Couchbase Tuning Guide for AMD EPYC™ Processor Based Servers

Publication #	**56477** Revision: **0.7**
Issue Date:	**January 2019**

Advanced Micro Devices

Contents

Chapter 1 Introduction 5
1.1 Challenges Involved in Tuning Couchbase 5
Chapter 2 BIOS Settings 6
Chapter 3 Linux Optimizations 7
3.1 Memory Subsystem 7
3.2 Storage subsystem 8
3.3 Network Subsystem 10
3.4 CPU Configuration 11
3.5 Example configuration files RHEL 7.5 Server 11
Chapter 4 Couchbase Settings 13
Appendix A: References 16

Revision History

Date	Revision	Description
January 2019	0.7	Initial NDA release.

Chapter 1 Introduction

Couchbase Server is an open source, multi-model NoSQL distributed database that incorporates a JSON document base and key-value store. Couchbase implements "memory-first architecture" to provide low latency data management for large-scale to deliver consistent high performance, availability and scalability for enterprise web, mobile, and Internet of Things applications.

Couchbase is straightforward to deploy and manage. Each Couchbase Server node consists of completely identical software, drastically simplifying automation. The entire cluster is managed through a single administrator console that offers single-click cluster expansion and rebalancing. Couchbase Server replicates data across multiple nodes to support failover. It also provides a comprehensive Management UI to visualize, monitor, and manage the individual nodes of the cluster as well as overall cluster status and statistics.

Starting from a few settings on the BIOS level, this tuning guide will provide suggestions as to how to tune various components of the Operating System in such a way that it optimizes Couchbase for best performance and discuss few settings in Couchbase Server to get the best out of it.

1.1 Challenges Involved in Tuning Couchbase

It is important to recognize that out-of-the-box Couchbase is already a very fast system, it buffers a lot of I/O in memory, auto-tunes itself by allocating threads based on number CPUs or the amount of disk space available. Couchbase already has facilities for moving data between memory and disk and letting the Operating System to do those tasks under Couchbase is a bad thing. Likewise, there are several sysctl tunable parameters in Linux kernel at multiple layers, modules, calls and functions for I/O, CPU, Memory, Storage, Networking to make Operating System work nicely towards best performance out of Couchbase Server.

The sizing of Couchbase Server cluster is another big topic, very critical to its overall stability and performance, is not covered in this tuning guide. While there are obviously many variables, the main point is this: You need to evaluate the overall performance and capacity requirements for your workload and dataset, and then divide that into the hardware and resources you have available. Your application wants the majority of reads to come out of the cache, and to have the I/O capacity to handle the writes. There needs to be enough capacity in all areas to support everything the system is doing while maintaining the required level of performance.

Chapter 2 BIOS Settings

One way to maximize a server's performance is to look into the BIOS settings to configure the server towards maximum system performance, as the default BIOS (factory) settings might favor saving power for a balanced performance. For our testing, we used a Dell R6415 which is a single socket 1U server. Not all platform providers will use the same nomenclature for settings, so we will describe the general setting as well as the Dell specific one when appropriate. Most server platform providers have a baseboard management controller (BMC) onboard that allows for remote configuration of the platform. For Dell, this device is called iDRAC. The iDRAC provides both a web interface and command line interface that allows administrators to perform remote management tasks.

From iDRAC web interface > Configuration > BIOS Settings

See "*Memory Population Guidelines for AMD EPYC Processors*" for details on how the EPYC 7xx1 series processors memory channels operate. With channel interleaving the two memory channels on each die will be interleaved, and there will then be 1 NUMA domain per die. This will generate 4 NUMA domains per socket. It is also important to make sure DDR is running at expected speed.

Memory Settings
Memory Interleaving is set to Channel Interleaving

For our setup, the system Memory Speed is at 2400MHz. Each processor core supports up to two logical processors. If this option is set to Enabled, the BIOS displays all the logical processors. This option is enabling Symmetric Multi-Threading (SMT). Prefetch settings are Enabled by default.

Processor Settings
Logical Processor (SMT): Enabled

Hardware Prefetcher: Enabled
Software Prefetcher: Enabled

Having all p-states and c-states enabled will allow for optimum performance. For Dell, we do this by enabling OS DBPM Control. Turbo mode allows processor cores to run faster than their base operating frequency.
→ **System Profile Settings**
System Profile → Custom
CPU Power Management → OS DBPM
Memory Frequency → Maximum Performance
Turbo Boost → Enabled
C States → Enabled

Chapter 3 Linux Optimizations

All data in Couchbase passes through a caching layer, so Couchbase's performance is very sensitive to cache hit ratios. Couchbase's documentation suggests that the cache miss ratio is a key performance monitoring variable. Tuning the Linux operating system's memory and storage subsystems provide an increase in Couchbase performance.

