TECHNOLOGY GUIDE

Intel Corporation

Enhanced Power Management for Low-Latency Workloads

Authors

David Hunt

Reshma Pattan

Chris MacNamara

Karen Shemer

Nilanjan Palit

Pinkesh Shah

1 Introduction

Reducing carbon emissions across the communications industry is a global initiative with many stakeholders. For network function virtualization (NFV) workloads, opportunity exists to leverage Intel® power management technologies to reduce power consumption at a specific processor frequency to match the immediate workload demand. This can be achieved both at runtime and when idle, delivering a path to reduced power consumption. This in turn makes the workload, processor, and data center energy efficient while supporting many of the ongoing green initiatives.

Recent enhancements to 3rd Generation Intel® Xeon® Scalable processors focus on easing the adoption of power management technology, enabling end users to benefit from power saving while not compromising on the performance requirements of low-latency workloads. Per-core P-states, commonly referred to as “frequency scaling technology”, has been updated to target the latency associated with changing frequency, resulting in less jitter and improved performance.

This technology guide provides a brief overview of the technology encompassing per-core P-states, and how advancements in that technology improve usage in low-latency packet processing applications resulting in a power saving benefit for the user.

This paper is intended for those interested in the new power saving enhancements on the latest Intel® Xeon® Scalable processors.

This document is part of the Network Transformation Experience Kit, which is available at https://networkbuilders.intel.com/network-technologies/network-transformation-exp-kits.

Table of Contents

Figures

Tables

Document Revision History

REVISION	DATE	DESCRIPTION
001	April 2021	Initial release.

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
BIOS	Basic Input/Output System
DPDK	Data Plane Development Kit (DPDK)
GPSS	Global P-state Coordination Timer
NFV	Network Function Virtualization
NIC	Network Interface Card
PMD	Poll Mode Drivers
PMU	Performance Monitoring Unit

1.2 Reference Documentation

Table 2. Reference Documents

REFERENCE	SOURCE
Advanced Configuration and Power Interface Specification	https://www.intel.com/content/dam/www/public/us/en/documents/articles/acpi-config-power-interface-spec.pdf
DPDK L3 Forwarding with Power Management Sample Application	http://doc.dpdk.org/guides/sample_app_ug/l3_forward_power_man.html
DPDK rte_power API	https://doc.dpdk.org/api/rte__power_8h.html
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3 (3A, 3B, 3C & 3D), System Programming Guide	https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-system-programming-manual-325384.html
Power Aware Packet Processing Presentation 1 at DPDK Summit 2017 Presentation	https://fast.dpdk.org/events/slides/DPDK-2017-09-Ireland-Power_Aware.pdf
Power Aware Packet Processing Presentation 2 at DPDK Summit 2017 Presentation	https://www.dpdk.org/wp-content/uploads/sites/35/2018/06/Power-Aware-Packet-Processing.pdf
Recent Power Management Enhancements in DPDK 2018 Presentation	https://www.dpdk.org/wp-content/uploads/sites/35/2018/10/pm-01-DPDK_Summit18_PowerManagement.pdf
Recent Power Management Enhancements in DPDK 2019 Presentation	https://static.sched.com/hosted_files/dpdkbordeaux2019/8b/DPDK_Userspace_2019_PowerManagement_0.9c.pdf
New 3rd Gen Intel® Xeon® Scalable Processor	https://hotchips.org/assets/program/conference/day1/HotChips2020_Server_Processors_Intel_Irma_ICX-CPU-final3.pdf

2 Overview

Intel® Xeon® Scalable Processors support per-core P-states to manage power consumption. Details are provided in the *Intel® 64 and IA-32 Architectures Software Developer's Manual: Volume 3*. However, changing core P-states (frequency) – to optimize for power vs. performance – carried a latency penalty due to the frequency transition time.

In the new 3rd Generation Intel® Xeon® Scalable processors, minimizing the impact of changing power states is addressed with the addition of the Fast Core Frequency Change feature. This new capability allows the core frequency to change from its current operating point to its new target frequency without stopping the clocks, reducing the block time from ~12 µs to ~0 µs (few hundred nanoseconds). The block time is a period when the core is not processing instructions and, in the past, has manifested itself as maximum latency or jitter in latency-sensitive workloads. This latency cost when changing frequency is significantly reduced, broadening the number of use cases that can leverage the per-core P-state power saving technology[1].

In many low-latency NFV designs, workloads can now use the enhanced per-core P-state technology controls to save power, which improves performance per watt by matching the performance and power consumed to the needs of the workload.

