RDBMS Tuning Guide for AMD EPYC™ 7002 Series Processors

Abstract

Tuning for Relational Database Management Systems (RDBMS) is an art form and many things affect performance. This document provides a summary of available tuning parameters that enable efficient configuration for the 2nd Generation AMD EPYC™ processors. Use the guidelines in this document to get started with performance tuning. The recommendations include BIOS settings, operating system tunable parameters, and database specific tunable parameters. Use this document as a guidance for tuning 2nd Gen AMD EPYC™ processors when used in on-premise RDBMS solutions. Make sure to refer to the database specific documentation for comprehensive tuning guidelines.

Publication	56949
Revision	1.0
Issue Date	July 2020

Target Audience

Users looking for information on recommended configuration settings for 2nd Gen AMD EPYC™ Processor based systems to maximize performance with RDBMS solutions. Prior knowledge of database systems and system configuration strategies are required to use this document.

AMD

Contents

Chapter 1 Introduction 3
1.1 2nd Gen AMD EPYC Processors 3
1.2 Performance Tuning 3
Chapter 2 Microarchitecture Overview 1
2.1 Zen 2 core 1
2.2 Core Complex Die (CCD) and Core-Complex (CCX) 1
2.3 Memory and I/O 2
2.4 NUMA 2
2.4.1 NPS1 3
2.4.2 NPS2 3
2.4.3 NPS4 3
2.4.4 L3 Cache as NUMA 3
Chapter 3 General Tuning Guidelines for RDBMS 4
3.1 Guidelines for CPU Selection 4
3.2 BIOS Options 5
3.3 Operating System Configuration 6
3.3.1 Memory Tuning 6
3.3.2 Network Tuning 6
3.3.3 I/O Layer Tuning 7
3.4 Linux Configuration 7
3.4.1 sysctl.conf 7
3.4.2 Linux tuned-adm profile 7
Chapter 4 Recommended Settings for Proprietary Databases 8
4.1 Oracle Database 8
4.1.1 General Non-Workload Specific Settings 8
4.1.2 Data Warehouse 9
4.1.3 Online Transaction Processing 10
4.2 Microsoft SQL Server 12
4.2.1 General Non-Workload Specific Settings 12

4.2.2 Data Warehouse 14
4.2.3 Online Transaction Processing 14
Chapter 5 Open Source Databases 15
5.1 Non-Workload Specific Settings 15
5.1.1 Initial Setup 15
5.2 My SQL 16
5.2.1 Installation 16
5.3 Online Transaction Processing 16
5.4 MariaDB 17
5.4.1 Initial Setup 17
5.4.2 General Non-Workload Specific Settings 17
5.4.3 Installation 17
5.4.4 Online Transaction Processing 17
5.5 PostgreSQL 17
5.5.1 Initial Setup 17
5.5.2 Compile and Install Postgres 18
5.5.3 General Non-Workload Specific Settings 18
5.5.4 Database Settings 18
5.6 Runtime Validation 19
5.6.1 Online Transaction Processing (OLTP) 19
5.6.2 Test Run 20
Chapter 6 References 21

Chapter 1 Introduction

1.1 2nd Gen AMD EPYC Processors

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

1.2 Performance Tuning

Performance tuning is an iterative process and is highly specialized to specific workload environments. This document intends to provide a starting point for the performance tuning of RDBMS solutions on the 2nd Generation AMD EPYC™ processors. These recommendations are based on tests run in AMD labs using HammerDB v3.3 workloads to measure and optimize system performance. Using the TPC-C workload do simulate Online Transaction Processing (OLTP) workloads, and the TPC-H workload to simulate Data Warehousing (DW) workloads

The settings described in the tables are the parameters that maximized performance under the tested workloads. Individual applications will have different results depending upon the balances between various system components.

Chapter 2 Microarchitecture Overview

Processor cores, memory controllers, I/O controllers, and security are incorporated into a System on Chip (SoC) of the AMD EPYC™ 7002 Series Processors.

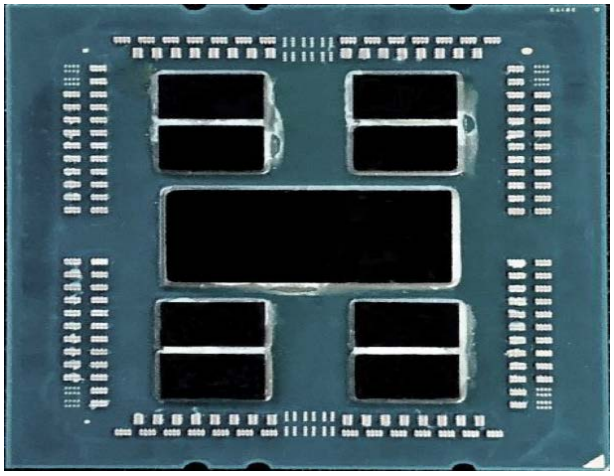

Figure 1 EPYC 7002 Configuration with 8 Core Complex Dies (CCDs) and central I/O Die (IOD)

2.1 Zen 2 core

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

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

Up to four Zen2 cores share a 16MB (last level) L3 cache. These cores and the L3 cache are referred to as a Core Complex (CCX), and there are two CCX's per Core Complex Die (CCD). There are up to 8 CCD's per SoC, or EPYC 7002 Series processor.

