AMD EPYC

TUNING GUIDE

AMD EPYC™

LOW LATENCY TUNING FOR AMD EPYC CPU-POWERED SERVERS

AMD
together we advance_data center computing

PID: 58649
v1.1
July 2024

Nimisha Raut, Sylvester Rajasekaran, and Chuck Newman

© 2024 Advanced Micro Devices, Inc. All rights reserved.

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks

AMD, the AMD Arrow logo, AMD EPYC, 3D V-Cache, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names and links to external sites used in this publication are for identification purposes only and may be trademarks of their respective companies.

* Links to third party sites are provided for convenience and unless explicitly stated, AMD is not responsible for the contents of such linked sites and no endorsement is implied.

DATE	VERSION	CHANGES
July, 2024	1.0	Initial release
July, 2024	1.1	Updated attached scripts.

LOW LATENCY TUNING FOR AMD EPYC CPU-POWERED SERVERS

CONTENTS

Chapter 1: Introduction

Various market segments require servers to consistently respond in under 10 µs. For example, financial services companies that deal with High-Frequency Trading (HFT) require consistently fast trade execution to maintain profitability. Under these circumstances, a difference of microseconds can have major business impacts. Ultra-low server and network latencies are therefore crucial.

This Tuning Guides provides guidance for tuning servers powered by high-frequency processors to meet these stringent low latency requirements by reducing unwanted Jitter. These guidelines cover hardware configuration, BIOS settings, operating system kernel configurations, and scripts that control the environment of the target applications. Please pay particular attention to "Hardware Configuration" on page 3, which describes high-level system configuration information used for the low-latency tuning described in this Tuning Guide. This information is especially important because this Tuning Guide clearly focuses on the server resources and tuning both the BIOS and OS to obtain optimal performance for HFT and similarly stringent environments.

This Tuning Guide includes and expands on the content previously published in *Performance Tuning Guidelines for Low Latency Response on AMD EPYC™ 7002 Series Processor Based Servers* (available from the AMD Documentation Hub) by including 3rd and 4th Gen AMD EPYC processors in addition to 2nd Gen AMD EPYC processors.

1.1 - Sysjitter

Sysjitter measures Jitter events and relative counters of the cores under observation and reports the count and duration of interrupts encountered during Sysjitter runtime. This helps HFT users tune their servers to meet their needs. See "Sysjitter" on page 25 to learn more about running Sysjitter using a tailored script and how to interpret results.

THIS PAGE INTENTIONALLY LEFT BLANK.

READY TO CONNECT? Visit www.amd.com/epyc

AMD
together we advance_data center computing

Chapter 2: Hardware Configuration

This Tuning Guide is based on findings from different generations of high-frequency AMD EPYC processors. Achieving µs latency requires understanding the hardware and firmware configuration of the System Under Test (SUT). Some of the important factors that affect response times include:

- Number of cores
- Execution threads per core
- Number of sockets
- Number of NUMA nodes
- CPU and memory arrangement in the NUMA topology
- Cache topology in a NUMA node.

Workloads that are not limited by memory bandwidth may benefit from running the memory at 4800 MT/s to synchronize with the AMD Infinity Fabric, such as when using 4th Gen AMD EPYC processors where the Infinity Fabric runs at 2400 MHz. Linux-based system tools such as `ipmitool`, `hwloc-ls`, `lshw`, and `dmidecode` display the configuration at varying levels of detail and in various formats.

Note: This Tuning Guide is intended for high frequency AMD EPYC SKUs such as the AMD EPYC 7F32, AMD EPYC 75F3, AMD EPYC 9374F, and AMD EPYC 4564P processors and may be inappropriate for servers with general purpose or cache-enhanced AMD EPYC processors. High frequency AMD EPYC SKUs are typically designated by an "F" in the SKU number.

THIS PAGE INTENTIONALLY LEFT BLANK.

READY TO CONNECT? Visit www.amd.com/epyc

AMD
together we advance_data center computing

Chapter 3: BIOS Configuration

You can eliminate many sources of system latency by properly configuring certain BIOS settings. This chapter recommends BIOS parameter tunings for low latency applications. Your OEM BIOS may vary from the default AMD options shown. Please work with your OEM to tune the corresponding BIOS parameter(s). The recommendations contained in this Tuning Guide are based on testing the processors listed in Table 3-1 and apply to all high-frequency AMD EPYC SKUs plus the Ryzen SKU shown.

	AMD EPYC 7F32	AMD EPYC 75F3	AMD EPYC 9374F	AMD EPYC 4564P
Base Frequency	3.7 GHz	2.95 GHz	3.85 GHz	4.5 GHz
Max Boost*	3.9GHz	4.0 GHz	4.3 GHz	5.7 GHz
# of Cores	8	32	32	16
L1 Cache Size	32 KB I + 32 KB D on chip per core	32 KB I + 32 KB D on chip per core	32 KB I + 32 KB D on chip per core	32 KB I + 32 KB D on chip per core
L2 Cache Size	512 KB I+D on chip per core	512 KB I+D on chip per core	1024 KB I+D on chip per core	1024 KB I+D on chip per core
L3 Cache Size	128 MB I+D on chip per chip, 16 MB per core	256 MB I+D on chip per chip, 8 MB per core	256 MB I+D on chip per chip, 8 MB per core	64 MB per NUMA zone
Max Memory Speed	3200 MT/s	3200 MT/s	4800 MT/s	5200 MT/s
PCIe® Support	128 lanes (max) PCIe Gen4	128 lanes (max) PCIe Gen4	128 lanes (max) PCIe Gen5	24 lanes (usable) 28 lanes (total) PCIe Gen5

Table 3-1: AMD processors tested for this Tuning Guide

3.1 - Considerations

Always review these important considerations below before building a low-latency system:

- Optimize the system topology where possible to match your operational needs. Be aware of the memory placement, install memory evenly across the NUMA nodes, and try to maximize the use of local memory. Place processes that share memory into the same cache domain. Isolate the cores executing your time- critical application from the operating system scheduler so that other applications and kernel threads do not steal execution time from your application.
- Isolate the application's cores from interrupts as much as possible. Use the Linux utilities and techniques described in the following Tuning Guides, which are available from the AMD Documentation Hub:
  - *Linux® Network Tuning Guide for AMD EPYC™ 7003 Series Processors*
  - *Linux® Network Tuning Guide for AMD EPYC™ 9004 Series Processors*
- Enabling **AMD Core Performance Boost** obtains maximum CPU core performance. Monitor the operating frequencies of individual cores while running your workload to determine whether the maximum frequency should be capped using the **AMD Fmax Boost Limit** parameter to maintain deterministic timing behavior because high fluctuations in CPU frequencies limit consistent system response. For the same reason, set the **Determinism Control** to **Performance Deterministic**. See Power/Performance Determinism for additional information.
- Please refer to the following Tuning Guides (available from the AMD Documentation Hub) for RHEL-specific tuning information:
  - *Red Hat Enterprise Linux® Tuning Guide for AMD EPYC™ 7003 Series Processors*
  - *Red Hat Enterprise Linux® Tuning Guide for AMD EPYC™ 9004 Series Processors*
- Please also see these additional Tuning Guides, which are also available from the AMD Documentation Hub:
  - *Workload Tuning Guide for AMD EPYC™ 7003 Series Processors*
  - *BIOS and Workload Tuning Guide for AMD EPYC™ 9004 Series Processor Based Servers*
  - *Performance Tuning Guidelines for Low Latency Response on AMD EPYC™ 7002 Series Processor Based Servers*
  - *Financial Services Industry Tuning Guide for AMD EPYC™ 9004 Series Processors*

3.2 - BIOS Profile Settings

Check your computer BIOS setup utility (provided by your server manufacturer) for any available pre-built profiles. You may be able to optimize one of these profiles for low latency. If no profiles are available, then you will need to adjust your system BIOS settings to create your custom `Low-Latency` workload profile, as described in "Custom Profile" on page 7. Please also refer to your OEM documentation for more information. Some possible references include:

- HPE ProLiant Gen 11 Workload profile dependencies for fourth Gen AMD EPYC™ processors* (see the Low Latency profile)
- DELL PowerEdge BIOS Performance and Workload profile (Detailed AMD BIOS profile settings)*
- Lenovo ThinkSystem Server with AMD EPYC™ processor (4th Gen) UEFI Manual*

This section examines two sample OEM pre-built BIOS profiles as an example:

- Low Latency
- Custom

3.2.1 - Low Latency Profile

If your system includes pre-built BIOS profiles, then AMD recommends first choosing the **Low Latency** BIOS profile. To do this:

1. From the BIOS **Platform Configuration** menu, select the **Low Latency** BIOS profile. This option enables the low-latency BIOS tuning options set by the OEM engineers.
2. Reboot the server, and then tune the OS as described later in this Tuning Guide.
3. Run Sysjitter workloads to observe the core jitters.