In past generations, such as 2nd Generation Intel® Xeon® Scalable processors, design patterns were applied in applications to facilitate the use of per-core P-states and the associated core frequency changes. This typically involved increased buffering to

[1] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

amortize the cost of a frequency change. Now, with the updated P-state transitions architecture in 3rd Generation Intel® Xeon® Scalable processors, the primary performance impacts of a frequency change are addressed. These include the time where the core is not processing instructions, and the maximum time it takes to update to a requested frequency. In one DPDK example using per core P states, a power saving of 30% or ~1.8W per core was observed. This saving was achieved by frequency scaling 30 cores on a 32 core 6338N from 2.2GHz to 800MHz. The CPU consumed 176 watts and when frequency scaled it consumed 123 watts, a saving of 53 watts.

2.1 Technology Overview

This technology guide details the power saving approach and new features that apply to low-latency packet processing workloads. The workloads in question are latency sensitive at the microsecond level and can benefit from power saving strategies. Many workloads benefit from a "slow-down" approach, which is discussed in this technology guide[2]. This paper discusses only DPDK scenarios that can benefit from Per Core P States.

The "slow-down" approach is well suited to packet processing workloads because there may be periods of time where only a few packets are flowing through a server and turning off the core is not an option. Using P-states allows a workload to scale down frequency and voltage dynamically and therefore achieve a lower power operating level.

2.2 What Are P-states?

On an Intel® Xeon® Scalable processor, software can manage the frequency of individual cores in a CPU. The per-core P-states allow each core on a CPU to run at a different frequency or voltage as referenced by the *Advanced Configuration and Power Interface Specification*. The higher the frequency, the more power consumed. In simple terms, if a core is not busy, the frequency can be scaled down to conserve power, and as a result the core runs slower and does not enter a sleep state. When not fully loaded, this is an effective approach for saving power. For more information, refer to the *P-State Hardware Coordination* section in the *Intel® 64 and IA-32 Architectures Software Developer's Manual: Volume 3*.

2.3 What Is a Low-Latency Packet Processing Workload?

In communication workloads, where latency can be a critical part of the design pattern, P-states can be used sparingly to avoid the impact of changing core frequency. The latency cost for changing a core P-state can be divided into two parts:

- The cost of reprogramming and the electrical (voltage and frequency) transition latency for changing the core frequency (previously this was ~12 µs, but now in the sub microsecond range ~0 µs)
- Intel CPUs have implemented a P-state filter that limits the number of P-state transitions in a given period of time to minimize the overall performance impact from such transitions. This limits the minimum reaction time to change a core frequency. In the new 3rd Generation Intel Xeon Scalable processors, this filter interval is now configurable in the BIOS, meaning software requests for a new frequency can happen much faster. This is now BIOS programmable to 500 µs, 50 µs, or 0 µs and improves the response time of a P-state change.

These effects show up in packet processing workloads as an additional (maximum) processing latency or jitter.

An application that uses the Data Plane Development Kit (DPDK, www.dpdk.org) is a typical use case for a low-latency packet processing workload.

2.4 P-state Technology Overview

Figure 1 shows P-state ranges. P1 is the base frequency, which is the as-delivered frequency when Intel® Turbo Boost Technology is not enabled. Software can control the frequency per core up and down from the P1 level shown in Figure 1. As an example, the operating system's P-state driver typically manages the frequency of each core depending on how busy it is, allowing turbo boost when required (if enabled). The P-state frequencies shown are for one SKU and are set for each SKU to be configured in 100 MHz denominations. The polling nature of low-latency workloads based on the DPDK is discussed and an alternative to the OS driver is needed to achieve a power saving benefit. For more information, see *Intel® 64 and IA-32 Architectures Software Developer's Manual: Volume 3*.

