Technology Guide

Intel® Turbo Boost Technology - Configure Per Core Turbo Overview

Authors

Reshma Pattan

Hamza Khan

1 Introduction

Intel® Turbo Boost Technology, opportunistically and automatically, allows the processor to run faster than the marked frequency if the part is operating below power, temperature, and the current limits. In a typical system, a subset of cores can benefit from higher frequency. Running at higher frequency consumes more power comparing to cores running at lower frequency. Hence, limiting higher frequency to only the cores requiring it provides a reduced power consumption and optimizes performance per watt. This step-by-step guide details how Intel Turbo Boost Technology can be engaged with any number of cores or logical processors that are enabled and active. This, typically results in increased performance of both multi-threaded and single-threaded workloads.

Intel Turbo Boost Technology enables the turbo frequency range for all processor cores of the system, which lets the cores of the processor to run at turbo frequency. This guide describes the cases where customers want to disable the turbo for some cores without disabling the turbo at a system level. In this document, we describe a recipe to use processor performance states (P-states) to enable or disable the turbo per core basis.

This document shows how to enable or disable the per core turbo on the 3rd Gen Intel® Xeon® Scalable processor (formerly codenamed Ice Lake) and 4th Gen Intel® Xeon® Scalable processor (formerly codenamed Sapphire Rapids) using processor performance states (P-states).

The document also describes the usage of a GitHub repository *CommsPowerManagement* power to enable or disable the turbo on cores.

This document is part of the Network Transformation Experience Kits.

Table of Contents

Tables

Document Revision History

Revision	Date	Description
001	September 2022	Initial release.
002	January 2023	Updated for public distribution on Intel Network Builders.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
BIOS	Basic Input/Output System
CPU	Central Processing Unit
P-States	Performance States

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
CommsPowerManagement	https://github.com/intel/CommsPowerManagement

2 Overview

Intel® Turbo Boost Technology allows core frequencies to boost to turbo frequency range for workloads. Based on the number of active cores running on the system, the maximum turbo frequency up to which the cores can boost up will differ and is predefined on the system. For cases where user wants to disable the turbo behavior on some cores, there is no direct BIOS knob available. In this document, we are proposing a recipe, which leverages the processor P-states to disable the per core turbo. Using the process P-states, the maximum frequency of a core can be limited to the base frequency of the system, so that the cores will never enter the turbo frequency range and stays within the base frequency.

3 System Requirements

3.1 Hardware

3rd and 4th Gen Intel Xeon Scalable processor, which supports turbo boost and hardware P-states.

3.2 BIOS

This section describes the BIOS settings that should be enabled to use the kernel driver `intel_pstate` and `acpi_cpufreq`.

To use the `intel_pstate` driver, user should enable the hardware P-states in the BIOS. The recipe in this document is tested with hardware P-states mode set to "native mode" in the BIOS:

- `Hardware Pstates: native mode`

To use the `acpi_cpufreq` driver, user must disable the hardware P-states in the BIOS as below:

- `Hardware Pstates: disable`

Other common setting to use the turbo frequencies on Intel® platforms is to enable the Turbo mode in BIOS as below:

- `Turbo: ON`

For the demonstration of the recipe in this paper, the below settings are disabled in BIOS. Otherwise, they are optional.

Disable the C-states (sleep states) and disable the hyper threading as below:

- `Cstates: Disabled`
- `Hyper Threading: Disabled`

3.3 Operating System Configuration

The recipe can be configured for both `intel_pstate` driver and `acpi_cpufreq` driver. To use `acpi_cpufreq` driver, the kernel must be booted with "`intel_pstate=disabled`". The `intel_pstate` driver will be enabled on the system by default, if the hardware P-states are enabled in the BIOS as given in section 3.2.

4 Recipe to Configure Turbo and Non-Turbo Cores

Before learning how to configure cores into turbo and non-turbo mode, consider using the script mentioned in section 4.1 for the configuration.

4.1 Cloning CommsPowerManagement

The `CommsPowerManagament` GitHub repository provides the Python-based script *power.py* for configuring the power settings on the intel platforms. The script is used to allow configuration of the recipe on the system. The script configures the power management related kernel *sysfs* files to set the system frequencies and other settings. To access the script, clone the repository from the reference `CommsPowerManagement` given in the Reference Documentation section. The command to clone the repository is

```
$git clone https://github.com/intel/CommsPowerManagement.git
```

```
Note: Python3 must be installed on the systems to run the power.py script
```

4.2 Recipe to Configure 50% Cores in Non-Turbo Mode and 50% Cores in Turbo Mode

Consider a system with two sockets, each containing 28 cores. The system has the P1 base frequency of 2200 Mhz and a maximum turbo frequency as all core turbo frequency of 2600 Mhz. To configure 50% cores in non-turbo mode, set the maximum frequency of the 50% cores to P1 base frequency and the rest of the cores to maximum turbo frequency using the Python script from the repository cloned in section 4.1. Follow the below mentioned steps:

1. Verify that turbo is enabled in the kernel by checking the *sysfs* files for different P-state drivers as shown here:

   If `intel_pstate` driver is used, the below file should have the value set as 0 if turbo is enabled.