   If you are not getting the expected low latency and low Jitter behavior on the CPU cores, then select the **Custom** BIOS profile and adjust individual BIOS settings as described below.
4. Run the checks listed in "Post-Reboot System Checks" on page 9 to make sure that your environment is ready for further tuning.

3.2.2 - Custom Profile

If your system does not include a pre-built low-latency BIOS profile, then you can either:

- Select the **Custom** BIOS profile, if available, and then follow Steps 1-7 of the procedure below.
- Adjust BIOS settings as needed without selecting the **Custom** profile by following Steps 2-7 of the procedure below.

To manually select BIOS parameters:

1. If available, from the BIOS **Platform Configuration** menu, select the **Custom** BIOS profile. This option allows you to manually tune BIOS parameters.
2. Manually adjust the BIOS parameters as shown in Table 3-2, below.
3. Reboot the server.
4. Observe the initialization screen that displays the selected BIOS profile option to verify that your custom profile is active.
5. Tune the OS as described later in this Tuning Guide.
6. Run the Sysjitter workload to observe the core jitter.
7. Run the checks listed in "Post-Reboot System Checks" on page 9 to make sure that your environment is ready for further tuning.

BIOS Parameters	Recommended Settings
Hyperthreading (SMT)	Disabled
NumaMemoryDomainsPerSocket	4* (Modify this setting based on your specific workload. Please see the appropriate *High Performance Computing* and/or *Financial Services Industry* Tuning Guide for your AMD EPYC processor (available from the AMD Documentation Hub.)
LastLevelCacheAsNUMANode	Disabled
PerformanceDeterminism	Performance
ApplicationPowerBoost	Enabled
ProcAMDBoost	Enabled
Turboboost	Enabled
Thermal Configuration	Optimal Cooling
Memory Patrol Scrubbing	Disabled
InfinityFabricPstate	P0
NumaGroupSizeOpt	Clustered
DataFabricCStateEnable	Disabled
Dynamic Power Capping	Disabled
AMD Memory Interleaving	Disabled
Virtualization	Disabled
Core C-states	Disabled
Prefetchers	Enabled
Periodic Directory Rinse	Adaptive

Table 3-2: Recommended custom BIOS parameter settings

Here is a sample low-latency profile initialization screen message:

```
Workload Profile: Low Latency
Power Regulation Mode: Static High Performance
Advanced Memory Protection Mode: Advanced ECC Support
Boot Mode: UEFI
```

Here is a custom profile initialization screen message:

```
Workload Profile: Custom
Power Regulation Mode: Static High Performance
Advanced Memory Protection Mode: Advanced ECC Support
Boot Mode: UEFI
```

3.3 - Post-Reboot System Checks

Perform all checks described below to verify that your system is ready for further tuning.

3.3.1 - BIOS, Memory, and CPU Version

These sample outputs are from executing the `xsos` command on a HPE system that shows the BIOS, memory, and CPU versions.

```
# yum install http://people.redhat.com/rsawhill/rpms/latest-rsawaroha-release.rpm

# yum install xsos rsar

# xsos --bios (produces the below output) DMIDECODE BIOS:
Vend: HPE Vers: 1.10
Date: 09/15/2022 BIOS Rev: 1.10 FW Rev: 1.10
System: Mfr: HPE
Prod: ProLiant DL325 Gen11 Vers: Not Specified Ser: DL3x5GEN11
UUID: 00313147-0000-4c44-3378-3547454e3131 CPU:
1 of 1 CPU sockets populated, 32 cores/64 threads per CPU
32 total cores, 64 total threads Mfr: Advanced Micro Devices, Inc. Fam: Zen Freq: 3850 MHz
Vers: AMD EPYC 9374F 32-Core Processor Memory:
Total: 393216 MiB (384 GiB)
DIMMs: 12 of 60 populated
MaxCapacity: 3145728 MiB (3072 GiB / 3.00 TiB) OS
Hostname: gstacn1-sut.amd.com
Distro:[redhat-release] Red Hat Enterprise Linux release 8.6 (Ootpa) [os-release] Red Hat Enterprise Linux
8.6 (Ootpa) 8.6 (Ootpa)
RHN:(missing)
RHSM:hostname = subscription.rhsm.redhat.com proxy_hostname =
YUM:3 enabled plugins: debuginfo-install, product-id, subscription-manager Runlevel: N 3 (default multi-
user)
SELinux: enforcing (default disabled)
Arch:      mach=x86_64 cpu=x86_64 platform=x86_64 Kernel:
Booted kernel: 4.18.0-372.26.1.el8_6.x86_64 GRUB default:
Build version:
Linux version 4.18.0-372.26.1.el8_6.x86_64 (mockbuild@x86-vm-08.build.eng.bos.redhat.com) (gcc version
8.5.0 20210514 (Red Hat 8.5.0-10) (GCC)) #1 SMP Sat Aug 27 02:44:20 EDT 2022
Booted kernel cmdline:
BOOT_IMAGE=(hd2,gpt2)/vmlinuz-4.18.0-372.26.1.el8_6.x86_64 root=/dev/mapper/rhel-root ro crashkernel=auto
resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
GRUB default kernel cmdline:
Taint-check: 0 (kernel untainted)
- - - - - - - - - - - - - - - - - - - -
Sys time: Thu Sep 22 12:55:58 PDT 2022
Boot time: Thu Sep 22 11:49:31 PDT 2022 (epoch: 1663872571) Time Zone: America/Los_Angeles
Uptime:      1:06, 1 user
LoadAvg:      [64 CPU] 0.00 (0%), 0.00 (0%), 0.00 (0%)
/proc/stat:
procs_running: 1      procs_blocked: 0      processes [Since boot]: 49094 cpu [Utilization since boot]:
us 0%, ni 0%, sys 0%, idle 100%, iowait 0%, irq 0%, sftirq 0%, steal 0%
IP4
Interface        Master IF MAC Address     MTU     State    IPv4 Address
=========        ========= ================      ======   =====    ==================
lo       -       -         65536   up      127.0.0.1/8
enp1s0f4u2u3 -   00:e0:4c:68:08:331500     up      10.216.181.163/22??SSH
ens1f0       -   ec:0d:9a:d4:29:1c1500     up      -
ens1f1       -   ec:0d:9a:d4:29:1d1500     up      -
Interface        Master IF MAC Address     MTU     State    IPv4 Address
=========        ========= ================      ======   ====     ==================
lo       -       -         65536   up      127.0.0.1/8
enp1s0f4u2u3 -   00:e0:4c:68:08:331500     up      10.216.181.163/22??SSH
ens1f0       -   ec:0d:9a:d4:29:1c1500     up      -
ens1f1       -   ec:0d:9a:d4:29:1d1500     up      -
```

3.3.2 - NUMA Node

Executing `numactl -H` shows the NUMA nodes.

```
available: 4
node 0 cpus:   nodes
0 1 2    (0-3)
3 4 5 6 7
node     0     size:    96416   MB
node     0     free:    81283   MB
node     1     cpus:    8 9 10 11 12 13 14 15
node     1     size:    96766   MB
node     1     free:    96222   MB
node     2     cpus:    16 17   18 19 20 21 22 23
node     2     size:    96766   MB
node     2     free:    96315   MB
node     3     cpus:    24 25   26 27 28 29 30 31
node     3     size:    96721   MB
node     3     free:    82053   MB
node     distances:
node     0       1      2       3
0:      10 12 12 12
1:      12 10 12 12
2:      12 12 10 12
3:      12 12 12 10
```