[2] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

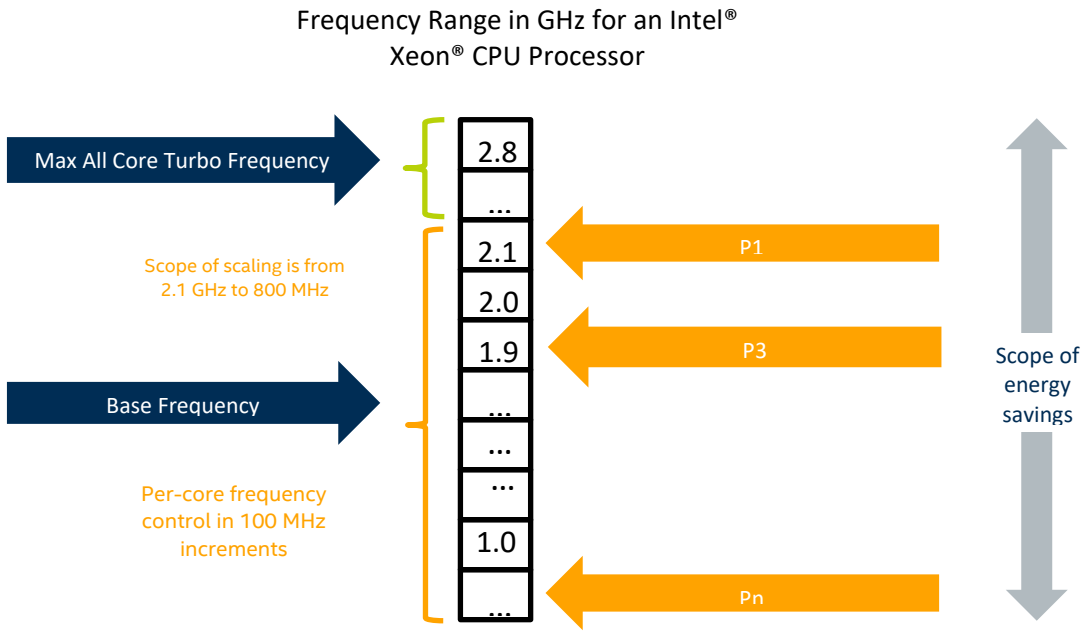

Figure 1. Example P-state Ranges[3]

3 Using P-state Technology with DPDK

Many NFV workloads must deliver on low latency to meet strict service level agreements. Many of these workloads have a unique characteristic. The software typically runs in a polling mode and, from a CPU perspective, utilization shows as 100%. This is because instructions are always being executed by the CPU core. Applications that use the Data Plane Development Kit (DPDK) are typical of this software architecture. To the kernel driver, the poll mode drivers (PMDs) used in the DPDK appear to be 100% busy, even though there may be very few packets flowing. Consequently, the kernel power governors in 100%-utilized cases are not always suitable for power management, because the core utilization is always viewed as 100% busy due to the polling nature of the polling drivers. In addition to polling, DPDK software threads don't typically share cores with other workloads and are typically isolated from OS schedulers, meaning they can control power technology without impact to other software running on the core.

For this reason, the DPDK has a power management library. The library has APIs to allow applications to request P-state changes when they know they should scale up or scale down the core frequencies of the associated workload.

The following sources provide detailed information on the concepts and implementation of power management when using the DPDK:

- Power Aware Packet Processing 2017 v1 (Presentation at DPDK Summit, 2017)
- Power Aware Packet Processing 2017 v2 (Presentation at DPDK Summit, 2017)
- Recent Power Management Enhancements in DPDK 2018 (Presentation at DPDK Summit, 2018)
- Recent Power Management Enhancements in DPDK 2019 (Presentation at DPDK Summit, 2019)
- The Data Plane Development Kit open-source project location https://github.com/DPDK/dpdk
- Power control API for DPDK applications https://doc.dpdk.org/api/rte__power_8h.html (`rte_power` DPDK API)
- A number of DPDK example applications that meet many deployment scenarios are available for reference.
  - A virtual machine power management application named `vm_power_manager` is described and code available on dpdk.org
    https://github.com/DPDK/dpdk/tree/main/examples/vm_power_manager
    https://doc.dpdk.org/guides/sample_app_ug/vm_power_management.html
  - Packet forwarding with DPDK and power controls via l3fwd-power is available on dpdk.org
    https://github.com/DPDK/dpdk/tree/main/examples/l3fwd-power
- Recent poll mode driver updates for power P-states are included in this patchset
  http://patchwork.dpdk.org/project/dpdk/list/?series=14756&state=%2A&archive=both

3.1 Architecture Block Diagram for DPDK Workloads

Figure 2 shows the typical interactions of a DPDK application with the power management subsystem. There are three main use-cases:

[3] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

1. Direct control, where the library calls interact directly with the Linux power governor to set the frequency as desired.
2. Virtualized environments, where the power management requests are sent through a `virtio` channel to a power governor running on the host OS that actions the requests.
3. An out-of-band mechanism, where platform metrics from the performance monitoring unit (PMU) are used to indicate the amount of work being done on a core, and power management decisions are made on behalf of the workload.