3.1 Memory Subsystem

If sufficient memory is provisioned for Couchbase per Sizing guidelines for the Couchbase cluster, Couchbase takes care of moving data between memory and disk to stay away from running out of memory. Swapping to disk hurts Couchbase performance, so to reduce the likelihood of swapping as much as possible set `vm.swappiness` to 1.

Transparent huge pages is enabled by default in Red Hat Enterprise Linux 7.5, it can cause very long delays for allocating new memory in the system, reshuffling pages in the background into large pages, and hence Couchbase's recommendation to disable it.

Transparent huge pages can be disabled by using the following init script:

```
# cat disable-thp
#!/bin/bash
### BEGIN INIT INFO
# Provides:          disable-thp
# Required-Start:    $local_fs
# Required-Stop:
# X-Start-Before:    couchbase-server
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Disable THP
# Description:       disables Transparent Huge Pages (THP) on boot
### END INIT INFO


case $1 in
start)
  if [ -d /sys/kernel/mm/transparent_hugepage ]; then
    echo 'never' > /sys/kernel/mm/transparent_hugepage/enabled
    echo 'never' > /sys/kernel/mm/transparent_hugepage/defrag
  elif [ -d /sys/kernel/mm/redhat_transparent_hugepage ]; then
    echo 'never' > /sys/kernel/mm/redhat_transparent_hugepage/enabled
    echo 'never' > /sys/kernel/mm/redhat_transparent_hugepage/defrag
  else
    return 0
  fi
;;
```

```
esac

# cp disable-thp /etc/init.d/disable-thp
# chmod 755 /etc/init.d/disable-thp
# service disable-thp start
# chkconfig disable-thp on
```

At present, Couchbase does not implement any NUMA related optimization, and the Couchbase recommendation is to disable NUMA in the BIOS or disable it for Couchbase.

NUMA imbalance across nodes can be corrected by modifying the couchbase-server startup script, prepend the daemon variable with `numactl –interleave all` to enable NUMA interleaving. `numactl --interleave all` informs the OS to round-robin the memory allocations between all NUMA nodes and prevent the imbalance in memory allocation. Also, when the NUMA is enabled, set `vm.zone_reclaim_mode=0`.

```
# yum install numactl
# cat /usr/lib/systemd/system/couchbase-server.service
…
…
[Service]
SyslogIdentifier=couchbase
User=couchbase
Type=simple
WorkingDirectory=/opt/couchbase/var/lib/couchbase
LimitNOFILE=70000
LimitMEMLOCK=infinity
ExecStart=numactl –interleave all /opt/couchbase/bin/couchbase-server -- -
noinput
ExecStop=/opt/couchbase/bin/couchbase-server -k
…
…
```

3.2 Storage subsystem

The guideline for storage is to use higher RPM local storage. SSD is a great choice, however, if SSDs can't be used for everything, take advantage of Couchbase's ability to segregate data storage from index storage and deploy the data on the hard drives and deploy the indexes on the SSD for the best outcome. In general, provisioning separate disks for data and indexes yields better results. Make sure the storage controller is battery backed, to take advantage of some of those filesystem mount options that offer big performance gains.

At the virtual file system layer, ext4 is the default in most Linux systems, xfs is better alternative for Couchbase largely because it is slightly better for append-only workloads.

The filesystem mount option, barrier controls the ordering of writes to disks. For battery-backed storage controllers, it is safe to disable barriers to reap big performance boost from disk I/O.

Filesystem mount options /etc/fstab:
EXT4 : noatime, barrier=o, data=writeback
XFS: noatime, nobarrier

The amount of dirty memory at which the background kernel flusher threads will start writeback into the disk and its frequency (interval) can be tuned with the following settings:

Set the limit for dirty bytes in the page cache

```
sysctl -w vm.dirty_background_bytes=104857600
sysctl -w vm.dirty_bytes=209715200
```

Set the max time for dirty pages

```
sysctl -w vm.dirty_writeback_centisecs=100
sysctl -w vm.dirty_expire_centisecs=200
```

Tune VFS cache reclaim

```
sysctl -w vm.vfs_cache_pressure=50
```

Choosing the correct scheduler algorithms for Disk I/O can provide big performance gains. The 'deadline scheduler' behaves like a FIFO but does basic reordering and merging within the scheduler queue. This guarantees a maximum latency for any given write that's in its queue. 'noop' is another option that is supported but less frequently used.

```
# echo deadline > /sys/block/<dev>/queue/scheduler
```

nr_requests : The I/O request queue is another place where performance can be improved. This is the queue that determines how many objects can be reordered prior to them being flushed to the disk. The longer the queue, the better your ordering of writes and the fewer head movements your spinning disk is going to encounter when it starts writing to the disk.

```
# echo 1024 > /sys/block/<dev>/queue/nr_requests
```

Finally make sure that the I/O is aligned on the filesystem down to the physical devices. Unaligned I/O can result in a compounding number of on-disk writes that any given logical write to your file system incurs and will hurt I/O performance. Alignment of partitions is more critical when using SSD and NVMe drives.