READY TO CONNECT? Visit www.amd.com/epyc

AMD
together we advance_data center computing

3.3.3 - Processor

This section shows sample `lscpu` output for a 1P system powered by an AMD EPYC 9374F processor.

```
# lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Address sizes:                   52 bits physical, 57 bits virtual Byte Order:         Little Endian
CPU(s):                          32
On-line CPU(s) list:             0-31
Vendor ID:                       AuthenticAMD
Model name:                      AMD EPYC 9374F 32-Core Processor CPU family:          25
Model:                           17
Thread(s) per core:              1
Core(s) per socket:              32
Socket(s):                       1
Stepping:                        1
BogoMIPS:                        7688.45
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl
nonstop_tsc cpuid extd_apicid aperfmperf rapl pni pclmulqd q monitor ssse3 fma cx16 pcid sse4_1 sse4_2
movbe popcnt xsave avx f16c rdrand lahf_lm cmp_legacy svm extapic cr8_legacy abm sse4a misalignsse
3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core perfctr_nb bpext perfctr_llc mwaitx cpb cat_l3
cdp_l3 invpcid_single hw_pstate ssbd mba perfmon_v2 ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2
erms invpcid cqm rdt_a avx512f avx512dq rdseed adx smap avx512ifma clflushopt clwb avx512cd sha_ni avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local avx512_bf16
clzero irperf xsaveerptr rdpru wbnoinvd amd_ppin cppc arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean
flushbyasid decodeass ists pausefilter pfthreshold avic v_vmsave_vmload vgif x2avic v_spec_ctrl avx512vbmi
umip pku ospke avx512_vbmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg avx512_vpopcntdq la57 rdpid
overflow_recov succor smca fsrm flush_l1d
Virtualization features: Virtualization:  AMD-V
Caches (sum of all):
L1d:                             1 MiB (32 instances)
L1i:                             1 MiB (32 instances)
L2:                              32 MiB (32 instances)
L3:                              256 MiB (8 instances) NUMA:
NUMA node(s):                    4
NUMA node0 CPU(s):               0-7
NUMA node1 CPU(s):               8-15
NUMA node2 CPU(s):               16-23
NUMA node3 CPU(s):               24-31
Vulnerabilities:
Gather data sampling: Not affected Itlb multihit:    Not affected
L1tf:                            Not affected
Mds:                             Not affected
Meltdown:                        Not affected Mmio stale data: Not affected Retbleed:     Not affected
Spec rstack overflow: Mitigation; Safe RET
Spec store bypass:               Mitigation; Speculative Store Bypass disabled via prctl
Spectre v1:                      Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Spectre v2:                      Mitigation; Retpolines, IBPB conditional, IBRS_FW, STIBP disabled, RSB
filling, PBRSB-eIBRS Not affected
Srbds:                           Not affected Tsx async abort:   Not affected
```

3.3.4 - DRAM

Executing `dmidecode` (`-t 17` -> memory) shows DRAM details.

```
# dmidecode -t 17
# dmidecode 3.3
Getting SMBIOS data from sysfs. SMBIOS 3.4.0 present.
# SMBIOS implementations newer than version 3.3.0 are not # fully supported by this version of dmidecode.

Handle 0x0027, DMI type 17, 92 bytes
Memory Device
        Array Handle: 0x0016
        Error Information Handle: Not Provided
        Total Width: 80 bits
        Data Width: 64 bits
        Size: 32 GB
        Form Factor: DIMM
        Set: None
        Locator: PROC 1 DIMM 1
        Bank Locator: Not Specified
        Type: DDR5
        Type Detail: Synchronous Registered (Buffered)
        Speed: 4800 MT/s
        Manufacturer: Micron
        Serial Number: 36593900
        Asset Tag: Not Specified
        Part Number: MTC20F1045S1RC48BA2
        Rank: 1
        Configured Memory Speed: 4800 MT/s
        Minimum Voltage: 1.1 V
        Maximum Voltage: 1.1 V
        Configured Voltage: 1.1 V
        Memory Technology: DRAM
        Memory Operating Mode Capability: Volatile memory
        Firmware Version: Not Specified
        Module Manufacturer ID: Bank 1, Hex 0x2C
        Module Product ID: Unknown
        Memory Subsystem Controller Manufacturer ID: Unknown
        Memory Subsystem Controller Product ID: Unknown
        Non-Volatile Size: None
        Volatile Size: 32 GB
        Cache Size: None
        Logical Size: None
…
…
…
```


READY TO CONNECT? Visit www.amd.com/epyc

AMD
together we advance_data center computing

Chapter 4: Operating System

4.1 - Accessing and Using the Embedded Scripts

This tuning guide contains scripts attached as plain text (`.txt`) files for easy access that you can open in your preferred text editor. You must have one of the following options installed on your system in order to access these files:

- **Acrobat Reader:** You can download this free tool from Download Adobe Acrobat Reader*.
- **Acrobat DC:** This paid tool is available from Adobe Acrobat*.

To access and use the embedded scripts:

1. Open this PDF file in either Adobe Acrobat DC or Adobe Acrobat Reader.
2. If needed, press [F4] to open the left-hand navigation pane. See Figure 4-1.

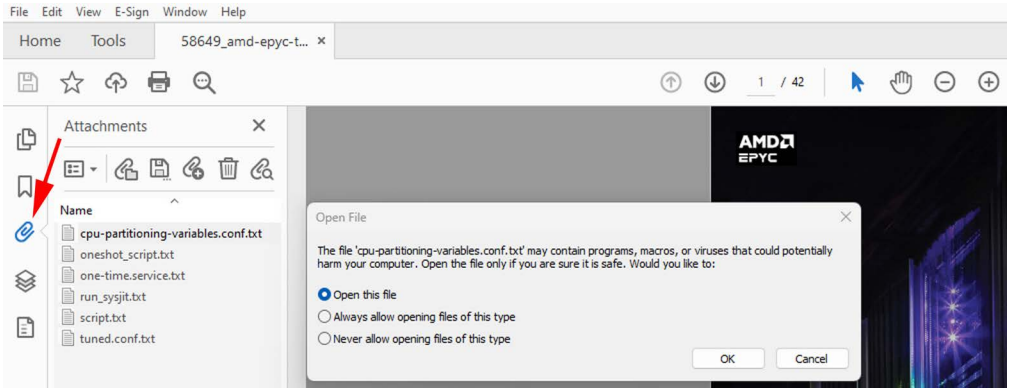

Figure 4-1: Opening the files attached to this PDF

3. Click the **Files** button (paper clip; see red arrow in Figure 4-1) to expand the list of files embedded in this PDF.
4. Double-click the desired file to open the **Open File** window with a security warning, then check the **Open this file** radio button and click **OK**.

   The selected file opens in your default text editor.
5. Save the file to your system as directed in the appropriate section below, being sure to change the extension from `.txt` to the appropriate extension, e.g., `.sh` or `.conf`. The following sections contain the correct extensions.
6. Repeat Steps 4 and 5 for the remaining embedded files.

4.2 - System Topology

4.2.1 - Operating System

Please refer to AMD EPYC™ Processors Minimum Operating System (OS) Versions for the most current information about operating system requirements for all generations of AMD EPYC processors before installing the OS on your system. This will help ensure optimal system performance.

Executing `xsos --os` shows operating system details.

```
# xsos --os
Hostname: gstacn1-sut.amd.com
Distro: [redhat-release] Red Hat Enterprise Linux release 8.6 (Ootpa) [os-release] Red Hat Enterprise Linux
8.6 (Ootpa) 8.6 (Ootpa)
RHN:    (missing)
RHSM:   hostname = subscription.rhsm.redhat.com proxy_hostname =
YUM:    3 enabled plugins: debuginfo-install, product-id, subscription-manager Runlevel: N 3 (default
multi-user)
SELinux: enforcing (default disabled)
Arch:    mach=x86_64 cpu=x86_64 platform=x86_64 Kernel:
Booted kernel: 4.18.0-372.26.1.el8_6.x86_64 GRUB default:
Build version:
Linux version 4.18.0-372.26.1.el8_6.x86_64 (mockbuild@x86-vm-08.build.eng.bos.redhat.com) (gcc version
8.5.0 20210514 (Red Hat 8.5.0-10) (GCC))#1 SMP Sat Aug 27 02:44:20 EDT 2022
```

4.2.2 - lstopo

You can view the physical topology by executing the `lstopo` command after installing the operating system.

