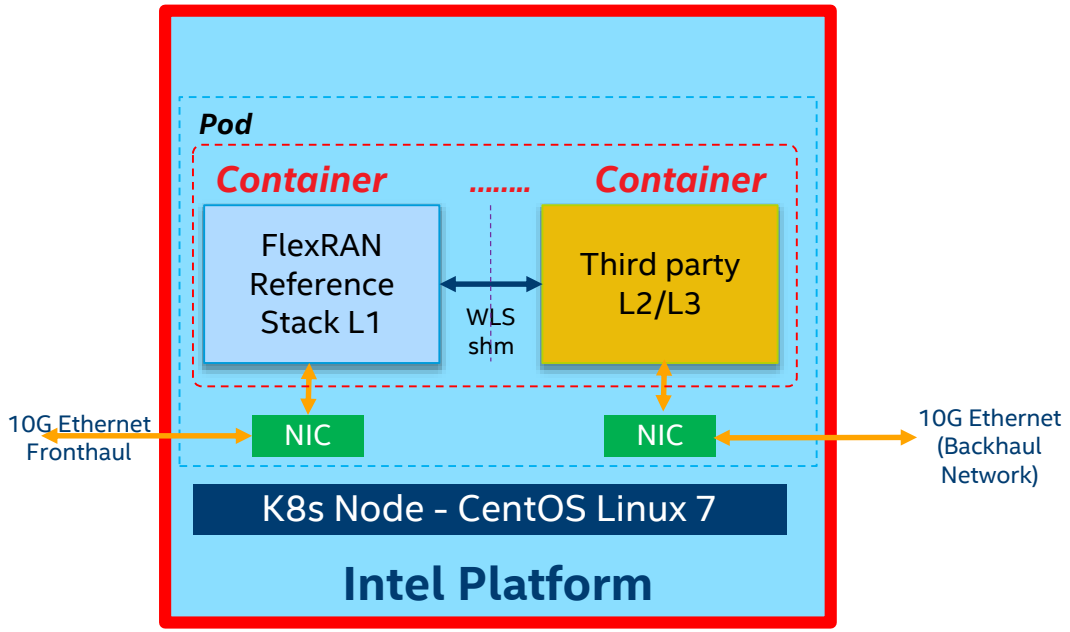
Notice revision #20110804

Agenda

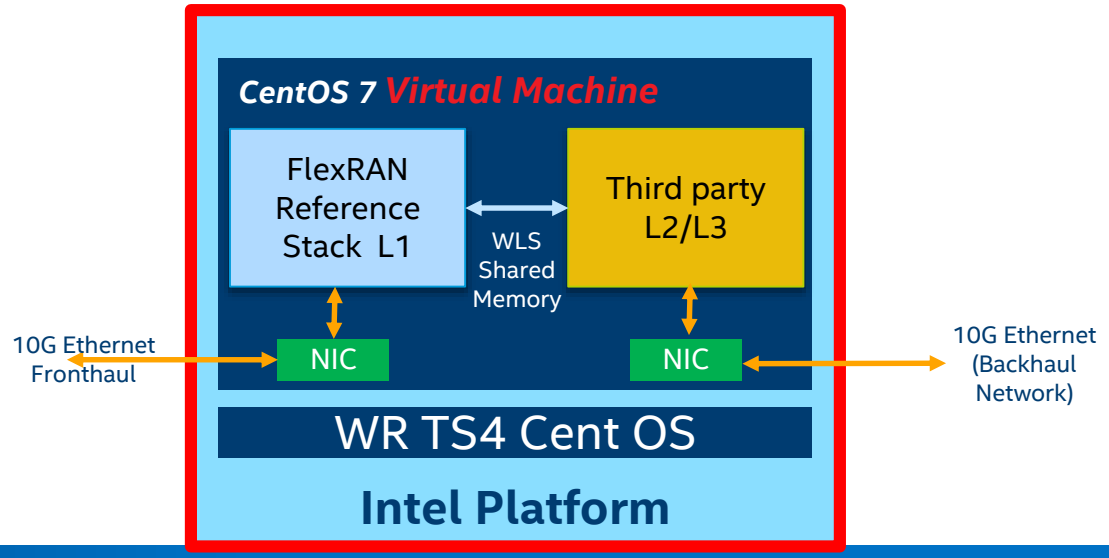
- Platform Overview
- What is Key parameter need to measure and validated.
- What Intel team doing.
  - KPIs for Real-time Processing for Wireless
  - Platform Level Optimization - Hardware and BIOS
  - Operating System and Kernel Level Optimization
  - Workload level Optimizations
  - Results