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). The IOD has eight DDR4 memory channels, capable of supporting up to two DIMMs per Channel (DPC).

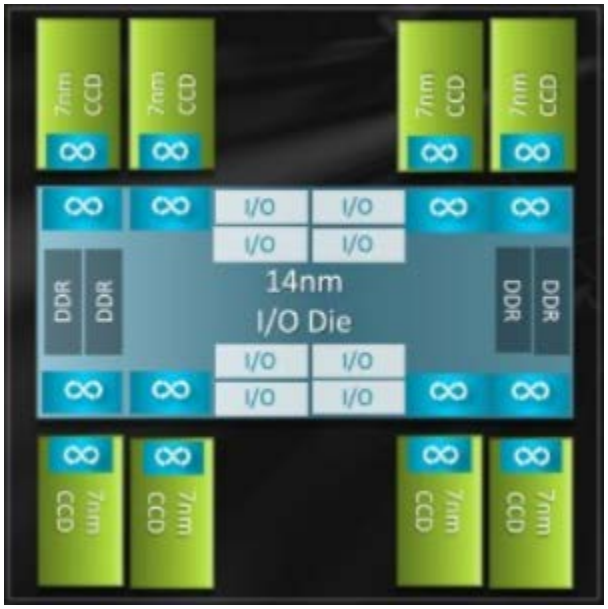

Figure 3 Single socket EPYC 7002 Processor internal connection between CCDs and Memory through memory IOD

2.3 Memory and I/O

Each EPYC 7002 processor supports 8 memory channels, with each memory channel supporting up to 2 DIMMs. Based upon BIOS settings these channels can be interleaved across a quadrant (2-way), 4-way, across an entire socket (8-way), and even across two sockets by interleaving all 16 channels in a dual socket platform. The system can have access to a maximum of 4TB of DDR4 memory at 3200MHz per processor.

The PCIe subsystem provides up to 128 lanes of high speed PCIe Gen4 I/O for single socket platforms and depending on the platform design up to 160 PCIe lanes for dual socket platforms. Platform designers have the option to turn some of the PCIe lanes into SATA ports for attaching SATA devices directly, which would result in fewer available PCIe lanes.

Two EPYC 7002 SoCs are interconnected via Infinity Fabric, which come from 64 of the PCIe phy's on each of the SoC's. This creates a high bandwidth, low latency interconnect for the two processors.

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 or 64 I/O lanes, depending on if it's a single socket or dual socket platform respectively.

2.4 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).

2.4.1 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.

2.4.2 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.

2.4.3 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.

2.4.4 L3 Cache as NUMA

Each L3 Cache, or Last Level Cache (LLC) as explained in Core Complex Die (CCD) and Core-Complex (CCX), is exposed as a NUMA node. On a dual processor system, with up to 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. When LLC as NUMA is enabled, the NPS setting is used to set the memory interleaving mode.

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 General Tuning Guidelines for RDBMS

This section describes the common tuning parameters for proprietary databases such as Oracle Database and open source databases Microsoft SQL Server such as MariaDB, MySQL, and PostgreSQL.

3.1 Guidelines for CPU Selection

Table 1 includes the basic guidelines on CPU selection along with the minimum required memory and I/O bandwidth for the CPUs in general RDBMS use cases. Selecting the right processor depends on the following factors:

Licensing costs: Some RDBMS vendors are licensed by the processor socket others license by the core, in which case lower core count higher frequency processors would be the processor of choice. In other cases where RDMBS vendors license by processor or socket, then higher core count processors will have lower overall license fees for the performance.

Database size, data throughput, and transactional throughput: For memory latency dependent applications, the single socket configurations will perform better. Instead if the application is dependent upon memory bandwidth, 2-socket systems with 8 memory channels per socket will perform better.