If you are using RHEL 8, then use the following repositories to install `lstopo`, which is a part of the `hwloc` package. You may also install the `hwloc-gui` package to get the option to format the lstopo output in multiple graphic formats. Subscriptions are required for repositories in RHEL 8.x. The required subscriptions for `lstopo` are:

```
# subscription-manager repos --enable=rhel-8-for-x86_64-supplementary-rpms
# subscription-manager repos --enable=rhel-8-for-x86_64-baseos-source-rpms
# subscription-manager repos --enable=rhel-8-for-x86_64-appstream-source-rpms
# yum install hwloc hwloc-gui -y
```

The `lstopo` tool provides multiple output formats. Figure 4-2 on the next page shows an example of graphical output generated by the following command line for a 32-core 4th Gen AMD EPYC 9374F.

```
# lstopo --physical --output-format png AMD_EPYC_9374F_32-Core_Processor.png
```

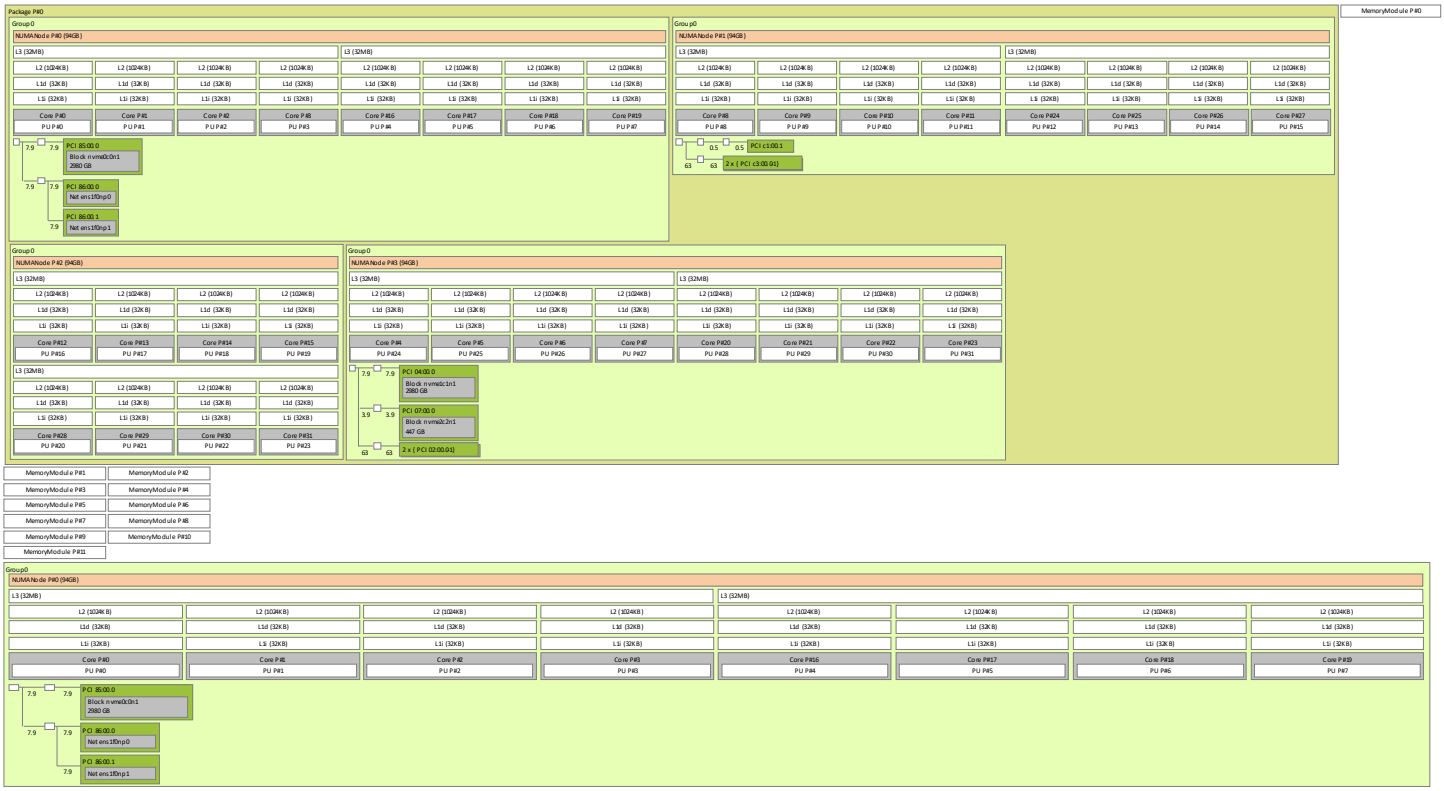

Figure 4-2: Graphical lstopo output for a single-socket system powered by a 32-core AMD EPYC 9374F processor

4.3 - Configuring and Tuning RHEL for Low Latency Performance

This section provides guidelines for configuring a RHEL kernel to allow a single-socket system powered by a 32-core AMD EPYC 9374F processor to achieve optimal performance for attaining the lowest possible latency with fewer jitters for various financial, trading, and matching use cases.

4.3.1 - The /proc filesystem

The `/proc` filesystem interface can expose processor level information, such as the internal kernel data, kernel subsystems, and system devices. The kernel parameters described in this section can help you lower system latency. The Linux `proc` filesystem is a special directory mounted at `/proc` that acts like a window into the inner kernel workings. It doesn't contain actual files, but rather virtual files that provide real-time information about the system.

Some of the important processor and memory information provided by the `/proc` filesystem includes:

- **Processor Information:** The `proc` filesystem contains files under directories such as `/proc/cpuinfo` that reveal details about your CPU, such as the number of cores, clock speed, and cache size. This information helps you understand your processing power and choose applications that can utilize it effectively.

- **Memory Monitoring:** Files such as `/proc/meminfo` provide statistics on memory usage. You can see details such as total memory, available memory, used memory, and memory buffers. This helps you monitor memory consumption and identify potential bottlenecks.
- **Process Management:** Each running process has a directory under `/proc/[PID]`, where `PID` is the process identifier. These directories contain information such as CPU usage, memory usage, open file descriptors, and threads. This allows you to monitor individual processes and identify resource hogs that might be slowing down your system.
- **Dynamic Tuning:** Most proc files are read only, but some allow limited writes, which allows you to adjust certain kernel parameters on the fly, such as memory allocation behavior. (This guide incorporates pertinent low latency tunings to the `[sysctl]` section of the customized `AMDLowLatency` TuneD profile found in "Setup /etc/tuned/AMDLowLatency/tuned.conf" on page 17.

Note: Improper changes can cause system instability. Use caution when applying tunings.

The `/sys` filesystem is similar to the `/proc` filesystem in that it has parameters that can be written to improve deterministic behavior. Make sure that all settings are correct before running Jitter observations on the `isolated_cores` and `no_balance_cores` physical (not virtual) cores of your AMD EPYC processor on single- and dual-socket systems. Please see "Setup /etc/tuned/cpu-partitioning-variables.conf" on page 18 for additional configuration information.

4.4 - Linux TuneD

Linux TuneD is a customizable application that helps tune and optimize your system to reach its best performance. You can use this tool to tune additional parameters. TuneD includes predefined profiles such as `throughput-performance`, `latency-performance`, and `cpu-partitioning` You can use these `tuned-adm` profiles instead of writing directly to the `/proc filesystem` or using `sysctl`. However, if you stop or disable the Tuned daemon, then you will need to restart to set the profile.

Low latency tuning often involves iterating to find the optimal parameter settings. You can make optimized parameters persistent by writing the values into a customized TuneD profile. AMD recommends that you simplify your debugging by keeping all the CPU partitioning, bootloader options, scheduler and `sysctl` parameters in the customized `AMDLowLatency` profile `tuned.conf` file that is managed by Linux TuneD.