FlexRAN now supports VM Cloud and Container Cloud

Kubernetes/Docker Container Deployment



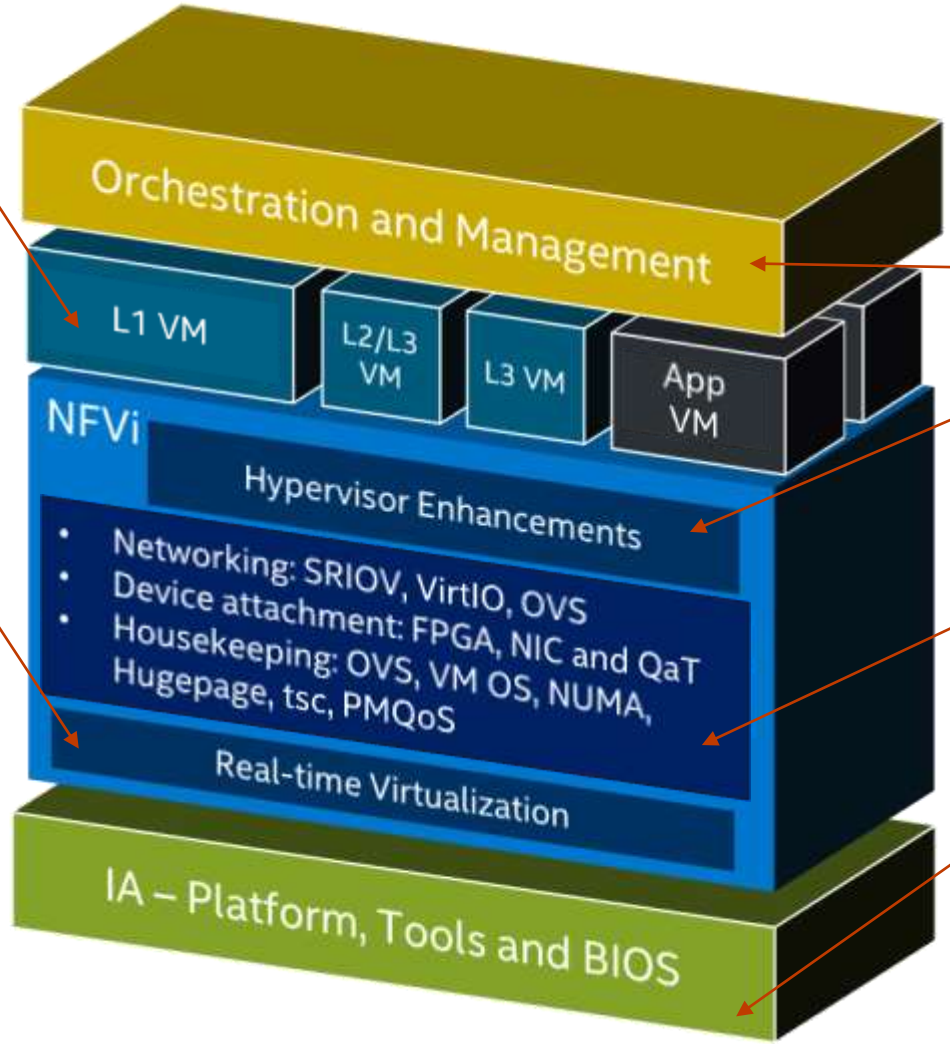
Openstack/VM Deployment



Real-time Platform optimizations for wireless

Kernel: High resolution timer, Full-Preemption, Full NOHz, Disable - (KSM, Hacking, Debug FS) and unwanted drivers and services.
Operating system: Disable IRQ Balance , Kernel file system optimization, Kernel Command line

Kernel: High resolution timer, Full-Preemption, Full NOHz, Disable - (KSM, Hacking, Debug FS) and unwanted drivers and services.
Operating system: Disable IRQ Balance , Kernel file system optimization, Kernel Command line, Disable Machine Check



Openstack: Low latency flavor, Shared core for VCPU0 *(More Optimizations for AVS Core, HT, RDT are part of the POC)*

Qemu: VCPU pinning, Hugepage backed guest, mlock guest, reliable TSC, Low latency VM