RDBMS CPU Recommendations						
System Type	#CPUs	Cores	Processor	Minimum Memory (GB)	DW Minimum I/O Bandwidth (Gbytes/Sec)	OLTP Minimum I/O Bandwidth (Gbytes/Sec)
Small	1	8	EPYC™ 7F32	64	0.78	0.8
	1	16	EPYC™ 7302P	256	1.56	1.6
	1	24	EPYC™ 7402P	256	2.34	2.4
Medium	1	16	EPYC™ 7F52	512	1.56	1.6
	1	24	EPYC™ 7402P	512	2.34	2.4
Performance Optimized	1	16	EPYC™ 7F52	512	1.56	1.6
	1	32	EPYC™ 7502	512	3.13	3.2
	1	64	EPYC™ 7702P	512	6.25	6.4
Memory Optimized	2	2x8	EPYC™ 7F32	1024	1.56	1.6
	2	2x24	EPYC™ 7F72	1024	4.69	4.8
	2	2x32	EPYC™ 7502	1024	6.25	6.4
High Performance	2	2x16	EPYC™ 7F52	2048	1.56	3.2
	2	2x48	EPYC™ 7642	2048	9.38	9.6
	2	2x64	EPYC™ 7742	2048	12.5	12.8

Table 1 Recommended BIOS Settings

3.2 BIOS Options

Table 2 describes the BIOS options that provide most impact on performance for common RDBMS systems. For more details on other Bios settings, see Workload Tuning Guide for AMD EPYC™ 7002 Series Processor Based Servers.

NUMA Nodes per Socket setting is a trade-off between minimizing local memory latency for NUMA-aware or highly parallelizable workloads versus maximizing per-core memory bandwidth for non-NUMA friendly workloads. The greater the number of memory channels interleaved, the higher the memory throughput is for certain memory operations. Increasing the number of NUMA nodes per socket will decrease the memory channels per NUMA node thereby reducing a particular NUMA nodes memory throughput, while decreasing the memory latency. Even with NPS4 setting there are multiple L3 caches in the NUMA domain. The ACPI SRAT L3 Cache as NUMA Domain setting will further split the NUMA domains such that each L3 cache has its own domain. When this is set to 'enable' the NUMA Nodes per Socket setting will still be used to determine the memory interleaving granularity.

General RDMBS Bios Settings		
Name	**Value**	**Description**
TSME	Disabled	Transparent Secure Memory Encryption (TSME) provides hardware memory encryption of all data stored on system DIMMs. The impact of this encryption is 5-7ns of memory latency.
Determinism Control	Enabled	Enables the Determinism Slider control.
Determinism Slider	Power	Ensure maximum performance levels for each CPU by throttling CPUs only when they reach the cTDP.
cTDP Control	Manual	Enable setting customized configurable TDP.
cTDP	OPN Max	Set configurable TDP in Watts.
Package Power Limit Control	Manual	Enable setting customized Package Power Limit (PPL)
Package Power Limit	OPN Max	Set value for PPL in Watts.
SMT Control	Auto	Enables Symmetric Multithreading (SMT), allowing for two hardware threads per core.
NUMA Node per Socket (NPS)	Varies by RDMS and workload	Determines the number of NUMA nodes to split the memory channels between. Higher number indicates fewer memory channels per NUMA node thereby lowering memory throughput for a particular NUMA node, but also lowering memory latency.
ACPI SRAT L3 Cache as NUMA Domain	Varies by RDMS and workload	Dictates whether to report each L3 cache as a NUMA domain to the OS. Allowing processes which use same data to be scheduled on the set of CPUs that share an L3 cache increasing chance of L3 Cache hit.

Table 2 Recommended BIOS Settings for most RDBMS Applications

3.3 Operating System Configuration

AMD recommends using the latest version of the Operating Systems available. The following table shows the minimum required versions of Operating System for using 2nd Generation EPYC™ Processors.

Minimum Operating System Releases	
Operating System	**Release Information**
RHEL/Centos 7	7.6.6
RHEL/Centos 8	8.0.2
SLES 12	SP4
SLES 15	SP1
Ubuntu	18.04.3
Oracle UEK	5.1
Windows Server 2019	1809

Table 3 Minimum Operating System Releases

3.3.1 Memory Tuning

RDBMS systems are generally large consumers of memory. EPYC 7002 Series Processors are designed with 8 memory channels per CPU socket, that increases the total addressable memory per socket and the memory throughput per socket. Enabling large pages in the operating system can improve system performance by reducing the amount of system resources to access page table entries. The size of the large pages varies from platform to platform. You can enable large pages on Linux based systems in one of the following two ways:

- Explicitly set the vm.nr_hugepages parameter
- Implicitly use transparent huge pages

SQL Server on Linux prefers the transparent huge pages, while most of the other vendors prefer explicitly defined huge pages. To enable large pages on Windows Server operating systems, you need to modify the group policy for the database user to enable the Lock Pages in Memory setting.