4.4.1 - Step 1: Create a Custom AMDLowLatency Profile using Linux TUNED

This section describes how to prepare your server using a customized low latency `AMDLowLatency` TuneD profile that reduces jitter and obtains low latency performance. The Linux `systemctl` "one-time.service" is invoked by the given `/usr/local/bin/oneshot_script.sh` during system startup, which in turn executes various core level assignments, services, RCU, and other settings to optimize system latency. You can copy these steps from the following example and use them with no additional code modifications required.

4.4.2 - Step 2: Configure the Custom AMDLowLatency TuneD Profile

4.4.2.1 - Setup firstTimeConfiguration

1. Execute the following command to create and then enter the directory:

```
# mkdir -p /etc/tuned/AMDLowLatency; cd /etc/tuned/AMDLowLatency
```

2. AMD strongly recommends disabling SMT in the BIOS to avoid any wrong core calculations by the script.

3. The `firstTimeConfiguration` file in that newly created directory chooses the number of cores per processor reserved for `HousekeepingCpus` purposes. The `AMDLowLatency` TuneD profile and its scripts help automatically assign the physical `HousekeepingCpus` cores from the available processor(s) and the processor cores.

   You will need to input a number from 1-4 into this file using your choice of text editor. This number serves as seed data that determines on how many `HousekeepingCpus` per processor need to be allocated by the `AMDLowLatency` TuneD profile via the scripts provided below. Table 4-1 shows the combination of `HousekeepingCpus` and potential physical CPU core assignments based on your chosen `firstTimeConfiguration` value.

   Table 4-1 is based on two AMD EPYC processors with 8 and 32 cores per processor. If you have other AMD EPYC processors with different core counts, then the `HousekeepingCpus` in the rightmost column of Table 4-2 will differ based on your `firstTimeConfiguration` input value.

   Example: Assume you entered 1 in the `firstTimeConfiguration` file for an 8-core AMD EPYC processor with either a single- or dual-socket (1P or 2P) system configuration, meaning that one (1) physical core per processor will be assigned as `HousekeepingCpus`:

   - **Model A with 1 processor (1P):** Physical `HousekeepingCpus`=0
   - **Model I with 2 Processors (2P):** Physical `HousekeepingCpus`=0,8

   Execute the following command to display your desired number:
   ```
   # echo 1 > /etc/tuned/AMDLowLatency/firstTimeConfiguration
   ```

Table 4-1 provides a few examples of the expected physical cores on each processor that will be assigned as `HousekeepingCpus` based on your `/etc/tuned/AMDLowLatency/firstTimeConfiguration` selection shown in the containing user choice as defined in the **firstTime Configuration** column.

Model	Type of SKU (Cores)	# of Sockets	firstTimeConfiguration	HousekeepingCPUs
A	8	1	1	0
B	8	1	2	0,1
C	8	1	3	0,1,2
D	8	1	4	0,1,2,3
E	32	1	1	0
F	32	1	2	0,1
G	32	1	3	0,1,2
H	32	1	4	0,1,2,3
I	8	2	1	0,8
J	8	2	2	0,1,8,9
K	8	2	3	0,1,2,8,9,10
L	8	2	4	0,1,2,3,8,9,10,11
M	32	2	1	0,32
N	32	2	2	0,1,32,33
O	32	2	3	0,1,2,32,33,34
P	32	2	4	0,1,2,3,32,33,34,35

Table 4-1: Housekeeping cores based on firstTimeConfiguration values

4.4.2.2 - Setup /etc/tuned/AMDLowLatency/tuned.conf

Add the `AMDLowLatency` TuneD contents by opening the `tuned.conf.txt` file attached to this PDF and then saving it to `/etc/tuned/AMDLowLatency/tuned.conf` on your system, as described in "Accessing and Using the Embedded Scripts" on page 13.

4.4.2.3 - Setup /etc/tuned/AMDLowLatency/script.sh

`script.sh` helps the AMDLowLatency TuneD profile perform a number of functions. Open the `script.txt` file attached to this PDF and then save it to `/etc/tuned/` on your system as a script (`.sh`) file, as described in "Accessing and Using the Embedded Scripts" on page 13.

Verify that
`/etc/tuned/AMDLowLatency/script.sh` is executable by executing the following command:

```
chmod +x /etc/tuned/AMDLowLatency/script.sh
```

4.4.2.4 - Setup /etc/tuned/cpu-partitioning-variables.conf

1. Execute the following command:
   `# cd /etc/tuned`
2. Edit your existing `/etc/tuned/cpu-partitioning-variables.conf` by adding/modifying the following entries to reflect `isolated`, `no_balance_cores`, and housekeeping CPU(s) via the `/etc/tuned/AMDLowLatency/script.sh` relative function:

Note: If you do not see an existing `/etc/tuned/cpu-partitioning-variables.conf` *and* `/usr/lib/tuned/cpu-partitioning/tuned.conf`*, then you may not have installed the* `cpu-partitioning` *TuneD profile. AMD recommends installing the* `cpu-partitioning` *profile as instructed in the cpu-partitioning-profile section of* Getting started with TuneD*.

3. Modify the `cpu-partitioning` TuneD profile by opening the `cpu-partitioning-variables.conf.txt` file attached to this PDF and then saving it to `/etc/tuned/` on your system as a `.conf` file, as described in "Accessing and Using the Embedded Scripts" on page 13.
4. Check whether the `cpu-partitioning` profile exists on your system by executing the following two commands:

```
# tuned-adm list
Available profiles:
- accelerator-performance     - Throughput performance based tuning with disabled higher latency STOP
states
- balanced                    - General non-specialized tuned profile
- cpu-partitioning            - Optimize for CPU partitioning
- desktop                     - Optimize for the desktop use-case
- hpc-compute                 - Optimize for HPC compute workloads
- intel-sst                   - Configure for Intel Speed Select Base Frequency
- jitter                      - Optimize for CPU partitioning for Jitter Reduction
- latency-performance         - Optimize for deterministic performance at the cost of increased power
consumption
- network-latency             - Optimize for deterministic performance at the cost of increased power
consumption, focused on low
                                latency network performance
- network-throughput          - Optimize for streaming network throughput, generally only necessary on
older CPUs or 40G+ networks
- optimize-serial-console     - Optimize for serial console use.
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides excellent performance across a
variety of common server
                                workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
Preset profile: throughput-performance

# tuned-adm profile_info cpu-partitioning
    Profile name:
    cpu-partitioning
    Profile summary:
    Optimize for CPU partitioning
    Profile description:
```

This customized `AMDLowLatency` TuneD profile uses the `cpu-partitioning` profile from `cpu-partitioning-variables.conf`. For example, if you have a single-socket system with a 32-core processor, then you may want to use 31 isolated cores for your workload and 1 housekeeping core for the operating system from the 32 total available physical cores, which results in:

- Using (1-31) isolated cores to observe Jitters using the open-source Sysjitter tool.
- Reserving Core 0 for housekeeping, as described in "Running Sysjitter" on page 25.

4.4.3 - Step 3: Setup sys Values Using a one-time.service via systemctl

Execute the custom tailored `one-time.service` and `oneshot_script.sh` bash scripts during the boot process to force the required parameters/values settings in the Linux environment.

4.4.3.1 - Configure one-time.service

1. Open the `one-time.service.txt` file attached to this PDF and then save it to `/etc/systemd/system/` on your system as a `.service` file, as described in "Accessing and Using the Embedded Scripts" on page 13.
2. Add the contents into `/etc/systemd/system/one-time.service` by executing the following command and then copying/pasting the contents below that command:
   ```
   # SYSTEMD_EDITOR=/bin/vi /usr/bin/systemctl edit --force --full one-time.service
   ```
3. Verify that you have successfully installed the one-time service on your system by executing the following command:
   ```
   # ls -l /etc/systemd/system/one-time.service
   -rw-r--r-- 1 root root 277 Jun  6 16:22 /etc/systemd/system/one-time.service
   ```

4.4.3.2 - Add /usr/local/bin/oneshot_script.sh

Open the `oneshot_script.txt` file attached to this PDF and then save it to `/usr/local/bin/` on your system as a script (`.sh`) file (`/usr/local/bin/oneshot_script.sh`), as described in "Accessing and Using the Embedded Scripts" on page 13.

