Reducing a Server's Memory Footprint when OMK applications are installed

> Exercise caution while editing **/usr/local/omk/conf/opCommon.nmis, /usr/local/omk/conf/opCommon.json** or **/etc/mongod.conf**; if a syntax error is induced all OMK applications will cease to function.
>
> Check out How to make configuration changes to opCommon.nmis and other files for some more details on best practices for modifying the configuration file.

Reducing a Server's Memory Footprint

NMIS 8

- In **omk/conf/opCommon.nmis:**
  - **omkd_workers** is set at **10** by default. **Try reducing this number to 2** and then **restart the omkd service**.
    If you THEN find YOU need more omkd_workers, increment this value by one and test again until you get to a suitable value for omkd_workers
  - **/omkd/omkd_max_requests: 100 to 500 (start at 100 and increase from this value if needed)**

NMIS 9

- In **omk/conf/opCommon.json:**
  - **omkd_workers** is set at **10** by default. **Try reducing this number to 2** and then **restart the omkd service**.
    If you THEN find YOU need more omkd_workers, increment this value by one and test again until you get to a suitable value for omkd_workers
  - **/omkd/omkd_max_requests: 100 to 500 (start at 100 and increase from this value if needed)**
- in **nmis9/conf/Config.nmis**:
  - **/system/nmisd_worker_max_cycles: 100**

Consider using zram or zswap to extend available memory, with default settings, provided the server does not have high CPU usage.

Advantages of zram

- An advantage of zram is that it does not need to be backed by an appropriately sized swap space - and this makes it easier to configure and maintain during the lifecycle of the server.
- Another advantage of zram is that it is more likely to function in low memory environments whereas zswap should not be used when the server ha s less than 1GB RAM.

- Another advantage of zram is that I've not encountered a server kernel that is not supporting it (zram), whereas I have encountered this with zswap on occasion.

Disadvantages of zram

- A disadvantage of zram is that the computer cannot hibernate when using zram - which is highly unlikely to be a requirement for a server.
- Another disadvantage of zram is that it requires a few installs initially to get it running, although once configured it is easier to maintain.
- Another disadvantage of zram is that pages will be "swapped in", i.e. decompressed when making a zram configuration change on machines with low memory.
  Instead of `restarting the service`, rebooting might be a better option.
  I prefer to set up an equivalent sized swap space as a fallback for when the zram service is restarted to change its settings.
- A final disadvantage that needs mentioning is that the command `free` reports 20% more RAM than is present when configured to use 20% of memory for zram swap, for example:

```
# We have a server with 8GB RAM.
# Expecting a compression ratio of 3.00, this server has been configured to use 20% of RAM (1.6GB) as
zram swap.
# This zram swap is displayed by the `free` command below as 4.7GB, being the uncompressed zram swap (1.6
GB * compression ratio of 3 = 4.8GB).

# The 1.6GB of RAM used as zram swap has not been deducted from the total Mem 7.8GB as displayed by the
`free` command:
# This server has 11GB of combined RAM + zram swap (7.8GB - 1.6GB + 4.7GB), not 12.5GB combined RAM +
zram swap (7.8GB + 4.7GB);

free -h
              total        used        free      shared  buff/cache   available
Mem:          7.8Gi       1.1Gi       264Mi        32Mi       6.4Gi       6.4Gi
Swap:         4.7Gi        46Mi       1.9Gi
```

- ### zram
  - https://www.kernel.org/doc/html/v5.5/admin-guide/blockdev/zram.html