SRIOV: NUMA node affinity VM, allocate Huge pages on same NUMA, IRQ affinity setting,
OVS and VirtIO: NUMA, Hugepage Pinning, CPU pinning
Power: Disable PM governor, set to performance, Set max Uncore, tsc reliable
VM OS: Pining to shared core

Processor Optimizations: Hyper threading, C-state, P-state, Prefetcher, VTx/VTd
Memory Optimizations: NUMA, Power management, Frequency
IO Optimizations: ASPM, VTd
SMI:

What Intel team measured today

- Infrastructure
  - Delivery of Device Interrupt
  - Delivery of Timer Interrupt
  - Context Switching overhead
- Application level
  - Packet forwarding
  - Packet Processing
- Usecase level
  - Consolidation and Scaling of workload (within and across the platforms)

Platform Level Optimization - BIOS

Processor Optimizations

Intel® Turbo Boost Technology opportunistically boosts the processor core's frequency to a higher frequency above the qualified frequency depending on the thermal design power (TDP) headroom.

Intel SpeedStep® technology switches both voltage and frequency in tandem between high and low levels in response to processor load. Enhanced Intel SpeedStep® Technology builds upon that architecture using separation between voltage and frequency changes and clock partitioning and recovery.

C-State: CPU C-States are sleep states. This works primarily by turning on/off certain components of the CPU. A lower C-State gives a lower sleep ratio, thus improving latency. Processor C3 state is a sleep state that stops all CPU internal clocks. Processor C6 is a deep sleep state that reduces the CPU internal voltage.

CPU P-States are power states. These work on reducing voltage and frequency of processor cores. Limiting P-States will limit the change in frequency, thus improving determinism. If power management is mandatory in the system design, the first option to consider is using a fixed P-state.

Uncore frequency Scaling should be enabled

Intel® Hyper-Threading Technology (Intel® HT Technology) - Turning hyper-threading off will dedicate entire resources for the core, thus giving better determinism than executing a task with this option turned on.

Intel® Virtualization Technology (Intel® VT) for IA-32, Intel® 64 and Intel® Architecture (Intel® VT-x) represents the hardware virtualization features of the processor. Ensure to enable it if virtual machines are used in system design.

Memory Optimizations

Memory power management can fluctuate the frequency. Keeping constant the frequency will improve real-time performance. Hence, disabling memory power management should improve real-time performance.

Frequency: If permitted by the BIOS settings, set the memory frequency to the maximum supported by both the DIMM module and the mother-board. This further enables to set a constant high frequency.

Enable NUMA optimization and if not needed disable COD/Sub-NUMA cluster

IO Optimizations

If the bottle-neck is in I/O bandwidth, try disabling PCIe* Active State Power Management (ASPM) and verify the performance. This disables the PCIe link power management.

The feature, Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d), enables enhancement for I/O related virtualization. Ensure to turn Intel® VT-d on, if virtual machines are used in the system.

Turn off the USB if this feature is not used. If not, selectively disable unused features such as boot from USB.

Miscellaneous Optimizations

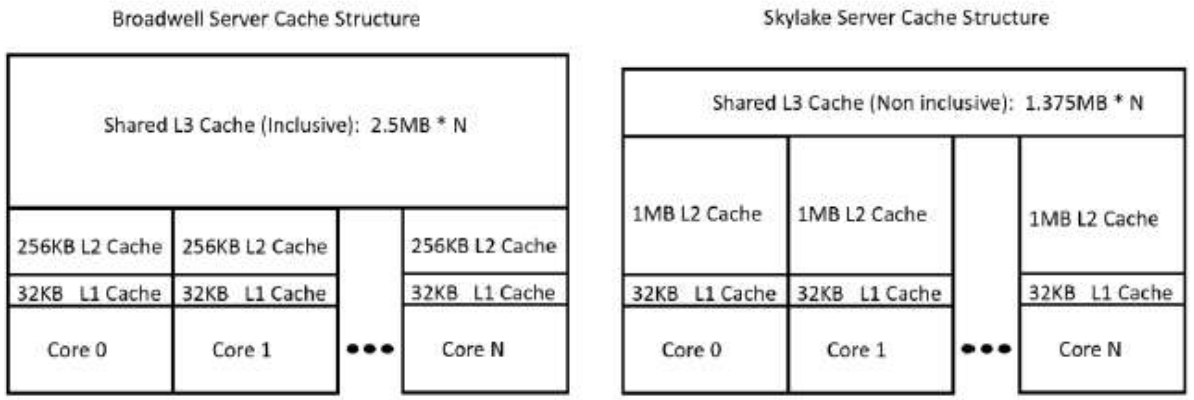
Some BIOS provided give a provision to selectively disable certain SMIs. If available disable the SMI related to the features that are not being used.