3.3.2 Network Tuning

Tuning and configuring the network of an RDBMS system is most critical for getting information in and out of the RDBMS quickly. Tuning also ensures that the Network does not overtake all system resources. In smaller systems that run transactions and queries directly on the local server this will be a less critical component in the tuning process. Review the Linux® Network Tuning Guide for AMD EPYCTM 7002 Series Processor Based Servers or Windows® Network Tuning Guide for AMD EPYCTM 7002 Series Processor Based Servers on network tuning for your specific operating system.

3.3.3 I/O Layer Tuning

The EPYC 7002 series processors support PCIe connections with the 4th generation specification which doubles the I/O bandwidth from PCIe 3.0. All the normal RDMBS I/O layer tuning rules will still apply. Such as, separating the Log devices from data devices, split the I/O across the remaining devices, controllers, PCIe busses, etc.

XFS Filesystem Mount Options	
Name	**Description**
noatime	This option disables updating the metadata associated with files in the filesystem with an updated access time. This tracking is unnecessary since databases maintain their own accesses in their logs, so tracking this at the file system is just extra overhead.
nobarrier	Disables the write barrier for the filesystem. Using a write barrier degrades I/O performance by requiring data to be flushed more frequently.

Table 4 XFS File System Mount Options

3.4 Linux Configuration

3.4.1 sysctl.conf

Sysctl.conf settings		
Name	**Value**	**Description**
kernel.numa_balancing	0	Disable Linux kernel auto NUMA balancing
vm.swappiness	1	

Table 5 General sysctl.conf settings

3.4.2 Linux tuned-adm profile

In general, the tuned-adm profile 'throughput-performance' has generated best results. This profile will setup the overall I/O and memory throughput of the system. It does so by configuring the cpu governor, kernel scheduler granularity, disk read ahead, swappiness behavior, and dirty cache write back settings. See the file '/usr/lib/tuned/throughput-performance/tuned.conf' for specific settings of this profile.

Chapter 4 Recommended Settings for Proprietary Databases

This chapter provides recommended settings for the following proprietary databases:

- Oracle database
- Microsoft SQL Server

4.1 Oracle Database

Oracle RDBMS is highly tunable. With hundreds of configuration parameters, combined with OS and hardware settings, bringing Oracle RDBMS to the optimal performing state can be challenging. Before beginning to Oracle RDBMS tuning, consider the following recommendations.

4.1.1 General Non-Workload Specific Settings

Default BIOS settings are not necessarily optimized for Oracle RDBMS performance. Refer to Table 2 Recommended BIOS Settings for most RDBMS Applications for recommended settings.

Oracle Bios Settings		
Name	**Value**	**Description**
NUMA Node per Socket	1	Interleave memory access across all eight channels in each socket, report one NUMA node per socket (Unless L3 Cache as NUMA is enabled).
ACPI SRAT L3 Cache as NUMA Domain	Disabled	Report each L3 cache as a NUMA domain to the OS.

Table 6 Oracle BIOS Settings

- Installation
  - Check and follow Oracle's recommended requirements for specific OS.
  - Consider using Oracle-provided pre-installation package, e.g. 'oracle-database-preinstall-19c' package or similar, to meet all prerequisites and dependencies.
- Monitoring
  - Oracle Monitoring tools, like Automatic Workload Repository (AWR) and Automatic Database Diagnostic Monitor (ADDM), give great insights into instance operational status.
  - It is important also to have a full compilation of operating system, database, and application statistics from your systems.
- Memory
  - Consider using HugePages where applicable. For more details, see https://docs.oracle.com/en/database/oracle/oracle-database/19/unxar/administering-oracle-database-on-linux.html#GUID-CC72CEDC-58AA-4065-AC7D-FD4735E14416

- I/O
    - Understand the workload profile and configure the corresponding storage options to match the IOPS demand (For example: SSD and/or NVME usage).
    - Maintain good data and log separation practice.
    - If using filesystem, Linux FS mounting options like 'nologging' and 'noatime', can provide IO performance benefits from turning off logging and journaling, respectively. Consider using asynchronous and direct I/O where possible. This can be implemented either in sqlmgr shell or in init.ora:
        - SQL> ALTER SYSTEM SET FILESYSTEMIO_OPTIONS=SETALL SCOPE=SPFILE;
        - Add to init.ora:
            - *.filesystemio_options='setall'

4.1.2 Data Warehouse

Most Oracle data warehouse instances are I/O intensive and involve very large datasets, with terabytes of information to aggregate and summarize. Data warehouse systems demand tremendous CPU and RAM resources, as Oracle RDBMS supports parallel features for table summarization, aggregation, DBA maintenance (table reorganization), and parallel data manipulation.