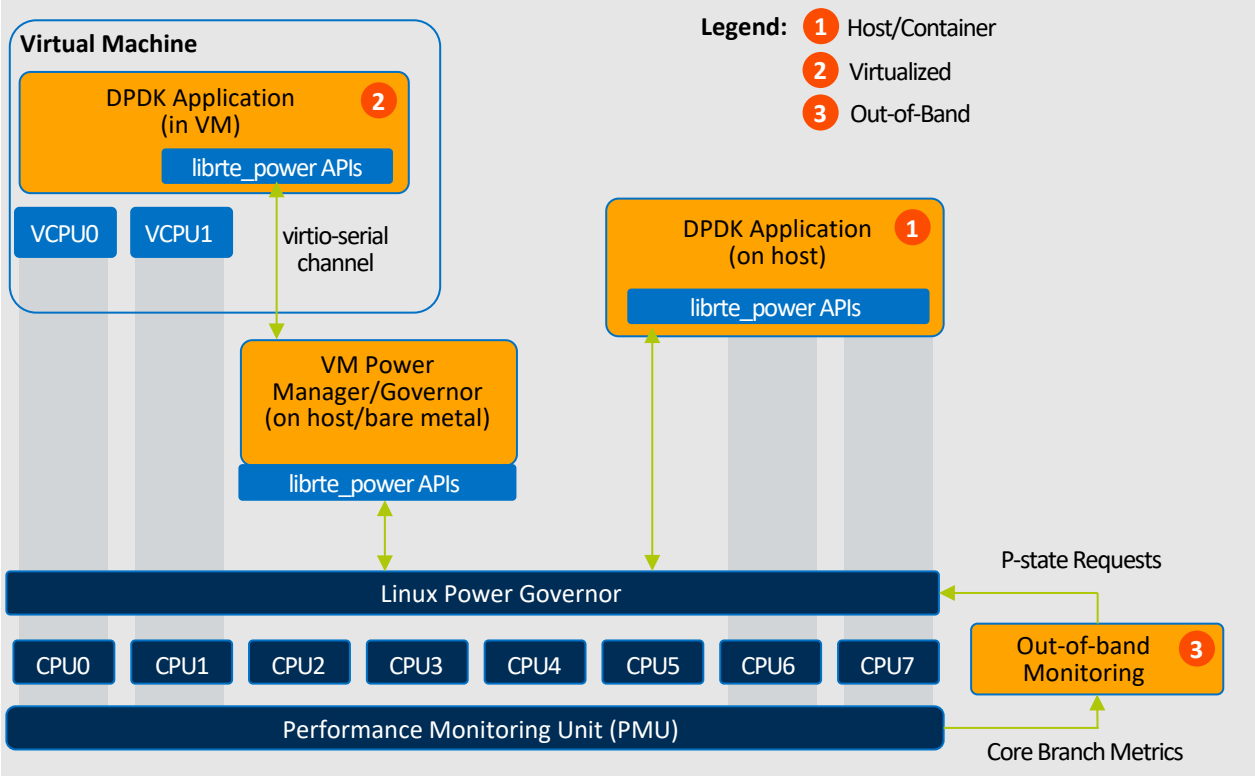

Figure 2. Typical DPDK P-state Interaction Use Cases

4 Technical Review of a P-state Transition

P-state transitions have improved on 3rd Generation Intel® Xeon® Scalable processor CPUs. The transition significantly reduces the Core Frequency Change time from over 10 microseconds to a few hundred nanoseconds. This change from ~12 µs to ~0 µs provides key benefits reducing jitter and max latency that may have been observed in previous generation CPUs when driving P-state transitions[4].

There are two main components in a frequency (P-state) transition:

1. The reaction time is an interval timer – essentially a low-pass filter known as the CPU's Global P-state Sequencing (GPSS) timer – within which requests are gathered and queued. After the timer expires (every 500 µs approximately), the last request for a given core is actioned. On the 3rd Generation Intel® Xeon® Scalable processor, this interval timer is configurable in the BIOS to lower values compared to legacy.
2. After the reaction timer expires, the frequency change portion of the transition starts. On a 2nd Generation Intel Xeon Scalable processor, during this time, no instructions are processed on the core for ~12 us. On a 3rd Generation Intel Xeon Scalable processor, this has been addressed, allowing for faster P-state transitions.

Combined, these components make up the total transition latency for a core's P-state change from request to new (target) frequency.

Figure 3 shows a frequency transition from P-state A to P-state B and back again, identifying the two primary components of frequency transitions. After a frequency change request is made, the CPU's Global P-state Sequencing Timer (GPSS) counts down, and when it expires, the frequency transition occurs.

