Database Tuning on Linux® OS: Reference Guide for AMD EPYC™ 7002 Series Processors

Publication # **56783** Revision: **1.0**

Issue Date: **November 2019**

Author: **Sylvester Rajasekaran**

Advanced Micro Devices

Contents

Revision History

Date	Revision	Description
November, 2019	1.0	Initial public release.

Purpose

This document is intended to provide generic guidelines on database related tuning for Linux OS. Refer to the Operating System Tuning methods given by individual database vendors.

Chapter 1 Introduction

The AMD EPYC™ 7002 Series Processors are built with leading-edge 7nm technology, Zen2 core and microarchitecture. The AMD EPYC™ SoC offers a consistent set of features across 8 to 64 cores, including 128 lanes of PCIe® Gen 4, 8 memory channels and access to up to 4 TB of high-speed memory. AMD EPYC 7002 processors are built with the following specifications:

AMD EPYC™ 7002 Series	
Process technology	7nm
Max Processor speed	3.4GHz
Max number of cores	64
Max memory speed	3200MHz
Max memory capacity	4TB
Peripheral Component Interconnect	128 lanes (max) PCIeGen4

Chapter 2 Microarchitecture Overview

2.1 Overview

Processor cores, memory controllers, I/O controllers, and security are incorporated into a Multi-Chip Module (MCM) of the AMD EPYC™ 7002 Series Processors.

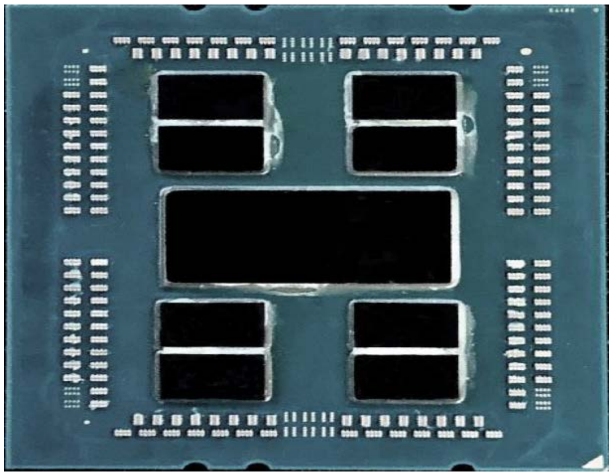

Figure 1 EPYC 7002 Configuration with 8 Core Complex Dies (CCDs) and central I/O Die (IOD)

2.2 Zen 2 core

The EPYC 7002 series processor is based the new Zen2 processor core, that includes an L1 write-back cache. Each core can support Simultaneous Multi-threading (SMT), allowing 2 execution threads to execute simultaneously per core. Each core includes a private 512KB L2 cache.

2.3 Core Complex Die (CCD) and Core-Complex (CCX)

Up to four Zen2 cores share a 16MB (last level) L3 cache. While the two L3 Caches are on the same chiplet, they are separate. The 4 cores and their associated caches are referred to as a Core-Complex (CCX). Each Core Complex Die (CCD) contains 2 CCXs

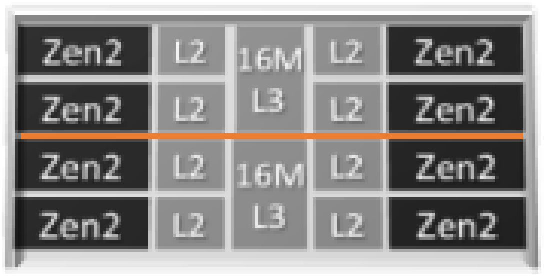

Figure 2 Two Core Complexes (CCXs) on a Core Complex Die (CCD)

Two CCDs may be abstracted as a quadrant. The CCDs connect to memory, I/O, and each other through the I/O Die (IOD). This die supports dual DDR4 memory channels.