Consider the following recommendations when tuning Oracle database for DW:

- Make sure table/schema statistics are up to date:
    - Check Table for Last Analyze:
        - SELECT owner, table_name, last_analyzed, global_stats from dba_tables;
    - If needed, re-Analyze table and index using DBMS_STATS package: e.g.
        - SQL> exec DBMS_STATS.GATHER_TABLE_STATS (ownname => 'SSB' , tabname => 'CUSTOMER',cascade => true, degree =>64);
        - SQL> exec DBMS_STATS.GATHER_TABLE_STATS (ownname => 'SSB' , tabname => 'LINEORDER',cascade => true, degree =>64);
- Enough memory for large dataset:as when Oracle performs a parallel full-table scan, the database blocks are read directly into the program global area (PGA). Having a properly sized PGA is essential.
    - E.g. sample setting from a test workload, 1500 scale-factor Star Schema Benchmark
        - *.pga_aggregate_target=412815M
- Parallelism: facilitate parallel executions of SQL statement via parallel_(min/max)_servers settings in init.ora, and tables/indexes degree of parallelism (DOP) setting
    - Init.ora settings: (where ##nn usually is certain ratio of available CPUs)
        - *.parallel_max_servers=##nn
        - *.parallel_min_servers= ##nn
    - Change DOP
        - ALTER table <table_name> parallel <new DOP value>;
    - Check if the query is parallelized: use EXPLAIN PLAN
        - E.g. PX COORDINATOR indicated pararallel query was in effect
            - | 2 | LOAD AS SELECT (CURSOR DURATION MEMORY) | SYS_TEMP_0FD9D67B7_26EDE43F | | | | | | | |

- | 3 | PX COORDINATOR | | | | | | |
- | 4 | PX SEND QC (RANDOM) | :TQ10001
- | 8 | 112 | | 3 (34)| 00:00:01 | Q1,01 | P->S | QC (RAND

- In-Memory: Oracle In-Memory column store (IM column store) is an optional area of the System Global Area (SGA) that stores copies of tables, partitions, and other database objects in a columnar format that is optimized for rapid scans. Storing database objects in memory instead of on disk enables Oracle Database to perform scans, queries, joins, and aggregates much faster.
  - To enable Oracle In-Memory, check if the SGA has sufficient space to store needed tables, objects, etc. and adjust SGA_TAGET accordingly. Then INMEMORY_SIZE is used to secure the 'in-memory' space.
    - E.g. in init.ora
    - *.sga_target=1050G
    - *. INMEMORY_SIZE = 800G;
  - To load tables into memory: alter table <tablename> INMEMORY;
    - E.g. ALTER TABLE lineorder INMEMORY;
  - Check and make sure memlock is sufficient for SGA allocation

sysctl.conf settings		
Name	**Value**	**Description**
vm.nr_hugepages	<up to 90% of system memory>	Allocates specified number of 2MB huge pages.
fs.file-max = 6815744	6815774	Maximum number of file-handles.
kernel.sem	250 32000 100 128	Semaphore settings: Maximum semaphores per array, max semaphores system wide, max operations per semop call, and max number of arrays.
kernel.shmall	1073741824	Maximum number of pages for all shared memory segments
kernel.shmmax	4398046511104	Maximum size in bytes of a shared memory segment

Table 7 Oracle sysctl.conf settings

4.1.3 Online Transaction Processing

Online Transaction Processing (OLTP) applications are high throughput, insert/update intensive systems. The transaction size is generally small affecting single or few rows at a time. Write performance to transaction logs is critical and often a major resource constraint. Solid State Disks (SSDs) are strongly recommended for both data areas and redo logs to provide the I/O capabilities to match the CPU performance. Typical tuning starting points are DB block sizes, correct sizing of the SGA and redo logs.

Blocksize: OLTP application often does better with small database block sizes (4k-8k). Consider db_block_size=8192

System Global Area (SGA): should be sufficiently sized to accommodate database activities yet not too big to take OS resources away and create unnecessarily swapping.

Example, setting SGA target to ½ available memory for a test workload

*.sga_target=750G

SORT_AREA_SIZE: Oracle uses this to allocate memory for sorting data. When a sort cannot be completed in memory, it's to be done on disk via Oracle temporary segments, which is considerably slower. This should be sufficiently sized as well.