[4] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

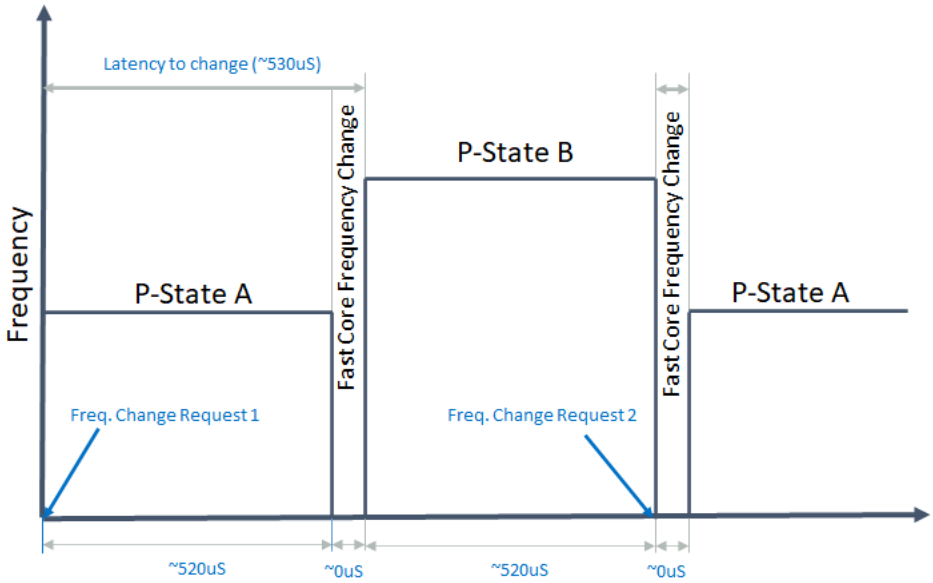

Figure 3. Frequency Transition Components

The two extremes of the time taken to action a frequency change are also shown, indicated as Change Request 1 and Change Request 2. If the change request is made just after a core frequency change, the request will take the full duration of the GPSS timer (while instructions are still being processed), then the core frequency change occurs, and instructions are processed at the new frequency. The fastest transition occurs when a request happens just before the expiry of the GPSS timer (Change Request 2), in which case, the only cost is the core frequency transition. Therefore, the maximum transition is approximately 530 µs (including the full GPSS timer countdown), and the minimum transition time is approximately 12 µs, where there is minimal GPSS timer impact[5].

In 3rd Generation Intel® Xeon® Scalable processors, the GPSS timer duration is configurable in the BIOS, resulting in a faster reaction to frequency change requests. Also, the core frequency transition time has been reduced from approximately 12 µs to a few hundred nanoseconds ~0 µs, meaning far less impact on latency-sensitive workloads. The fast core frequency change does not require any OS software change and is enabled in the processor by default.

4.1 Example Microbenchmark Demonstrating Fast Core Frequency Change

All that is required to observe the latency due to the frequency transition is to repeatedly run a workload of an expected duration and insert some P-state transitions at regular intervals. This needs to be on an isolated core so that the Linux kernel or external interrupts do not affect the results. Refer to Appendix A System Tuning for Latency more information on isolating cores and reducing interrupts.

In the following charts, the known-duration workload is executed many times with the maximum execution time recorded for each batch of runs. When a frequency scale down occurs, the additional ~10 µs is clearly visible until the frequency changes. At that point, the workload takes longer due to the frequency being reduced. Similarly, a frequency scale up causes the workload to return to the original duration, as expected.

Figure 4 demonstrates the effect of the P-state changes on a machine that has a 2nd Generation Intel® Xeon® Scalable processor (without Fast Core Frequency Change) when a frequency change request is made. The ~12 µs additional latency can be clearly observed[6].

[5] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

[6] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

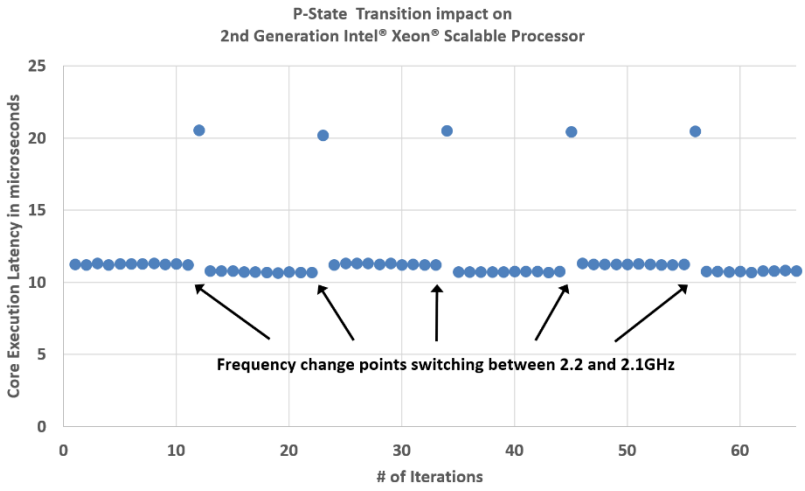

Figure 4. 2nd Generation Intel® Xeon® Scalable Processor P-state Change Request

Figure 5 shows the effect of running the same benchmark on a machine that has a 3rd Generation Intel® Xeon® Scalable processor with the Fast Core Frequency Change feature. The additional 12 µs of latency is no longer present. This demonstrates that latency is significantly reduced on systems with the Fast Core Frequency Change feature.[7]