Create partition aligned using 'parted'.

Optimal: Use optimum alignment as given by the disk topology information. This aligns to a multiple of the physical block size in a way that guarantees optimal performance.

```
Example:
fdisk -l /dev/nvme0n1
parted /dev/nvme0n1 mklabel gpt
parted -a optimal /dev/nvme0n1 mkpart primary 0% 50%
parted -a optimal /dev/nvme0n1 mkpart primary 51% 100%
```

3.3 Network Subsystem

First, setting initial buffer sizes for transmit and receive buffers for the network will decrease the amount of time after a reboot that it takes for the network to get back to an optimal state for higher network performance.

Receive Packet Steering (RPS) is used to direct packets to specific CPUs for processing. By default, on a Linux kernel, all the interrupts for the network are going to be handled by CPU 0. This could result in artificially creating bottlenecks in the network because all the packets can't be processed fast enough.

echo F to the RPS CPUs parameter and allows all CPUs to take part in processing these interrupts by spreading the load across the system or it can be configured to spread the load across a set of CPUs in a NUMA node.

```
# cat /sys/class/net/p2p1/device/numa_node 2

# echo f > /sys/class/net/device/queues/rx-queue/rps_cpus
```

set_irq_affinity_cpulist.sh script comes with a Mellanox Driver, and can be used to set CPU affinity to handle IRQs:

```
# tar -xf MLNX_OFED_LINUX-4.1-1.0.2.0-ubuntu16.04-x86_64.tgz
# cd MLNX_OFED_LINUX-4.1-1.0.2.0-ubuntu16.04-x86_64
# ./mlnxofedinstall --hypervisor --force-fw-update --enable-mlnx_tune -force
# /usr/sbin/set_irq_affinity_cpulist.sh
1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61 p2p1

Jumbo frame improves Couchbase performance:
# ip link set p2p1 mtu 9000
```

"*Linux® Network Tuning Guide for AMD EPYC™ Processor Based Systems*" is a guide that can be leveraged for configuring the network for systems in a Couchbase cluster. Following the principles outlined in the guide, make sure to tune the size of the TX and RX rings, change the number of interrupts queues to match the cores on the NUMA node which the NIC is collocated, and pin those interrupts to the correct cpu cores. The iperf utility can be used to stress test the network infrastructure to ensure that it is setup properly. It is recommended to follow the tuning guide especially in the setting for the iommu for the O/S, as this will have significant impact on system performance. This is normally done by setting the iommu to pass-through mode by adding

the kernel parameter, "`iommu=pt`," on the kernel boot line (for RHEL 7.5 this is done by modifying the /etc/default/grub file and running grub2-mkconfig utility).

3.4 CPU Configuration

Ensure the performance governor is set to run all cores. On RHEL 7 view the frequency governors available using `cpupower frequency-info –governor` command. To set the frequency governor use `cpupower frequency-set –governor performance`. Generally, the tuned-adm throughput-performance profile works best for Hadoop workloads, and this will set the governor to performance.

Disable CC6:

```
# cpupower -c all idle-set -d 2
```

3.5 Example configuration files RHEL 7.5 Server

```
/etc/default/grub
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap
rhgb quiet iommu=pt"

/etc/rc.local
cpupower -c all idle-set -d 2
ethtool -G p2p1 rx 4096 tx 4096
/usr/sbin/set_irq_affinity_cpulist.sh
1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61 p2p1
echo "deadline" > /sys/block/${a}/queue/scheduler
echo 1024 > /sys/block/${a}/queue/nr_requests


# cat /etc/sysctl.conf
vm.zone_reclaim_mode=0
vm.swappiness=1
vm.dirty_background_bytes=209715200
vm.dirty_bytes=104857600
vm.dirty_writeback_centisecs=100
vm.dirty_expire_centisecs=200
vm.vfs_cache_pressure=50
vm.overcommit_memory=0
net.ipv4.tcp_sack = 1
net.core.netdev_max_backlog = 25000
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
net.core.rmem_default = 33554431
net.core.wmem_default = 33554432
net.core.optmem_max = 33554432
net.ipv4.tcp_rmem =8192 33554432 2147483647
net.ipv4.tcp_wmem =8192 33554432 2147483647
```

```
net.ipv4.tcp_low_latency=1
net.ipv4.tcp_adv_win_scale=1
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv4.conf.all.arp_filter=1
net.ipv4.tcp_retries2=5
net.ipv6.conf.lo.disable_ipv6 = 1
net.core.somaxconn =  65535
```

Chapter 4 Couchbase Settings

The single most important parameter for tuning Couchbase is the amount of memory allocated to it and it is the best way to prevent disk I/O bottlenecks.