Example: setting from a test workload

*. sort_area_size=65536

Data/Log Files: monitor and understand the workload IO profile then adjust accordingly

Example: an IO profile of a TPC-C-like test workload collected from AWR:

	Reads:	Reqs	Data	Writes:	Reqs	Data	Small	Large
Filetype Name	Data	per sec	per sec	Data	per sec	per sec	Read	Read
Log File	0M	0.0	0M	333.1G	1738.2	1.1G		
Data File	3.4G	475.7	11.465M	91.7G	3.1E+04	311.112	300.75us	
Control File	15M	3.1	.05M	5M	1.0	.017M	157.61us	
TOTAL:	3.4G	478.9	11.514M	424.8G	3.3E+04	1.4G	299.82us	

4.2 Microsoft SQL Server

4.2.1 General Non-Workload Specific Settings

- Bios Settings: Besides the settings listed in Table 2 Recommended Bios Settings for most RDBMS Applications, the following settings will help improve performance for MS SQL Server 2019.

General RDMBS Bios Settings		
Name	**Value**	**Description**
NUMA Node per Socket	OLTP: AUTO or 1 DW: 1,2 or 4 See note below	Interleave memory access across all eight channels in each socket, report one NUMA node per socket (Unless L3 Cache as NUMA is enabled).
ACPI SRAT L3 Cache as NUMA Domain	Enabled	Report each L3 cache as a NUMA domain to the OS.

Table 8 SQL Server 2019 Specific BIOS settings

- Windows O/S:

Windows O/S Settings		
Name	**Value**	**Description**
Lock Pages in Memory	<Enable>	Enable SQL Server user to lock pages in memory through the group policy editor.

Table 9 SQL Server 2019 Specific BIOS settings

- sysctl.conf

Sysctl.conf settings		
Name	**Value**	**Description**
kernel.numa_balancing	0	Disable Linux kernel auto NUMA balancing
vm.max_map_count	262144	Increase settings for Virtual Address Space
vm.swappiness	1	

Table 10 SQL Server 2019 General sysctl.conf settings

- Linux Database Settings: mssql.conf file

mssql.conf settings		
Name	**Value**	**Description**
memory.memorylimitmb	85-90% of system memory	Limits the max memory visible to SQL Server.
control.writethough	0	For NVMe and any drives that support force unit access (FUA)
traceflag	3979	Disables WAL (FlushFileBuffers) forced flush feature.
traceflag	834	Use large pages for buffer pool.
traceflag	652	Disable read ahead.

Table 11 SQL Server 2019 mssql.conf entries

- Other Database Settings: These settings are used to maximize the performance of SQL Server.

 Several of these settings alter and disable certain database features that detract from performance. This will impact checkpointing, logging and recoverability of the database. Make sure to review carefully before running these settings in production environments.

Set recovery_mode to simple:

```
use master;
ALTER DATABASE <db_name> set recovery simple;
```

Set target recovery interval to 0.

```
ALTER DATABASE <db_name> set TARGET_RECOVERY_TIME = 0 seconds;
```

Set page verify to 'none'.

```
ALTER DATABASE <db_name> set PAGE_VERIFY None;
```

Turn off autogrowth on db files and tempdb files.

```
ALTER DATABASE <db_name> MODIFY FILE (NAME=<dbfile_name>, FILEGROWTH=0)
```

- sp_configure settings

SQL Server 2019 configuration settings		
Name	**Value**	**Description**
Show advanced options	1	Enables View/Modify advanced sp_configure options
Automatic soft-NUMA disabled	0	Allows SQL Server to use its soft-NUMA.
Lightweight pooling	1	Lightweight threaded pools for user connections
Recovery interval	32767	Set recovery interval to highest allowed (32767 minutes).

Table 12 SQL Server 2019 configuration settings

4.2.2 Data Warehouse

SQL Server data warehouses demand a high consumption of CPU, memory, and I/O resources. SQL Server provides parallel queries to optimize query execution and index operations for computers that have more than one microprocessor (CPU). Because SQL Server can perform a query or index operation in parallel by using several operating system worker threads, the operation can be completed quickly and efficiently. In an environment where only 1 query or operation is running at a time (batched queries/updates etc.), SQL Server can set the max degree of parallelism to twice the number of cores to fully utilize all threads in the system. In other cases where several query threads are running simultaneously, it may be beneficial to reduce the max degree of parallelism such that a single query does not consume all the resources forcing other queries to run single threaded.

Normally, setting NPS to 1 will generate best query throughput performance. However, single socket systems with 32 or more cores have shown good performance with higher NUMA node per socket (NPS) BIOS setting. In those cases, setting this to 2 or 4 may provide a performance boost. In our testing systems with between 32-cores and 48-cores NPS setting to 2, and anything greater than 48-core set to 4. Recommend testing this setting in specific workloads to evaluate its effect on performance.