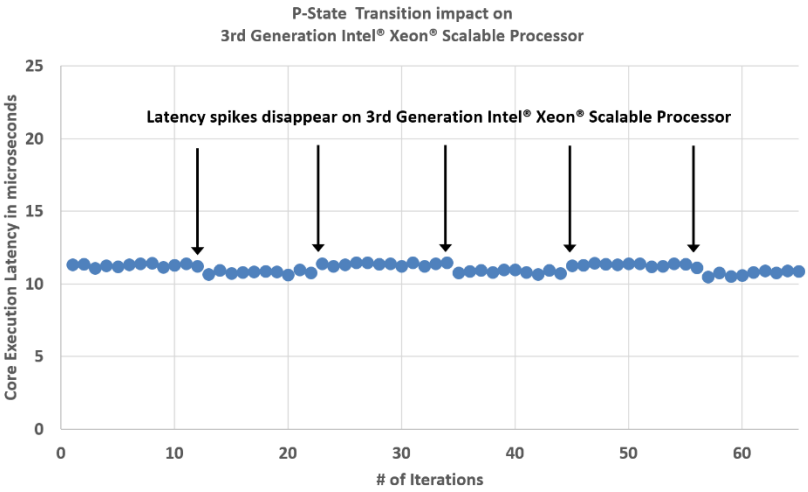

Figure 5. 3rd Generation Intel® Xeon® Scalable Processor P-state Change Request

4.2 Traffic Forwarding Application Demonstrating Fast Core Frequency Change

In previous sections, a microbenchmark was used to analyze the difference in the P-state transitions between 2nd and 3rd Generation Intel Xeon Scalable processors. In this section, we measure the latency of traffic between a network traffic generator and a DPDK forwarding application (`basicfwd`) and observe the per-second latency reported.

The scatter graphs in this section show 200 seconds of traffic, with each dot representing one second of measurements. A frequency change request is issued 10 times each second by writing the frequency request register directly from user space. This ensures that the latency effect of the transition is visible.

In Figure 6, the effect of the core frequency change transition can be seen clearly on the traffic, with all maximum latency figures showing as above 22 µs, and some going as high as 36 µs.

[7] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

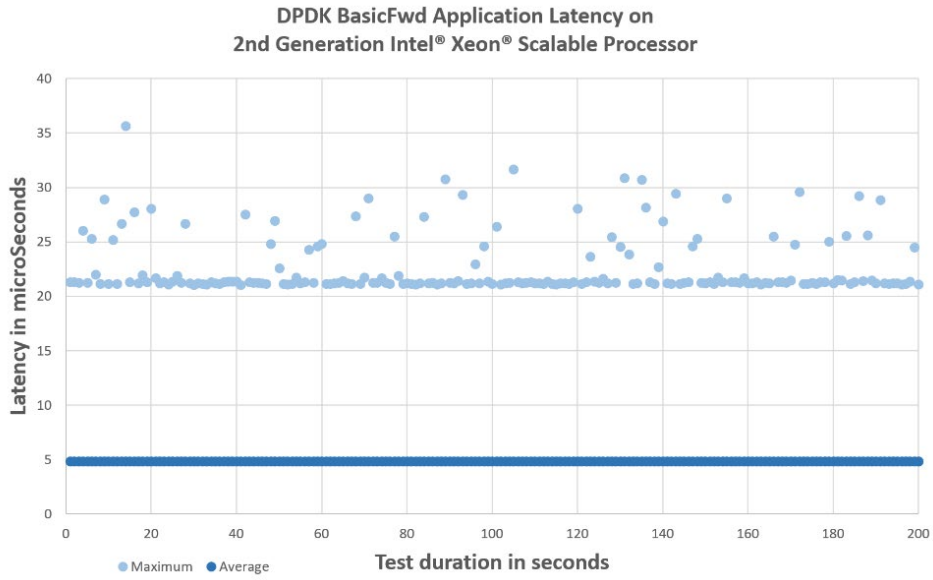

Figure 6. 2nd Generation Intel® Xeon® Scalable Processor Network Traffic Latency

In comparison, when we switch to a 3rd Generation Intel® Xeon® Scalable processor as shown in Figure 7, we see that the latency is much improved due to the Fast Core Frequency Change feature. The additional 10 µs in latency is no longer visible, and the determinism has been improved, with no sample going above 9 µs.[8] Note in Figure 6, the max latency varies up to a max of 36 µs, this is better in the next gen shown in Figure 7.