   ```
   $ cat /sys/devices/system/cpu/intel_pstate/no_turbo
   ```

   If `acpi_cpufreq` driver is used, the below file should have the value set as 1 if turbo is enabled.

   ```
   $ cat /sys/devices/system/cpu/cpufreq/boost
   ```

2. Check what is the supported turbo frequency range and P1base frequency of the system. Use the below command to view this information. The P1 base frequency information and maximum turbo frequency should be used for configuring the turbo and non-turbo cores.

   ```
   $ ./power.py -i
   ```

3. Set 50% of cores of the system to non-turbo mode by setting the maximum core frequency to the P1 base frequency of 2200 Mhz on cores 0 -13.

   ```
   $ ./power.py -M 2200 -r 0-13
   ```

4. Set 50% of cores of the system to turbo mode by setting the maximum core frequency to the all core turbo frequency of 2600 Mhz on cores 14 - 27.

   ```
   $./power.py -M 2600 -r 14-27
   ```

   The script sets the maximum frequency for the cores to the below *sysfs* files.

   /`sys/devices/system/cpu/cpu<id>/cpufreq/scaling_max_freq`

   `Example:` `/sys/devices/system/cpu/cpu14/cpufreq/scaling_max_freq`

4.3 Run stress-ng

You can view if the cores are configured correctly and are running at desired frequencies. The `stress-ng` workload can be used on the Linux machine by installing the `stress-ng` relevant packages for your Linux distribution. Run the following stress-ng template command to load the cores with N workers executing specified matrix method on a matrix of specified size for the specified time duration. It is suggested to refer to the stress-ng manual to understand more details on the command line options available.

Template: `$stress-ng -c <core number> --matrix <N> --matrix-method <method> --matrix-size <size> -t <timeout>`

```
Below command is the example where the core 1 will run 5 workers executing the matrix
multiplication for a matrix size of 2048 for the duration of 1 hour.
```

Example: `$stress-ng -c 1 --matrix 5 --matrix-method mult --matrix-size 2048 -t 1h`

User can opt to disable the C-states before running the above workload. This will help in clearly observing if the cores are scaling up to the desired set frequencies during the workload.

Note: For the purpose of this document, we have disabled the C-states in BIOS. C-states, which cannot be disabled in BIOS like C1, can be disabled using the `power.py` script as shown below:

```
$./power.py -d <cstate name> -r 0-27
```

Example: `$./power.py -d C1 -r 0-27`

The script uses below *sysfs* files to disable the C-states.

```
/sys/devices/system/cpu/cpu0/cpuidle/state<id>/disable
```

The name of a particular C-state can be found from the below file:

```
cat /sys/devices/system/cpu/cpu<id>/cpuidle/state<id>/name
```

4.4 Checking Cores Running in Turbo and Non-Turbo Mode

To see if cores are running in turbo and non-turbo mode, run `turbostat` command on the cores 0 - 27 and observe the `Bzy_Mhz` colum of the `turbostat` output. Cores 0 - 13 should be running at P1 base frequency of 2200 Mhz and cores 14 - 27 should be running at the all core turbo frequency of 2600Mhz.

```
$ ./turbostat -c 0-27 --quiet -i 1
```

Here is a sample of the `turbostat` output.

Package	Core	CPU	Avg_MHz	Busy%	Bzy_MHz
-	-	-	2474	100.00	2480
0	0	0	2195	100.00	2200
0	1	1	2195	100.00	2200
0	2	2	2195	100.00	2200
0	3	3	2195	100.00	2200
0	4	4	2195	100.00	2200
0	5	5	2195	100.00	2200
0	6	6	2195	100.00	2200
0	7	7	2195	100.00	2200
0	8	8	2195	100.00	2200
0	9	9	2195	100.00	2200
0	10	10	2195	100.00	2200
0	11	11	2195	100.00	2200
0	12	12	2195	100.00	2200
0	13	13	2195	100.00	2200
0	14	14	2581	100.00	2587
0	15	15	2572	100.00	2578
0	16	16	2582	100.00	2588
0	17	17	2567	100.00	2573
0	18	18	2594	100.00	2600
0	19	19	2567	100.00	2573
0	20	20	2552	100.00	2558
0	21	21	2570	100.00	2576
0	22	22	2567	100.00	2573
0	23	23	2558	100.00	2564
0	24	24	2561	100.00	2567
0	25	25	2556	100.00	2562
0	26	26	2564	100.00	2570
0	27	27	2581	100.00	2587

5 Summary

This document describes how to configure per core turbo on the 3rd Gen Intel Xeon Scalable processor. This enables the user to choose the set of cores to run in turbo mode and some cores to non-turbo mode. The recipe will benefit for controlling the per core turbo behavior and use some cores at their base frequency in order to save the operating power.

0123/DN/WIT/PDF 739138-002US