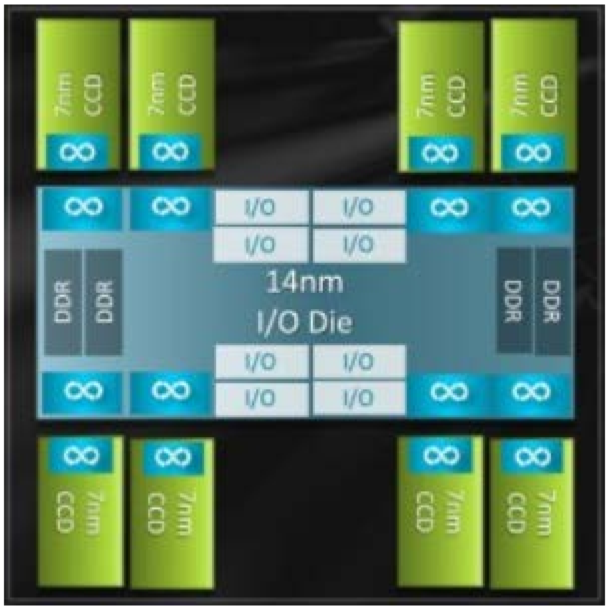

Figure 3 Single socket EPYC 7002 Processor internal connection between CCDs and Memory through memory IOD

2.4 Memory and I/O

Each EPYC 7002 processor supports 8 memory channels. Each memory channel supports up to 2 DIMMs. Based upon BIOS settings these channels can be interleaved across a quadrant (2-way), all the way through 16-channel interleave, that is, across all memory channels of a 2-socket system. The system can have access to a maximum of 4TB of DDR4 memory at 3200MHz per processor.

The PCI subsystem provides up to 128 lanes of high speed I/O.

While all memory and I/O connect to the single I/O Die, they can be abstracted into separate quadrants each with 2 DIMM channels and 32 I/O lanes.

Two EPYC 7002 SoCs are interconnected via Socket to Socket Global Memory Interconnect (xGMI) links, referred to as the Infinity Fabric.

2.5 NUMA

The EPYC 7002 Series processors use a Non-Uniform Memory Access (NUMA) Micro-architecture. The four logical quadrants in an AMD EPYC 7002 processor (as described in *Core Complex Die (CCD) and Core-Complex (CCX)*) allow the processor to be partitioned into different NUMA domains. These domains are designated as NUMA per socket (NPS).

2.5.1 NPS1

The processor is a single NUMA domain, i.e. all the cores on the processor, all memory connected to it and all PCIe devices connected to the processor are in one NUMA domain. Memory is interleaved across the eight memory channels.

2.5.2 NPS2

The processor is partitioned into two NUMA domains. Half the cores and half the memory channels connected to the processor are grouped together into one NUMA domain. Memory is interleaved across the four memory channels in each NUMA domain.

2.5.3 NPS4

The processor is partitioned into four NUMA domains. Each logical quadrant of the processor is a NUMA domain. Memory is interleaved across the two memory channels in each quadrant. PCIe devices will be local to one of four NUMA domains on the processor depending on the quadrant of the IO die that has the PCIe root for that device.

2.5.4 L3 Cache as NUMA Domain

Each L3 Cache (as explained in *Core Complex Die (CCD) and Core-Complex (CCX)*) is exposed as a NUMA node. On a dual processor system, with upto 16 L3 Caches per processor, this setting will expose 32 NUMA domains.

Using BIOS settings, each server can be configured as NPS1, NPS2 or NPS4, with an additional option to configure L3 cache as NUMA nodes.

AMD EPYC 7002 models are available in different core counts per processor and not all of them can support all NPS settings. See *https://developer.amd.com/wp-content/resources/56338_1.00_pub.pdf* for details on NUMA architecture and settings.

Chapter 3 Database Tuning on Linux OS

3.1 Overview

Operating Systems require tuning to increase the efficiency of database platforms and to minimize unnecessary slowdown on the overall system performance and the performance of the applications stack that rely on the database platform. To make an effective tuning for your specific system you must have a baseline of your current system including your observations/readings. This understanding of system requirements helps to tune the systems in a slow and steady pace until you reach the best golden ratio of parameters for your system.

3.1.1 Linux OS Tools

Linux provides multiple tools such as sar, iostat, vmstat, dstat, fio, strace, numactl, numastat etc. that help with observing metrics to identify bottlenecks and assess improvements on Processors, Network, Storage etc.

3.2 Memory and NUMA Settings

AMD EPYC Processors bring up to 8 Memory Channels for memory intensive Database and Applications and respective Memory allocations (Database Buffer Cache) for higher scalability and superior performance.

In AMD EPYC processors the memory is divided into multiple NUMA nodes, it could be NPS 1/2/4 where you could modify in the BIOS settings. By default, the system allocates memory on the same node as a thread and scheduled to run the application. The system uses the allocated memory region locally to the DRAM of the associated NUMA node. In a two NUMA node system, 50% of the memory is allocated to each NUMA node.