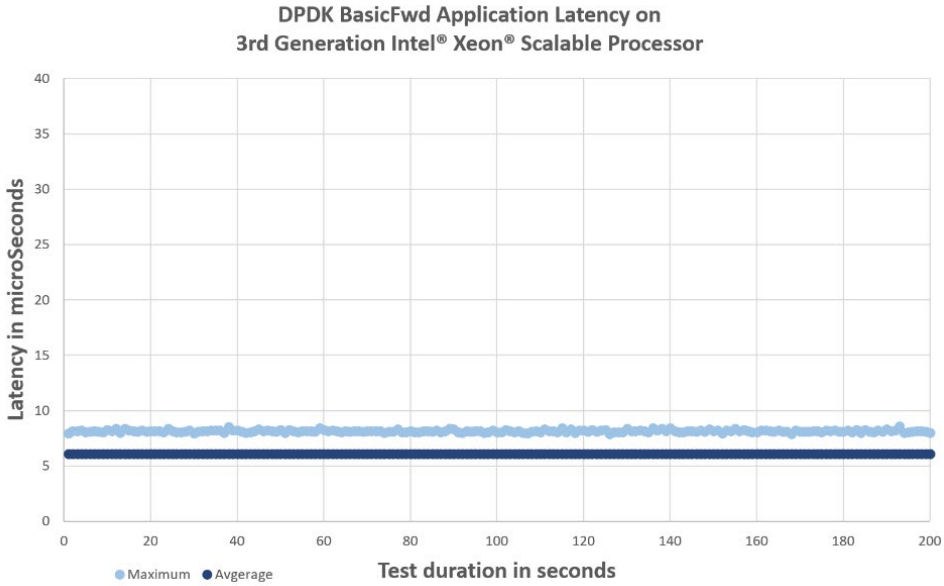

Figure 7. 3rd Generation Intel® Xeon® Scalable Processor Network Traffic Latency

[8] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

4.3 Power Saving Benefit and Motivation

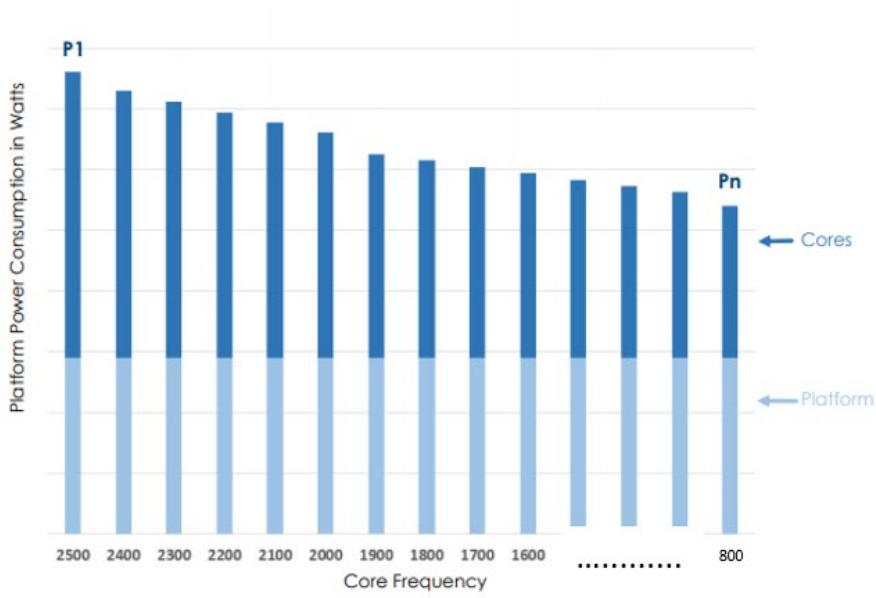

Figure 8. Frequency Scaling and Power Saving

In Figure 8, we show an example of how scaling the core frequency can save power, which is the approach applied to the DPDK use cases discussed in the overview. This involved using the DPDK "l3fwd" example application to forward packets on 30 of the 32 cores on the 6338N with a TDP of 185W. Software reduces the frequency from 2.2 GHz to 800MHz on each of the 30 cores running DPDK. This resulted in a CPU power reduction of 30% from 176 watts to 123 watts and saving of ~1.8W per core. This demonstrates a power saving can be achieved by leveraging frequency scaling for DPDK workloads[9].

It's important to note that the savings described are from just one scenario and savings can vary depending on the product, SKU, architecture, environment, test setup and use case.

5 Summary

In previous CPU generations, power management was possible only if the appropriate design patterns were applied, for example selecting a queue size to absorb the latency increase of 12 µs. With the introduction of 3rd Generation Intel® Xeon® Scalable processors, and the addition of the new capability to reduce the core frequency transition time from approximately 10 µs to under 1 µs, the adoption of power management has become significantly more beneficial.