Set the Platform to "Performance" mode if the BIOS provides the option

The BIOS setting provided here are high-level guidelines. System Designers need to evaluate the impact of these features on the workload and the deployment needs to enable or disable the features.

Platform Level Optimization - Hardware

- CPU SKUs – Core Count and Cache Implication for Density and Pooling gains



The last level cache (LLC) in Skylake server is a non-inclusive, distributed, shared cache. The size of each of the banks of last level cache has shrunk to 1.375 MB per bank compared to 2.5MB in Broadwell.

- Support for SKUs which support Intel® RDT – CAT, CMT, MBM and MBT
- PCIe Link bandwidth, Training and Socket affinity
- SRIOV support and > 4G addressability for memory map in BIOS
- Understanding the limitation of Intel® DCA on certain PCIe links
- DRAM DIMM Frequency and Type

Kernel and Operating System Level Optimization

- Disable Unwanted Kernel and Operating system Services
- Pre-emptRT Kernel: The first step to enable real-time performance is to ensure that the Linux kernel is patched with the appropriate preempt-RT patch from https://www.kernel.org/pub/linux/kernel/projects/rt and get the equivalent kernel from https://www.kernel.org/pub/linux/kernel/. Refer to https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO for patching instructions.
- Host Kernel Command line
  - `crashkernel=auto biosdevname=0 iommu=pt usbcore.autosuspend=-1 selinux=0 enforcing=0 nmi_watchdog=0 softlockup_panic=0 intel_iommu=on audit=0 cgroup_disable=memory hugepagesz=1G hugepages=21 hugepagesz=2M hugepages=0 default_hugepagesz=1G isolcpus=1-55 rcu_nocbs=1-55 kthread_cpus=0 irqaffinity=0 idle=poll nohz_full=1-55`
- Guest Kernel Commandline
  - `console=tty0 console=ttyS0 biosdevname=0 net.ifnames=0 no_timer_check isolcpus=1-14 irqaffinity=0 default_hugepagesz=1G hugepagesz=1G hugepages=5 clocksource=tsc tsc=perfect intel_idle.max_cstate=0 processor.max_cstate=1 initrd=initramfs.img BOOT_IMAGE=vmlinuz nohz_full=1-14`
  - https://www.kernel.org/doc/html/latest/admin-guide/kernel-parameters.html
- Understand the impact of Kernel threads (kthreads) that run on all cores
  - https://www.kernel.org/doc/Documentation/kernel-per-CPU-kthreads.txt
- Kernel threads and local timer interrupts to the CPU can cause interference to the DPDK/RT thread or VCPU. This can have a major detrimental effect on the latency, in both native and virtualized VMs. The NOHZ host and guest kernel should be used to reduce interference.
- The interrupt balancing daemon irqbalanced tries to spread the interrupt load across the cores of a multi-CPU system. This is good for throughput-oriented workloads, but counterproductive when the shortest possible path for interrupt injection into the guest is required.
- The page fault handling for guests is implemented in an asynchronous manner which can cause arbitrary delays.
- Kernel Samepage Memory tries to reduce the memory consumption of guests by merging pages with the same content and copying them on write.

Operating System Configurations

1. **CPU isolation:** To achieve deterministic latency, dedicated CPUs should be allocated for realtime application. This can be achieved by isolating cpus from kernel scheduler.

2. **IRQ affinity:** All the non-realtime IRQs should be affinitized to non realtime CPUs to reduce the impact on realtime CPUs. Some OS distributions contain an irqbalance daemon which balances the IRQs among all the cores dynamically. It should be disabled as well.

3. **Tickless:** Frequent clock ticks cause latency. CONFIG_NOHZ_FULL should be enabled in the linux kernel. With CONFIG_NOHZ_FULL, the physical CPU will trigger many fewer clock tick interrupts(currently, 1 tick per second).

4. **Preempt-rt:** Reducing time in the interrupt context and move the work to kernel threads

5. **TSC:** Mark TSC clock source as reliable. A TSC clock source that seems to be unreliable causes the kernel to continuously enable the clock source watchdog to check if TSC frequency is still correct. On recent Intel platforms with Constant TSC/Invariant TSC/Synchronized TSC, the TSC is reliable so the watchdog is useless but cause latency.

6. **Idle:** The poll option forces a polling idle loop that can slightly improve the performance of waking up an idle CPU.