4.4.3.3 - Apply the AMDLowLatency TuneD Profile

```
# tuned-adm profile
AMDLowLatency Trying to
(re)start tuned...
TuneD (re)started, changes applied.
```

4.4.3.4 - Verify /usr/local/bin/oneshot_script.sh

Verify that `oneshot_script.sh` is an executable, then execute the following commands after successfully completing the procedures described in "Add /usr/local/bin/oneshot_script.sh" on page 19 and "Generate a New GRUB File" on page 20:

```
# dracut -f -v
# chmod +x /usr/local/bin/oneshot_script.sh
# systemctl daemon-reload
# systemctl enable one-time.service
# systemctl list-unit-files | grep one-time.service
one-time.service              enabled
```

This `one-time.service` status shows that the new service is configured and enabled correctly and ready for the system to execute `/usr/local/bin/oneshot_script.sh` during the next system reboot.

Note: You can execute `/usr/local/bin/oneshot_script.sh` *any time after booting, if required.*

4.4.4 - Post Configuration and Setup

4.4.4.1 - Generate a New GRUB File

Generate a new GRand Unified Bootloader (GRUB) file based on the new kernel boot parameter arguments. These examples are derived from RHEL 8.6 and its kernel.

```
# uname -r (Showing the current running RHEL 8.6 Kernel Version)
4.18.0-372.26.1.el8_6.x86_64

# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
Generating grub configuration file ...
Done

# ls -lhtr /boot/efi/EFI/redhat/grub*
-rwx------ 1 root root 1.8M Dec 3 2023 /boot/efi/EFI/redhat/grubx64.efi
-rwx------ 1 root root 1.0K Jul 28 15:37 /boot/efi/EFI/redhat/grubenv
-rwx------ 1 root root 5.4K Jul 28 15:37 /boot/efi/EFI/redhat/grub.cfg

Check the output files of grub2-mkconfig (grubenv and grub.cfg)
# ls -lhtr /boot/vmlinuz-4.18.0-372.26.1.el8_6.x86_64
-rwxr-xr-x. 1 root root 7.8M Jan 14 2023 /boot/vmlinuz-4.18.0-372.26.1.el8_6.x86_64
```

4.4.4.2 - Verify the Default Boot Kernel to RHEL 8.6

For example:

```
# grubby --default-kernel
/boot/vmlinuz-4.18.0-372.26.1.el8_6.x86_64

# grubby --set-default /boot/vmlinuz-4.18.0-372.26.1.el8_6.x86_64
id="16699b9231234d6f83e12e4549b18673-4.18.0-372.26.1.el8_6.x86_64"
The default is /boot/loader/entries/9ea2c5a3e5c34cdaa62c976eb7408d77-4.18.0-372.26.1.el8_6.x86_64.conf
with index 0 and kernel /boot/vmlinuz- 4.18.0-372.26.1.el8_6.x86_64 /boot/vmlinuz-4.18.0-
372.26.1.el8_6.x86_6
```

4.4.4.3 - Get the GRUBBY Information and its Index

The index value displayed by the Linux `grubby --info` command refers to the order or boot priority of a kernel entry in the GRUB configuration. Index 0 represents the default boot entry. It is the kernel that GRUB will load by default if you don't choose a different option during startup. For example:

```
# grubby --info /boot/vmlinuz-4.18.0-372.26.1.el8_6.x86_64

index=0
kernel="/boot/vmlinuz-4.18.0-372.26.1.el8_6.x86_64"
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
$tuned_params"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-4.18.0-372.26.1.el8_6.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-372.26.1.el8_6.x86_64) 8.6 (Ootpa)"
id="9ea2c5a3e5c34cdaa62c976eb7408d77-4.18.0-372.26.1.el8_6.x86_64"
```

1. Perform a TuneD verification:
   `# grep -E 'TUNED_BOOT_CMDLINE=|TUNED_BOOT_INITRD_ADD=' /etc/tuned/bootcmdline`

```
TUNED_BOOT_CMDLINE="skew_tick=1 nohz=on nohz_full=1-7,9-15 rcu_nocbs=1-7,9-15 tuned.non_isolcpus=00000101
nosoftlockup quiet selinux=0 mce=ignore_ce audit=0 nmi_watchdog=0 hugepagesz=2MB hugepages=800
rcu_nocb_poll hugepagesz=1GB hugepages=4 default_hugepagesz=2MB transparent_hugepage=never tsc=reliable
pcie_aspm=off cpuidle.off=1 processor.max_cstate=0 sfc.performance_profile=latency sfc.rss_cpus=1
isolcpus=nohz,domain,managed_irq,1-7,9-15 modprobe.blacklist=mgag200,drm_kms_helper"
TUNED_BOOT_INITRD_ADD="(hd0,gpt2)/tuned-initrd.img"
```

   The preceding output confirms that the section of the AMDLowLatency TuneD profile cmdline_AMDLowLatency_NN boot parameters (where NN-01...05) are now configured successfully with the required low latency boot parameters given in tuned.conf.

2. Execute the following command:
   `# tail /var/log/tuned/tuned.log`

```
2024-06-26 12:17:29,337 INFO     tuned.plugins.plugin_sysctl: reapplying system sysctl
2024-06-26 12:17:29,392 INFO     tuned.plugins.plugin_vm: transparent_hugepage is already set in kernel
boot cmdline, ingoring
value from profile
2024-06-26 12:17:29,394 INFO     tuned.plugins.plugin_systemd: setting 'CPUAffinity' to '0 8' in the '/etc/
systemd/system.conf'
2024-06-26 12:17:29,400 INFO     tuned.plugins.plugin_script: calling script '/usr/lib/tuned/cpu-
partitioning/script.sh' with
arguments '['start']'
2024-06-26 12:17:29,480 INFO     tuned.plugins.plugin_bootloader: generating initrd image from directory '/
tmp/tmp.xDAazDUFYd'
2024-06-26 12:17:29,482 INFO     tuned.plugins.plugin_bootloader: installing initrd image as '/boot/tuned-
initrd.img'
2024-06-26 12:17:29,483 INFO     tuned.plugins.plugin_bootloader: removing directory '/tmp/tmp.xDAazDUFYd'
2024-06-26 12:17:29,516 INFO     tuned.plugins.plugin_bootloader: installing additional boot command line
parameters to grub2
2024-06-26 12:17:29,528 INFO     tuned.daemon.daemon: static tuning from profile 'AMDLowLatency' applied
2024-06-26 12:17:29,535 INFO     tuned.daemon.daemon: terminating TuneD in one-shot mode
```

   The preceding `tuned.log` output confirms that the AMDLowLatency TuneD profile has started and configured successfully with the required low latency boot parameters.

3. Execute the following command to reboot the system:
   `# systemctl reboot`

4. Be aware of the following important points:

   - A GRUB generation bug in some releases of TuneD causes the first token in the `cmdline` directive in TuneD profiles to be consumed/truncated. Work around this by adding the single token option `quiet` as the first token, which has been added into the custom `AMDLowLatency` TuneD profile configuration file.
   - After rebooting, verify that the `/proc/cmdline` kernel boot parameters are incorporated from `/etc/tuned/AMDLowLatency/tuned.conf`. Use this file to ensure that your preferred boot time arguments are persistent.
   - Check `cat /etc/default/grub` and verify that your changes are in effect after the final reboot.

Note: The one-time.service will call `/usr/local/bin/oneshot_script.sh` when the system reboots and comes back online and implements all the required settings as per the service's script.

5. Check the relevant logs for any errors by executing the following commands and reviewing the sample outputs for your reference and debugging. The following examples are from a dual-socket (2P) system powered by 8-core AMD EPYC processors with SMT=OFF:

   - `# cat  /etc/tuned/AMDLowLatency/firstTimeConfiguration`
     This output confirms that you have configured your system with one physical core per processor as the Housekeeping Core. The sample outputs shown below from each command reflect how the `HousekeepingCPUsPerProcessor` variable is being calculated, along with the Isolated Cores (`/sys/devices/system/cpu/isolated`) reflected in the bootloader.