4.2.3 Online Transaction Processing

Due to the high insert and or update nature of online transactional systems, the write performance to transaction logs is critical and often a major resource constraint. Solid State and NVMe drives are strongly recommended for both data areas and redo logs to provide the I/O capabilities to match the CPU performance. The next level of contention issues is generally around transactional latencies whether they be in memory speed, network latencies or data disk latencies.

Chapter 5 Open Source Databases

Open Source databases such as MySQL, MariaDB, and PostgreSQL are highly tunable. You can achieve optimal performance using a combination of the following settings:

- BIOS settings
- RAM selection depending on the use case
- Tuning configuration parameters specific to the database and operating system type, version, and settings.

Other factors that affect performance include network bandwidth, the type of hardware storage, workload size, and the version of the database. Before running the tests, based on the size of the workload you anticipate running in your system, you should size it to the system with appropriate number of cores and RAM size.

5.1 Non-Workload Specific Settings

5.1.1 Initial Setup

The initial system setup and non-workload related settings are similar for the three Open Source databases – MySQL/MariaDB and PostgreSQL. Initialize the BIOS as described in: Table 2 Recommended BIOS Settings for most RDBMS Applications.

System config selections are typically one of the three configuration types:

- 8-core with 64 GB RAM
- 16-core with 128 GB RAM
- 32-core with 256 GB RAM

Typically, a Linux based OS has been used in the following configurations. RAIDED disk system is recommended with 4 to 6 NVME disks providing at least 20K IOPs (higher IOPs are better) improves the database performance. All the disks can be RAIDED to store data and logs. *mdadm* can be used to RAID the disks. A data directory should be created on the RAIDED volume and this data directory would be the base directory for the database.

Example:

mdadm --create /dev/md0 --level=0 --raid-devices=4 /dev/nvme1n1 /dev/nvme1n2 /dev/nvme1n3 /dev/nvme1n4

Wait for the command to complete

```
mkfs.xfs /dev/md0
UUID=`sudo blkid -s UUID -o value /dev/md0`
echo "UUID=$UUID /nvmeraid xfs discard,defaults,nofail 0 2" >> /etc/fstab
```

Mount the created /dev/md0 onto the data directory

mkdir /nvmeraid

mount -a

5.2 My SQL

5.2.1 Installation

MySQL can be downloaded and installed from: https://dev.mysql.com/downloads/mysql/5.5.html?os=3&version=5. Follow the vendor documentation for installation instructions.

5.3 Online Transaction Processing

The MySQL database configurations are defined in the file: /etc/ my.cnf. Some of the parameters in my.cnf that can be tuned include:

my.cnf Parameter Settings		
Name	**Value**	**Description**
Datadir	<Directory to store database data and log files	This is the directory that is mounted on the raided volume created in the initial setup and is the database directory
innodb_buffer_pool_size	100000M	Based upon the amount of RAM available to the test system, the innodb_buffer_pool_size should be setup to 70%-80% of total RAM dedicated to the server.
Innodb_buffer_pool_instances	64	The number of regions that the InnoDB buffer pool is divided into. Reduces contention as different threads write to cached pages.
max_connections	4000	Maximum number of connections (plus 1 for SUPER account) allowed.
Innodb_log_buffer_size	1G	Enables large transactions to run without a need to write the log to disk before the transactions commit.
Innodb_log_file_size	4G	Controls the checkpoint activity. Larger log files have fewer checkpoints, but also make crash recovery take longer.
Table_open_cache	8000	Number of open tables for all threads.
innodb_flush_method	O_DIRECT	Defines the method used to flush data to InnoDB log and data files.
Innodb_doublewrite	0	Disables the doublewrite buffer, will increase performance.

Table 13 my.cnf parameter settings for MySQL

5.4 MariaDB

MariaDB and MySQL are similar in many aspects, users can find several similarities in installation and configuration.

5.4.1 Initial Setup

The initial setup of MariaDB is similar for all open source databases. See Non-Workload Specific Settings for MySQL/MariaDB/PostgreSQL.

5.4.2 General Non-Workload Specific Settings

Initialize the BIOS with settings described in Table 2 Recommended Bios Settings for most RDBMS Applications.

The non-workload specific settings for MariaDB can be setup as per the details in the section See Non-Workload Specific Settings for MySQL/MariaDB/PostgreSQL.

5.4.3 Installation

MariaDB can be downloaded and installed from:

https://downloads.mariadb.org/mariadb/repositories/#distro=RedHat&distro_release=rhel7-ppc64le--rhel7&mirror=guzel&version=10.2. Follow the vendor documentation for installation instructions.

5.4.4 Online Transaction Processing

Like MySQL, the MariaDB database settings are defined by the /etc/my.cnf file and follow the guidelines mentioned for MySQL.