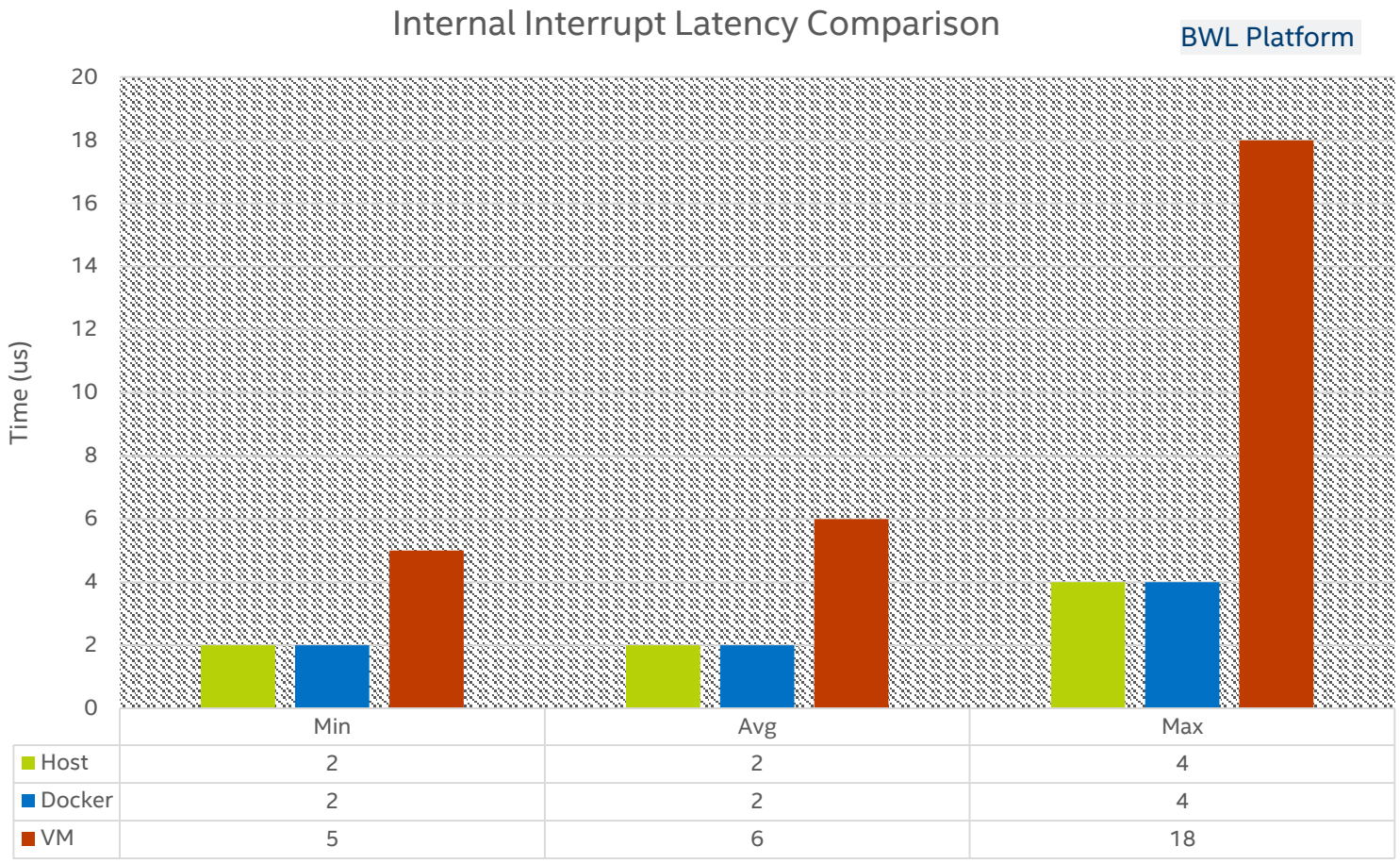
7. **RCU_NOCB:** RCU is a kernel synchronization mechanism. Remove Real-Time CPUs from the candidates for running RCU callbacks

8. **Disable the RT throttling:** RT Throttling is a Linux kernel mechanism that occurs when a process or thread uses 100% of the core, leaving no resources for the Linux scheduler to execute the kernel/housekeeping tasks.