Replication:
Data replication is distributed throughout the Couchbase cluster to prevent a single point of failure. Data replication is configurable on a bucket-level and node-basis. Couchbase supports up to 3 replicas (which means up to 4 copies of data). Please note the difference in the terminology in Couchbase in contrast to Hadoop. In Hadoop, replication set to 3, mean a total of 3 copies, but in Couchbase, it refers only to number of replicas, excluding active data.

Disk Priority:
To allocate disk I/O resources differently between the workloads, any buckets that require higher disk I/O access than others, this sets disk priority to high:

```
# curl -v -X POST -u Administrator:password \
   http://10.1.1.101:8091/pools/default/buckets/usertable -d
'threadsNumber=8'
```

Data Ejection Watermarks:
Couchbase uses as much of the allocated memory as possible for caching data in memory. If there is more data than the allocated memory, Couchbase starts ejecting data from memory so it just exists on disk. To tune at which point the ejection occurs, which is governed by a series of watermarks:

The default high watermark is 85%, to change to 90%:

```
# /opt/couchbase/bin/cbepctl 10.1.1.104:11210 -b usertable -p password\
    set flush_param mem_high_wat 90%
```

Memory Optimized (MOI) vs Standard Global Secondary Indexes (GSI)
Indexes are created to lower query latencies and keeping indexes in memory reduces latencies a significantly! MOI is designed for lower latency and highest throughput needs and MOI requires machines with larger memory to keep the index in RAM. Standard GSI can spill to disk when memory runs out. IO Subsystem performance becomes extremely important for standard GSI to be able to perform well. Unlike standard GSI, high performance IO subsystem is not required for MOI. As MOI runs at in-memory speeds, initial and ongoing indexing times are faster with MOI compared to Standard GSI.

Allocating higher CPU resources for Index service:
Though all Services (data, query, index, search etc) in Couchbase can be run on all nodes, Couchbase recommend that individual workloads run on their own set of nodes. This affords workload isolation and independent scaling. Also, hardware can be apportioned suitably based on the nature of the workload. For example, indexes are generally memory intensive and queries are CPU intensive. Independent scalability for best computational capacity per service is achieved by the architectural nicety that Multi-Dimensional-Scaling(MDS) offers us.

If all services are run on the same server, and to allocate more CPU resources to Index service:

```
# curl -X POST -u  'Administrator:password'
http://10.1.1.101:9102/settings -d
'{"indexer.settings.max_cpu_percent":1600}'
```

Fragmentation and Compaction:
Couchbase uses an append-only file structure and every so often in order to clean up dead bytes it needs to compact the disk data structures. Compaction impact Disk I/O, and to attenuate the impact, setting fragmentation threshold to a higher percentage would mean that compaction runs less frequently.

To specify percentage of disk fragmentation when the bucket compaction is triggered:

```
# couchbase-cli setting-cluster -c 192.168.0.1:8091 -u Administrator -p
password \
    --compaction-db-percentage=70
```

To Enable auto compaction starting at 2:00 AM (during off-peak period)

```
# couchbase-cli setting-cluster -c 192.168.0.1:8091 -u Administrator -p
password \
   --compaction-period-from=2:00
```

REST endpoints available to view settings:

Bucket:

```
# curl http://10.1.1.101:8093/admin/settings -u
'Administrator:password' | jq
```

Index:

```
# curl http://10.1.1.101:9102/settings -u 'Administrator:password' | jq
```

To collect all Couchbase logs as zip file for troubleshooting by Couchbase experts:

```
# /opt/couchbase/bin/cbcollect_info -v  name_of_file.zip
```

Appendix A: References

https://developer.couchbase.com/documentation/server/3.x/admin/UI/ui-monitoring-statistics.html

https://docs.couchbase.com/server/5.5/install/sizing-general.html

https://www.kernel.org/doc/Documentation/sysctl/vm.txt

https://blog.couchbase.com/indexing-best-practices/

https://docs.couchbase.com/server/4.1/install/install-ports.html

https://blog.couchbase.com/faster-indexing-and-query-with-memory-optimized-global-secondary-indexes-gsi-part-ii/

https://www.slideshare.net/Couchbase/tuning-couchbase-server-the-os-and-the-network-for-maximum-performance-couchbase-connect-2015

https://docs.couchbase.com/server/5.5/install/install-production-deployment.html