5.5 PostgreSQL

5.5.1 Initial Setup

Create a software RAID0 on the SSD disks and create a filesystem: [Run after sudo bash]

[Change the device path if needed. Check using 'lsblk']

mdadm --create /dev/md0 --level=0 --raid-devices=4 /dev/nvme1n1 /dev/nvme1n2 /dev/nvme1n3 /dev/nvme1n4

Wait for the command to complete

```
mkfs.xfs /dev/md0
UUID=`sudo blkid -s UUID -o value /dev/md0`
echo "UUID=$UUID /nvmeraid xfs discard,defaults,nofail 0 2" >> /etc/fstab
```

```
# Create postgres user

        adduser postgres --home /nvmeraid
        mount -a

Add user postgres to /etc/sudoers (optional)

        root    ALL=(ALL:ALL) ALL
        postgres ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges #admin ALL=(ALL) ALL

#postgres ALL=(ALL) NOPASSWD:ALL      # Allow members of group sudo to execute any command

#sudo   ALL=(ALL:ALL) NOPASSWD:ALL
```

5.5.2 Compile and Install Postgres

To setup the database, download the sources and compile the sources as 'postgres' user.

To install, run:

#sudo make install

Logged in as 'postgres' user, initialize the database:

export PGHOME=/usr/local/pgsql

export PGDATA=/nvmeraid/pgdata

pg_ctl initdb –D $PGDATA

5.5.3 General Non-Workload Specific Settings

Initialize the BIOS with settings described in the table *General BIOS settings* in the *BIOS Options* section.

5.5.4 Database Settings

The database settings vary depending on the number of cores and memory size of the system. Recommend using huge pages where available by setting the following in sysctl.conf

sysctl.conf settings		
Name	**Value**	**Description**
vm.nr_hugepages	<70-80% of system memory>	Allocates specified number of 2MB huge pages.

Table 14 sysctl.conf settings

limits.conf settings		
Name	**Value**	**Description**
memlock	Large enough to hold the huge pages setting.	Memory the user process can lock into its address space.

Table 15 limits.conf entries

Check the pg_hba.conf file to verify that the settings are such that the server and client can communicate properly.

These are the settings that had the most impact on the performance of the system. Some of these impact certain features of the database, please refer to database documentation before implementing them in a production environment.

postgresql.conf settings		
Name	**Value**	**Description**
shared_buffers	70-80% of system RAM	Memory for database buffers
huge_pages	ON	Enable use of huge pages for database.
wal_level	Replica	Determines how much information is written to the WAL
synchronous_commit	OFF	Commit does not wait for transaction record to be flushed to disk
work_mem	8000M	Maximum amount of memory used by a query operation
temp_buffers	8000M	Maximum amount of memory for each session to be used for temporary table accesses

Table 16 postgresql.conf settings impacting performance

5.6 Runtime Validation

5.6.1 Online Transaction Processing (OLTP)

Described below is a runtime validation for MySQL database using HammerDB. Similar setup and methodology can be used for MariaDB and PostgreSQL. See:
https://www.hammerdb.com/document.html

Install HammerDB on the MySQL server under test (SUT) from:
https://www.hammerdb.com/docs/ch01s05.html

Install MySQL client on the client machine

mkdir /home/mysql

copy libmysqlclient.so.20 to /home/mysql

In bashrc , include the following line:

export LD_LIBRARY_PATH=/home/mysql:$LD_LIBRARY_PATH

Build database: Run HammerDB cli from the installed HammerDB with the following options

dbset db mysql

diset connection mysql_host localhost

diset connection mysql_port 3306

diset tpcc mysql_count_ware 400

diset tpcc mysql_partition true

diset tpcc mysql_dbase test1

diset tpcc mysql_pass amd1234!

diset tpcc mysql_num_vu 128

diset tpcc mysql_storage_engine innodb

print dict

buildschema

5.6.2 Test Run

diset tpcc mysql_driver timed

diset tpcc mysql_rampup 2

diset tpcc mysql_duration 5

vuset logtotemp 1

loadscript

vurun

You can vary the number of virtual users or warehouses to get the optimal performance.

The performance measurement parameter – TPM is displayed on the HammerDB cli screen.

Chapter 6 References

- Database Tuning on Linux® OS: Reference Guide for AMD EPYC™ 7002 Series Processors
- Microsoft Windows® Tuning Guide for AMD EPYC™ 7002 Series Processors
- VMware® vSphere Tuning Guide for AMD EPYC™ 7002 Series Processors
- VMware® vSAN Tuning Guide for AMD EPYC™ 7002 Series Processors