In addition, for latency-sensitive applications and any workload using P-states, the advantages of the fast core frequency change feature ensure that power management can now be applied with minimal impact on latency.

[9] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

Appendix A System Tuning for Latency

This appendix outlines some of the more important aspects of tuning a system for latency-sensitive measurements and workloads[10].

A.1 BIOS Settings

There are no BIOS settings to enable the Fast Core Frequency Change feature. It is available in all 3rd Generation Intel® Xeon® Scalable processors.

A.2 Kernel Choice

Some distributions are easier than others to get the system to the desired state. Choose a distribution where it is possible to enable `CONFIG_NO_HZ_FULL` on the kernel command line. Some kernels do not compile with this option, and therefore it is not possible to switch the option on without building a custom kernel.

A.3 Kernel Boot Parameters

A.3.1 ISOLCPUS and RCU_NOCBS

Interrupts are usually the main cause of latency outliers. To avoid interrupts, use the `isolcpus` and `rcu_nocbs` parameters to isolate the range of cores on which latency is going to be measured. Example: `isolcpus=20-22 rcu_nocbs=20-22`

A.3.2 NOHZ_FULL

Use the nohz_full option to prevent the kernel from interrupting the core at its tick interval. When enabled, be sure to check that the kernel supports the option by running the `dmesg | grep NO_HZ` command. Example: `nohz_full=20-22`

If you see something like `NO_HZ: Full dynticks CPUs: 20-22`, then it is good. A message saying `unsupported` indicates that a different kernel is required.

A.4 Reliable TSC

To avoid the kernel doing clock resource reliability checking of the Time Stamp Counter (TSC), add `tsc=reliable` to the kernel boot parameters.

A.5 Monitoring Interrupts

A good way to find the source of obvious jitter is to determine if there are any interrupts being issued to your low-latency core. Use the following `bash` script to see what is happening on a specific core:

```
#!/bin/sh
awk 'NR==1 {next}; {if ($('$1'+2) != "0") { print $1,$('$1'+2)} }' /proc/interrupts
```

By running this script, and passing the relevant core number as a parameter, the changes in the number of interrupts occurring on that core should be visible. Run this script several times over a few minutes to see if there are any unexpected interrupts occurring on the core.

A.6 Other Settings

A.6.1 NIC interrupts

NICs bound to kernel drivers can generate interrupts to all cores.

To unbind any unused devices, run:

```
dpdk-devbind.py
```

To see if any interrupts are being sent to core, use:

```
cat /proc/interrupts
```

This is particularly true in the case where the NIC is using a kernel driver with `multiqueue` enabled, in which case there are interrupts sent to every core on the system.

To search for interrupts on a core due to this configuration, run:

```
cat /proc/interrupts | grep TxRx-20
```

where `20` indicates core 20.

[10] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

A.6.2 Fix UNCORE Frequency

Set the UNCORE minimum and maximum frequencies to be equal by writing to the bottom two bytes of the `MSR_UNCORE_RATIO_LIMIT MSR (0x620)` register.

Example: `wrmsr -a 0x620 0x1515`

This example sets the UNCORE minimum and maximum frequency values to `0x15`, which is 2.1 GHz.

A.7 Device Driver Adjustments

A.7.1 Writing Per Core Frequency

To request a frequency on a core write the register using msr-tools. This command requests frequency on core 21 to be 800MHz using the Hardware P-States model specific register 0x774.

"wrmsr -p 21 0x774 0x0808"

A.7.2 Process FIFO Scheduling

To set the scheduling type and priority of the running `dpdk` process, use the command:

```
chrt -f -p 99 `pidof basicfwd`
```

This can stabilize the latency somewhat.

A.8 System Monitoring Tools

Be careful with the monitoring tools that are executed during latency testing. Some monitoring tools can have a significant impact on latency-sensitive workloads. For example, the `turbostat` utility, which is used to query the frequencies and C-states of the cores, can schedule itself on all cores and read several registers, causing significant additional latency on all cores.

A.9 Device Driver Adjustments

Depending on the device driver you are using, tweaks may be necessary for latency as opposed to throughput. For example, the i40e driver is tuned for throughput, and can, on occasion, add 30 µs of latency when flushing out packets remaining in its buffers when traffic stops. This affects the results of latency tests. In the DPDK device driver for the Intel® Ethernet Converged Network Adapter Xx7xx series, in the `i40e_ethdev.h` file, reduce the `I40E_QUEUE_ITR_INTERVAL_DEFAULT` setting from `32` to `0`.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

0421/DN/WIPRO/PDF 637284-001US