Workload level Optimizations Focus

- Design the workloads to take advantage of multicore
- Avoid locks and inter-core dependency to achieve maximum parallelization
- Pin high priority threads to Isolated CPU/vCPU to avoid contention, and set FIFO
- Use Hugepage backed memory to avoid TLB misses
- Choose the best IO model for achieving parallelization – Centralized or Distributed
- Avoid system calls in the high priority execution threads to avoid losing control to kernel
- Avoid using Disk or emulated device in the RT threads
- Lock the Process address space to the main memory
- Reduce the complexity and the dataset handled by the threads
- Invest time in understanding the size of the task that needs to executed by the threads to achieve maximum parallelization gain
- Consider using acceleration (look aside/inline) to achieve higher density
- Avoid copies of packet and reduce packet in flight
- Consider using IA tools to investigate hotspots and optimization iteratively
- Use C groups and Name spaces to partition the resources when multiple process run in the same domain
- Restrict most of the RT processing in userspace

Real-time Performance



Preliminary Data, Draft data for reference

Measurements shown are preliminary numbers. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests 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. For more complete information visit http://www.intel.com/performance.

Native test HSW-EP@2.2GHz – Simple forwarding one CPU 2- Unidirectional – Packet Size 64-9K - Rate 10-100%

RFC2544 - Throughput/Latency - Aggregated Results - Part 1 of 2

Trial	Frame Size	Iter.	Agg Rx Tput (fps)	Agg Rx Tput (Mbps)	Agg Frame Loss (frames)	Agg Tx Rate (%)	Agg Rx Tput (%)	Avg Latency (ns)	Min Latency (ns)	Max Latency (ns)
1	64	12	14851255	7603.84	0	99.800	99.800	7592.000	4460	21200
1	128	12	8429094	8631.39	0	99.800	99.800	5616.000	4560	19500
1	256	12	4519934	9256.82	0	99.800	99.800	5485.000	4900	19240
1	512	12	2344928	9604.83	0	99.800	99.800	5610.000	5040	19080
1	1024	12	1194924	9788.82	0	99.800	99.800	6097.000	5660	19220
1	1280	12	959616	9826.47	0	99.800	99.800	6212.000	5960	20060
1	1518	12	811118	9850.22	0	99.800	99.800	6522.000	6260	20140
1	2000	12	617574	9881.19	0	99.800	99.800	7111.000	6720	19640
1	4000	12	310323	9930.35	0	99.800	99.800	8724.000	8500	21160
1	6000	12	207226	9946.85	0	99.800	99.800	10746.000	10560	20600
1	9000	12	138304	9957.87	0	99.800	99.800	13636.000	13320	24160

Max Latency: 24us for 9k packet size @99.8% rate other packet sizes reaching 99.8% with lesser latency (at 99% Max latency 19us)

Hyperthreading: Similar latency observed with Hyperthread enabled and DPDK Adjacent Hyperthread disabled

VM test HSW-EP@2.2GHz – Simple forwarding on Non RT vCPU 2 - Unidirectional – Packet Size 64-9K - Rate 10-100%

RFC2544 - Throughput/Latency - Aggregated Results - Part 1 of 2

Trial	Frame Size	Iter.	Agg Rx Tput (fps)	Agg Rx Tput (Mbps)	Agg Frame Loss (frames)	Agg Tx Rate (%)	Agg Rx Tput (%)	Avg Latency (ns)	Min Latency (ns)	Max Latency (ns)
1	64	12	14851255	7603.84	0	99.800	99.800	7441.000	6820	33860
1	128	12	8429094	8631.39	0	99.800	99.800	5424.000	4980	31240
1	256	12	4519934	9256.82	0	99.800	99.800	5310.000	4900	30520
1	512	12	2344928	9604.83	0	99.800	99.800	5430.000	5100	30320
1	1024	12	1194924	9788.82	0	99.800	99.800	5735.000	5480	31060
1	1280	12	959616	9826.47	0	99.800	99.800	6074.000	5800	31360
1	1500	12	820724	9848.69	0	99.800	99.800	6324.000	6100	31880
1	2000	12	617574	9881.19	0	99.800	99.800	6858.000	6660	30940
1	4000	12	310323	9930.35	0	99.800	99.800	8492.000	8300	32800
1	6000	12	207226	9946.85	0	99.800	99.800	10431.000	10240	33000
1	9000	12	138304	9957.87	0	99.800	99.800	13192.000	12980	33200

Max Latency: 33.2us for 64bytes packet size @99.8% rate other packet sizes reaching 99.8% with lesser latency

Preliminary Data, Draft data for reference