    - **init.d (recommended)**

```
# install bc using apt or yum
sudo apt install bc; # for Debian derivative OS
sudo yum install bc; # for Redhat derivative OS

# Git clone zram-services to the servers /tmp/zram-services directory:
cd /tmp/
git clone https://github.com/eylles/zram-service.git
# Change directory into the cloned ./zram-service/ directory
cd zram-service/
# Checkout our trusted commit:
git checkout fa033ceebb297e01b3215dd32c9c0aea56a6bb8b
# Make absolutely sure the command `sha512sum zram.sh` returns the following
checksum for zram.sh:
sha512sum zram.sh
6bf8c67834d375419e536c5e5c3070f0a1756f83f5bbcd7cc2d0f77b977f0df72536aef9025b8fc787870
cf89bfa2a8c66317242f6d6bccfdba86fca70e663f6  zram.sh
# Move the zram.sh script to /etc/init.d/zram (notice file at /etc/init.d/zram has
no .sh extension):
sudo mv zram.sh /etc/init.d/zram
# set root as owner and group of /etc/init.d/zram:
sudo chown root:root /etc/init.d/zram

# Cleanup as we are finished with our cloned zram-service
cd /tmp/
rm -rf zram-service

# Create the config file:
sudo echo -e 'ALGORITHM=lzo\n# 20% * expected compression ratio of 3 = 60%
\nRAM_PERCENTAGE=60\nPRIORITY=100'|sudo tee /etc/default/zram-config
# Check the config settings
cat /etc/default/zram-config
ALGORITHM=lzo
RAM_PERCENTAGE=20
PRIORITY=100
```

```
# Enable the zram init.d service:
# EITHER, for Debian derivative OS:
sudo update-rc.d zram defaults
# OR, for Redhat derivative OS:
sudo chkconfig --add zram

# Start the zram service:
sudo service zram start

# Check the zram service status
# - Note that DISKSIZE is the UNCOMPRESSED SIZE
# - At a compression ratio of 3.00 the actual memory used by zram for this device
will be 1.6GB:
sudo zramctl
NAME       ALGORITHM DISKSIZE   DATA  COMPR  TOTAL STREAMS MOUNTPOINT
/dev/zram0 lzo           4.7G 360.9M 109.3M 133.3M       2 [SWAP]

# Lastly, establish the best available compression-algorithm from best to worst:
zstd | lz4 | lzo-rle | lzo
# zstd is only supported by kernel 5.1 and newer:
cat /sys/block/zram0/comp_algorithm
[lzo] lzo-rle lz4 lz4hc 842 zstd
# zstd is the best and available - edit /etc/systemd/zram-generator.conf and set
compression-algorithm=zstd
sudo nano /etc/default/zram-config

# After changing compression-algorithm=zstd in /etc/default/zram-config, restart the
systemd service for device zram0 to update to using zstd:
sudo service zram restart

# and check the change has taken effect:
cat /sys/block/zram0/comp_algorithm
lzo lzo-rle lz4 lz4hc 842 [zstd]
# Confirm the device is in use:
sudo zramctl
NAME       ALGORITHM DISKSIZE   DATA  COMPR  TOTAL STREAMS MOUNTPOINT
/dev/zram0 zstd          4.7G 360.9M 109.3M 133.3M       2 [SWAP]


sudo swapon --show;echo -e '\nzramctl:';sudo zramctl;echo -e '\nmm_stat (
https://www.kernel.org/doc/Documentation/blockdev/zram.txt ):';cat /sys/block/zram0
/mm_stat;echo -e '\ncompression ratio';cat /sys/block/zram0/mm_stat|awk '{print $1
/$2}'
NAME           TYPE      SIZE   USED PRIO
/dev/dm-1      partition 708M     0B   -1
/dev/zram0     partition 4.7G 498.4M  100
/data/swapfile file        4G     0B   -2

zramctl:
NAME       ALGORITHM DISKSIZE   DATA  COMPR  TOTAL STREAMS MOUNTPOINT
/dev/zram0 zstd          4.7G 360.9M 109.3M 133.3M       2 [SWAP]

mm_stat ( https://www.kernel.org/doc/Documentation/blockdev/zram.txt ):
378421248 114624368 139722752 1675169792 660004864    21330   242167

compression ratio
3.3014


# And finally check memory:
free -h
              total        used        free      shared  buff/cache   available
Mem:          7.8Gi       1.1Gi       264Mi        32Mi       6.4Gi       6.4Gi
Swap:         4.7Gi     499.4Mi       1.9Gi

# Check the available swap devices:
sudo swapon --show
NAME       TYPE      SIZE   USED PRIO
/dev/zram0 partition 4.7G     0B  100
```