Using NUMA policies through NUMACTL command will help to run an application in a way you want to execute and make affinity to the respective cpunodes/cores/memory. For the database platform you could force the NUMA through Database specific NUMA settings. This will help you to adjust the buffer cache according to your database requirements.

Making more DRAM available to the database enables the database to have a larger cache to fit your active dataset into memory. This reduces the I/O and boosts the database performance.

3.3 I/O Schedulers and Elevators

SSD / NVMe drives give better performances for different multi-threaded, I/O intensive applications which are constantly doing the transactions. To ensure better performance on the SSDs, you must make sure that your I/O schedulers / Elevators are kept in correct settings based on your I/O test. See tools recommendations for the AMD EPYC processors on *Phoronix.com* to identify the right scheduler for your system.

Example

- Solid State Drives (SSD) with SATA / SAS
  - Much more IOPS than traditional HDD
  - Low latency
  - To host your Operating System or other multi-threaded applications on **SSD**, use **"deadline" / "noop"** to start.

*cat /sys/class/block/sda/queue/schedulernoop [**deadline**] cfq*

- NVMe drives connected with PCIe
  - More IOPS than SSD
  - Lower latency
  - To host your I/O intensive Database Platform and/or similar multi-threaded applications on **NVMe**, use "**deadline**" / "**none**" to start.

cat /sys/class/block/nvme0n1/queue/scheduler
*[**none**] mq-deadline kyber*

3.4 Transparent HugePages (THP)

Based on the guidelines provided by *https://www.kernel.org/,* performance critical computing applications dealing with large memory working sets are already running on top of *libhugetlbfs* and in turn *hugetlbfs*. Transparent HugePages support is an alternative means of using hugepages for the backing of virtual memory with hugepages that supports the automatic promotion and demotion of page sizes and without the shortcomings of *hugetlbfs*. Databases like PostgreSQL have a choice of using THP to improve the systems performance.

The first factor takes a single page fault for each 2M virtual region touched by user and reduces the enter/exit kernel frequency by 512 times factor. This only matters the first time the memory is accessed for the lifetime of a memory mapping.

The second factor will affect all subsequent accesses to the memory during the entire application run time. This includes the following two components:

- TLB miss will run faster (especially with virtualization using nested page tables but almost always also on bare metal without virtualization)
- A single TLB entry maps to larger amount of virtual memory and reduces the number of TLB misses.

When *transparent hugepages* are enabled you can see the following results

```
cat /sys/kernel/mm/transparent_hugepage/enabled
[always] madvise never
```

Note: Transparent HugePages (THP) reduces the complexity of using huge pages. As the goal of THP is improving performance, its developers (both from the community and Red Hat) have tested and optimized THP across a wide range of systems, configurations, applications, and workloads. This allows the default settings of THP to improve the performance of most system configurations. However, THP is not recommended for all database workloads. Make sure to check the individual database administration guide for better support of THP.

HugePages are in *2 MB* size instead of the default 4K pages in Linux x86_64, which allows databases to use their shared buffers allocations on these pre-allocated memory segments.

vm.nr_hugepages=***NNNNN*** (Depends on your Database Shared Buffers / SGA requirement)

3.4.1 *Formula*

If you have 256 GB DRAM on your system and if you'd like to allocate, for example, **150 GB** for your Shared Buffers / SGA, then do the following in the OS Settings,

Requirements at ***/etc/sysctl.conf***

- Edit this file and add the below values with a good ceiling (90000 * 2 MB = 180,000 MB = **175 GB**)
  vm.nr_hugepages = 90000
- After completing, run the below command to verify and validate that the values are accepted by Linux
  ```
  # sysctl -p
  # sysctl -a | grep -w vm.nr_hugepages
  vm.nr_hugepages = 90000
  ```
- Also change ***memlock*** values (**KB**) and make sure it's lesser than the total available System's DRAM. This is a locked-in memory space which is highly useful for Databases

Requirements at ***/etc/security/limits.conf***

- Edit memlock settings in the ***/etc/security/limits.conf*** file and add/modify to a value which is reasonable and is not exceeding the total available System's DRAM
  ```
  * soft memlock 10000000
  * hard memlock 10000000
  ```
- You could represent the (*) as the particular userid of the database instead of global users locked-in memory on that system
- After completing, login to the database userid (e.g.) oracle / postgres and type the following command
  ```
  $ ulimit -l
  10000000
  ```

3.5 Socket Buffer Size

The default 16 K Linux socket buffer size is insufficient. Increasing socket buffers will be helpful for any N-tier type applications such as Java for sending and receiving buffers up to 4 MB (***4194304*** bytes) during communication with the Database server.