   - `# cat /etc/systemd/system.conf`
     `CPUAffinity=0 8`
     This output confirms that your system has two allocated `housekeeping_cpus` (0 and 8) by the `AMDLowLatency` TuneD profile `include=cpu-partitioning` profile and its derivation from the system's available two processors. These should be the lowest-numbered CPUs of each processor.

   - `# cat /sys/devices/system/cpu/isolated`
     `1-7,9-15`
     This output confirms that your system has 14 allocated `isolated_cores` (1-7,9-15) by the `AMDLowLatency` TuneD profile from the system's available processors. These cores are allocated from the respective sequences of each processor apart from `housekeeping_cpus`.

   - `# cat /proc/sys/kernel/watchdog_cpumask`
     `0,8`
     The preceding `watchdog_cpumask` output confirms that the `[sysctl]` section of the `AMDLowLatency` TuneD profile `kernel.watchdog_cpumask` is now configured successfully with the correct housekeeping cores (0,8).

6. Check the list of available profiles, which should list the new AMDLowLatency TuneD profile.

```
# tuned-adm list
Available profiles:
- AMDLowLatency              - A Customized Low Latency profile AMDLowLatency with modifications along with
cpu-partitioning profile
- accelerator-performance     - Throughput performance based tuning with disabled higher latency STOP
states
- balanced                    - General non-specialized tuned profile
- cpu-partitioning            - Optimize for CPU partitioning
- desktop                     - Optimize for the desktop use-case
- hpc-compute                 - Optimize for HPC compute workloads
- intel-sst                   - Configure for Intel Speed Select Base Frequency
- jitter                      - Optimize for CPU partitioning for Jitter Reduction
- latency-performance         - Optimize for deterministic performance at the cost of increased power
consumption
- network-latency             - Optimize for deterministic performance at the cost of increased power
consumption, focused on low latency network performance
- network-throughput          - Optimize for streaming network throughput, generally only necessary on
older CPUs or 40G+ networks
- optimize-serial-console     - Optimize for serial console use.
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides excellent performance across a
variety of common server workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
It seems that tuned daemon is not running, preset profile is not activated.
Preset profile: AMDLowLatency
```

7. Verify that the AMDLowLatency profile is active.

```
# tuned-adm active
It seems that tuned daemon is not running, preset profile is not activated.
Preset profile: AMDLowLatency
```

8. Wait for the system to completely reboot, then login as root via the "console/terminal" user, and then verify that the user PID(s) are only associated with Cores 0 and 8 for Housekeeping cores.

```
# taskset -cp $$
pid 3709's current affinity list: 0,8
```

4.5 - Verify Tuning Parameter Configuration

4.5.1 - Boot Parameters

After reboot, be sure to also check the running kernel to verify that the required settings are enabled, including the boot parameters options set by the `AMDLowLatency` TuneD profile. This example shows information for a dual-socket 3rd Gen AMD EPYC 72F3 system with SMT=OFF and how the cores are isolated according to the `AMDLowLatency` TuneD profile.

```
# cat /proc/cmdline
BOOT_IMAGE=(hd0,gpt2)/vmlinuz-4.18.0-372.26.1.el8_6.x86_644 root=/dev/mapper/rhel-root ro crashkernel=auto
resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet skew_tick=1 nohz=on
nohz_full=1-7,9-15 rcu_nocbs=1-7,9-15 tuned.non_isolcpus=00000101 nosoftlockup quiet selinux=0
mce=ignore_ce audit=0 nmi_watchdog=0 hugepagesz=2MB hugepages=800 rcu_nocb_poll hugepagesz=1GB hugepages=4
default_hugepagesz=2MB transparent_hugepage=never tsc=reliable pcie_aspm=off cpuidle.off=1
processor.max_cstate=0 sfc.performance_profile=latency sfc.rss_cpus=1 isolcpus=nohz,domain,managed_irq,1-
7,9-15 modprobe.blacklist=mgag200,drm_kms_helper
```

4.5.2 - GRUB File Content Changes

Check the GRUB file contents and look for new changes with the additional tuned parameters in bold text below. The GRUB file shows how the custom `AMDLowLatency` TuneD profile calls the TuneD mechanism with the boot command parameters defined in the `AMDLowLatency` TuneD related configuration files.

```
# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)" GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap
rhgb quiet"
GRUB_DISABLE_RECOVERY="true" GRUB_ENABLE_BLSCFG=true
GRUB_CMDLINE_LINUX_DEFAULT="${GRUB_CMDLINE_LINUX_DEFAULT:+$GRUB_CMDLINE_LINUX_DEFAULT }
\$tuned_params"
GRUB_INITRD_OVERLAY="${GRUB_INITRD_OVERLAY:+$GRUB_INITRD_OVERLAY }\$tuned_initrd"
```

4.5.3 - Checking Files Affected by TuneD AMDLowLatency Profile and initrd Image

Verify that the custom `AMDLowLatency` TuneD profile has successfully loaded by executing the following command and noting the timestamps to verify that `/boot/tuned-initrd.img` lists the affected `AMDLowLatency` profile-related files and their respective timestamps.

```
# tuned-adm profile AMDLowLatency; ls -ltra --time-style=full-iso /etc/tuned/bootcmdline /etc/tuned/
active_profile /etc/tuned/profile_mode /boot/tuned-initrd.img /boot/efi/EFI/redhat/grub.cfg; lsinitrd /
boot/tuned-initrd.img

Trying to (re)start tuned...
TuneD (re)started, changes applied.
-rw-r--r--. 1 root root    7 2024-06-26 13:08:31.805413814 -0700 /etc/tuned/profile_mode
-rw-r--r--  1 root root   14 2024-06-26 13:08:31.805413814 -0700 /etc/tuned/active_profile
-rw-------. 1 root root 2048 2024-06-26 13:08:31.989413826 -0700 /boot/tuned-initrd.img
-rwx------  1 root root 7003 2024-06-26 13:08:32.000000000 -0700 /boot/efi/EFI/redhat/grub.cfg
-rw-r--r--  1 root root 1566 2024-06-26 13:08:32.054413830 -0700 /etc/tuned/bootcmdline
Image: /boot/tuned-initrd.img: 4.0K
========================================================================
Version:

Arguments:
dracut modules:
========================================================================
drwx------   4 root     root            0 Jun 26 13:08 .
drwxr-xr-x   3 root     root            0 Jun 26 13:08 etc
drwxr-xr-x   2 root     root            0 Jun 26 13:08 etc/systemd
-rw-r--r--   1 root     root           16 Jun 26 13:08 etc/systemd/system.conf
drwxr-xr-x   3 root     root            0 Jun 26 13:08 usr
drwxr-xr-x   3 root     root            0 Jun 26 13:08 usr/lib
drwxr-xr-x   3 root     root            0 Jun 26 13:08 usr/lib/dracut
drwxr-xr-x   3 root     root            0 Jun 26 13:08 usr/lib/dracut/hooks
drwxr-xr-x   2 root     root            0 Jun 26 13:08 usr/lib/dracut/hooks/pre-udev
-rwxr-xr-x   1 root     root          478 Jun 26 13:08 usr/lib/dracut/hooks/pre-udev/00-tuned-pre-udev.sh
========================================================================
```

Chapter 5: Sysjitter

The Solarflare sysjitter utility measures the extent to which the system introduces jitter and how that jitter impacts user-level processes. Sysjitter runs a thread on each processor core, measures elapsed time when the thread is de-scheduled from the core and produces summary statistics for each processor core. You can download Sysjitter from https://github.com/Xilinx-CNS/cns-sysjitter*. After downloading, review the Sysjitter README file for instructions on building and running Sysjitter.

Note: Be sure to run Sysjitter when your system is idle.

5.1 - Running Sysjitter

Install Sysjitter, then open the `run_sysjit.txt` file attached to this PDF and save it to `/opt/` on your system as a script (`run_sysjit.sh`) file, as described in "Accessing and Using the Embedded Scripts" on page 13. See "Sample Sysjitter Output - Individual Core Statistics" on page 27 for sample Sysjitter output for a 2P 32-core AMD EPYC 72F3 processor.

While you are saving the `/opt/run_sysjit.sh` file, be sure to change the Sysjitter binary installation location according to your system's installation location. The test environment used for this tuning guide located this file in `/opt/LowLatency_Jitter/AMD/sysjitter-1.4/sysjitter`, per line 35 in `/opt/run_sysjit.sh`.

Next, verify that `opt/run_sysjit.sh` is executable by executing the following command
`# chmod +x /opt/run_sysjit.sh`.

5.1.1 - Preparation and Checkup

1. Make sure that the system being monitored with Sysjitter is healthy and ready for the Jitter observation. If load average values are close to 0.00, then you may proceed to execute the Sysjitter script. This example shows a good load average for executing the Sysjitter script:

```
# uptime
 15:46:00 up 46 days, 4 min,  2 users,  load average: 0.00, 0.01, 0.02

# uptime
15:44:08 up 46 days, 4 min, 2 users, load average: 0.03, 0.07, 0.02
```

   Here is an example of a system that is not ready to run Sysjitter. In this case, wait a few more minutes until the load average is near 0.00 and then try again.

```
# uptime
15:22:08 up 46 days, 4 min, 2 users, load average: 5.03, 4.07, 0.88
```

2. Verify that your system is set with the `AMDLowLatency` TuneD profile and that your terminal/console where you're running Sysjitter is currently set with Housekeeping Core (0) only.

```
# tuned-adm active
It seems that tuned daemon is not running, preset profile is not activated.
Preset profile: AMDLowLatency
```

   - This message shows that the system TuneD profile is set to `AMDLowLatency`.
   - The TuneD daemon is not running because it has been switched off via the startup Bash script. This is normal. `/usr/local/bin/oneshot_script.sh`.

5.2 - Launching Sysjitter

1. Verify that you have all of the following prerequisites before launching Sysjitter:
   - A single Putty terminal opened to execute the following steps.
   - Close any other sessions you may have to the system (including the BMC console terminal) by exiting the terminal and closing the tab.
2. AMD strongly recommends that you manually run `/usr/local/bin/oneshot_script.sh` every time before running the Sysjitter script.
3. Go to the /opt directory, then run the run_sysjit.sh bash script.
   `# ./run_sysjit.sh 100 605`

   This command uses the following arguments:
   - `100` (preferred) tells Sysjitter to ignore interrupts shorter than 100 ns in length.
   - `605` is how many seconds (10 minutes and 5 seconds, in this example) to run the Sysjitter script. Add 5 extra seconds for every run (e.g., 305, 310, etc.).Also, start your Sysjitter investigations from shorter (1 minute `65` seconds, 5 minutes `305` seconds, etc.) periods. This gives you a quick glance at jitter-related issues and interrupts from various services to the isolated cores.

On completion, Sysjitter outputs a CSV file using the current date and time under a new directory. For example:
`/lowlatency/20240626134405PDT/amd-lowlat.20240626134405PDT`

5.3 - Sample Sysjitter Output - Individual Core Statistics

This example shows sample output of the individual core statistics files and the formatted CSV file in the `tab` file:

```
# Sysjitter Monitored Processor #0 Isolated Cores (1-7)
-rw-r--r-- 1 root root 5539 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.01
-rw-r--r-- 1 root root 5689 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.02
-rw-r--r-- 1 root root 5539 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.03
-rw-r--r-- 1 root root 5789 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.04
-rw-r--r-- 1 root root 5589 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.05
-rw-r--r-- 1 root root 5589 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.06
-rw-r--r-- 1 root root 5489 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.07

# Sysjitter Monitored Processor #1 Isolated Cores (9-15)
-rw-r--r-- 1 root root 5889 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.09
-rw-r--r-- 1 root root 5739 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.10
-rw-r--r-- 1 root root 5539 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.11
-rw-r--r-- 1 root root 5589 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.12
-rw-r--r-- 1 root root 5739 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.13
-rw-r--r-- 1 root root 5739 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.14
-rw-r--r-- 1 root root 5489 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.15
```

The Sysjitter-monitored summary results for Processor #0 (1-7) and Processor #1 (9-15) isolated cores are stored in the following files located in the output directory `/lowlatency` with `[ repdir=/lowlatency/${mydate} ]` as per the script (for example: `20240711130504PDT`):

- `-rw-r--r-- 1 root root 1633 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.txt` shows unformatted Sysjitter output.
- `-rw-r--r-- 1 root root 3445 Jun 26 13:25 sysjitter.amd-lowlat.20240626132446PDT.tab` shows Sysjitter output formatted using the column -t command.

Figure 5-1 shows sample Sysjitter output for 65 seconds.

core #:	1	2	3	4	5	6	7	9	10	11	12	13	14	15
threshold(ns):	100	100	100	100	100	100	100	100	100	100	100	100	100	100
cpu_mhz:	3690	3690	3690	3690	3690	3690	3690	3703	3690	3690	3690	3690	3690	3690
runtime(ns):	65053952001	65053952011	65053952011	65053952021	65053952031	65053952011	65053952031	64825569247	65053952483	65053952021	65053952021	65053952011	6.51E+10	6.51E+10
runtime(s):	65.054	65.054	65.054	65.054	65.054	65.054	65.054	64.826	65.054	65.054	65.054	65.054	65.054	65.054
int_nr:	0	1	1	1	1	1	2	1	3	2	1	1	1	35
int_n_per_sec:	0	0.015	0.015	0.015	0.015	0.015	0.031	0.015	0.046	0.031	0.015	0.015	0.015	0.538
int_min(ns):	0	431	441	411	421	431	190	269	190	371	391	391	391	110
int_median(ns):	0	431	441	411	421	431	421	269	220	10789	391	391	391	120
int_mean(ns):	0	431	441	411	421	431	305	269	260	5579	391	391	391	169
int_90(ns):	0	431	441	411	421	431	421	269	371	10789	391	391	391	300
int_99(ns):	0	431	441	411	421	431	421	269	371	10789	391	391	391	401
int_999(ns):	0	431	441	411	421	431	421	269	371	10789	391	391	391	401
int_9999(ns):	0	431	441	411	421	431	421	269	371	10789	391	391	391	401
int_99999(ns):	0	431	441	411	421	431	421	269	371	10789	391	391	391	401
int_max(ns):	0	431	441	411	421	431	421	269	371	10789	391	391	391	401
int_total(ns):	0	431	441	411	421	431	611	269	782	11160	391	391	391	5936
int_total(%):	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTE Inspect **int_n** and **int_total** rows for each cores. Make sure the lowest jitters results are good enough to run your specific low latency workloads on these isolated cores. Also you may could have seen additional jitters during your Sysjitter runs and we recommend you to do your investigations on **SMI, NMI** and **other Interrupts** through any additional background Linux Services may needs to stopped and disabled before you get a clean system.

Figure 5-1: Sample Sysjitter output (65 seconds)

5.4 - Analysis

The `/lowlatency/20240624145109PDT/sysjitter.amd-lowlat.20240624145109PDT.tab` output provided in the previous section shows various statistics related to Interrupts observed by Sysjitter on the 14 `IsolatedCpus` cores (1-7 and 9-15) during the 605-second run. The HFT segment of the Financial Services industry (FSI) places exceptional demand for low-latency computer hardware. Always exercise great care when architecting these solutions to ensure that deployments perform as required. AMD strongly recommends examining the per-core output across multiple output files for each core to see the time-history of jitter events during the Sysjitter execution time window.

Please contact your AMD representative for further assistance with tuning your system settings if you need a deeper low latency performance tuning analysis.

THIS PAGE INTENTIONALLY LEFT BLANK.

Performance Tuning for Low Latency Response on AMD EPYC™ Processor-Based Servers

PID: 58649

Nimisha Raut is a SMTS Software System Design Engineer at AMD.
Sylvester Rajasekaran is a PMTS S & A Engineer at AMD.
Chuck Newman is a Software Performance Engineer at HPE.