Requirements at ***/etc/sysctl.conf***

- Edit this file and add the below values with a good socket buffer size
  net.core.rmem_max = 4194304
  net.core.wmem_max = 4194304
- After completing, run the below command to check and validate the values are accepted by Linux
  ```
  # sysctl -p
  ```

```
# sysctl -a | egrep -w "net.core.rmem_max|net.core.wmem_max"
net.core.rmem_max = 4194304
net.core.wmem_max = 4194304
```

3.6 File Systems

File systems provide better user data management. Many types of File systems are available in Linux and each of them bring capabilities based on the media used (HDD / SSD / NVMe etc.). To obtain the best performance, you must select the optimal File system of your choice and follow recommendations provided by the Database Administrator's Guide of each databases.

Most commonly used Database File systems are

- **ext4** - This is the most current version of the native Linux file systems and compatible with the older ext2 and ext3 versions. It supports disk volumes up to 1 exabyte and file sizes up to 16 terabytes. It is the standard choice for Linux file systems.
  - Common **ext4** File system mount options used: *defaults, noatime, _netdev, nofail, nosuid, nobarrier*
- **xfs** - High performance journal file system, great for a system with large files such as a media server.
  - Common **xfs** File system mount options used: *nobarrier*, *noatime*

3.7 RAID

Redundant Array of Independent Disks (RAID) puts multiple drives together to improve on what a single drive can do. Depending on your configuration requirements, RAID can increase your system's performance. RAID (Striping and Mirroring) will be very helpful for I/O bound applications on a Database where multiple files such as data files, index files, temporary files, rollback files, redolog / WAL files can reside. RAID Mirror protects from media failures.

The following are few commonly used RAID:

- RAID0: fast writes, reads, no redundancy
- RAID1: slower writes, fast reads
- RAID5: slow for random writes, fast for sequential writes, fast reads and slow recovery
- RAID10: fast reads and writes

3.8 Processor Settings

3.8.1 Performance

Tuned is a profile-based system tuning tool that uses the udev device manager to monitor connected devices and enable both static and dynamic tuning of system settings.

When *tuned* has an effective profile you can see the following results

tuned-adm active
Current active profile: ***throughput-performance***

3.8.2 Power

The default power scheme is "**ondemand**" and it's better to use performance governor:

Use the following command to verify:

> *cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor*
> ***performance***

If it's not ***performance,*** then use the command below to change

> *echo “**performance**” > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor*

Use tools such as ***cpufrequtils, cpupower*** and ***turbostat*** to monitor and understand performance frequencies.

3.8.3 Network

Network plays an important role in Databases and dependent applications to function properly to deliver the data ingress and egress through a properly configured switch, port, F/W etc. You can do the following for efficient Database performance:

- Use higher bandwidth Ethernet to lower latency such as 10 / 25 / Greater GigE.
- Use Trunking/Bonding for Higher Availability and better Bandwidth.
- Enable ARP to filter ARP flux.
- Set higher MTU 9000 (Jumbo Frames) for Cluster Databases environment will help to reduce per-packet overhead both at the end host and at the network switches.

Thus, expected microscopic benefits of jumbo frames are increased throughput and reduced number of CPU cycles and instructions for packet processing.

Use tools such as ***nmap, tc, sar -n, dropwatch, ss -ntmp* –** to identify network traffic packet drops and issues between the servers on your topology.

3.9 Swappiness

The default swappiness setting on Linux is a value range between 0 and 100. For Database server environment we recommend having "0" to keep the coherency and avoid swapping processes out of memory.

Useful metrics to measure swappiness,

```
cat /proc/sys/vm/swappiness
0
```

if your value is greater than "0" then then run the below command in root as

```
 # echo 0 > /proc/sys/vm/swappiness
```

Chapter 4 Tuning is a Continuous Effort for Continuous Best Performance

Make sure to understand the system and the multiple components and resources. This helps you with understanding the best options for performance in the BIOS settings such as NUMA Per Socket (NPS), Memory Interleaving, Enabling and Disabling SMT etc. When the system is in operation, review the performance metrics from the Database, OS, Filesystems, I/O and Networking Parameters. The results from the metric review will help you to precisely tune your system to achieve its best performance level.