- **systemd**

```
# as a non-root user install Rust and Cargo:
curl https://sh.rustup.rs -sSf | sh
source $HOME/.cargo/env

# as a non-root user navigate to an apppropriate directory and git clone zram-
generator
cd ~/git_repos/
git clone https://github.com/systemd/zram-generator.git
cd zram-generator/

make build
# disregard the one error at completion - we haven't installed ronn:
/bin/sh: 1: ronn: not found
make: *** [Makefile:41: man] Error 127

sudo make install NOBUILD=true
# again disregard the one error at completion - we haven't installed ronn
install: cannot stat 'man/zram-generator.8': No such file or directory
make: *** [Makefile:65: install] Error 1

# create the config file with zram-size set at 20% of RAM
sudo echo -e '[zram0]\nhost-memory-limit=none\n# 20% * expected compression ratio of
3 = 60%\nzram-size=ram/1.666666667\ncompression-algorithm=lzo'|sudo tee /etc/systemd
/zram-generator.conf
# check the config settings
cat /etc/systemd/zram-generator.conf
[zram0]
host-memory-limit=none
zram-size=ram/5
compression-algorithm=lzo

# Once installed and configured, the generator will be invoked by systemd early at
boot, there is no need to do anything else:
#
# Start the systemd service - it defaults to enabled when installed:
sudo systemctl daemon-reload
sudo systemctl start /dev/zram0
# check service status
sudo systemctl status /dev/zram0

# confirm the device is in use
# - Note that DISKSIZE is the UNCOMPRESSED SIZE
# - At a compression ratio of 3.00 the actual memory used by zram for this device
will be 1.6GB:
sudo zramctl
NAME       ALGORITHM DISKSIZE   DATA  COMPR  TOTAL STREAMS MOUNTPOINT
/dev/zram0 lzo           4.7G 360.9M 109.3M 133.3M       2 [SWAP]

# Lastly, establish the best available compression-algorithm from best to worst:
zstd | lz4 | lzo-rle | lzo
# zstd is only supported by kernel 5.1 and newer:
cat /sys/block/zram0/comp_algorithm
[lzo] lzo-rle lz4 lz4hc 842 zstd
# zstd is the best and available - edit /etc/systemd/zram-generator.conf and set
compression-algorithm=zstd
sudo nano /etc/systemd/zram-generator.conf

# After changing compression-algorithm=zstd in /etc/systemd/zram-generator.conf, to
restart the systemd service for device zram0 to update to using zstd:
sudo systemctl restart systemd-zram-setup@zram0.service

# and check the change has taken effect:
cat /sys/block/zram0/comp_algorithm
lzo lzo-rle lz4 lz4hc 842 [zstd]
# confirm the device is in use:
sudo zramctl
NAME       ALGORITHM DISKSIZE   DATA  COMPR  TOTAL STREAMS MOUNTPOINT
/dev/zram0 zstd          4.7G 360.9M 109.3M 133.3M       2 [SWAP]
```

```
# Calculate the compression ratio for zram0 - it may take a short while to return
the realistic expected compression ratio: sudo swapon --show;echo -e '\nzramctl:';
sudo zramctl;echo -e '\nmm_stat ( https://www.kernel.org/doc/Documentation/blockdev
/zram.txt ):';cat /sys/block/zram0/mm_stat;echo -e '\ncompression ratio';cat /sys
/block/zram0/mm_stat|awk '{print $1/$2}'
NAME           TYPE      SIZE   USED PRIO
/dev/dm-1      partition 708M     0B   -1
/dev/zram0     partition 4.7G 360.9M  100
/data/swapfile file        4G     0B   -2

zramctl:
NAME       ALGORITHM DISKSIZE   DATA  COMPR  TOTAL STREAMS MOUNTPOINT
/dev/zram0 zstd          4.7G 360.9M 109.3M 133.3M       2 [SWAP]

mm_stat ( https://www.kernel.org/doc/Documentation/blockdev/zram.txt ):
378421248 114624368 139722752 1675169792 660004864    21330   242167

compression ratio
3.3014


# And finally check memory:
free -h
              total        used        free      shared  buff/cache   available
Mem:          7.8Gi       1.1Gi       264Mi        32Mi       6.4Gi       6.4Gi
Swap:         4.7Gi   360.9.1Mi       1.9Gi

# Check the available swap devices:
sudo swapon --show
NAME       TYPE      SIZE   USED PRIO
/dev/zram0 partition 4.7G     0B  100
```

- ## zswap
  - **First check whether the linux kernel supports zswap:**

    ```
    grep -i zswap /boot/config-$(uname -r)

    # CONFIG_ZSWAP is not set
    ```

    The output '**CONFIG_ZSWAP is not set**' signals that the kernel does not support zswap.
    The output '**CONFIG_ZSWAP=y**' signals that the kernel does support zswap.

  - **https://www.kernel.org/doc/html/latest/vm/zswap.html** provides that:
    - *Zswap is a lightweight compressed cache for swap pages. It takes pages that are in the process of being swapped out and attempts to compress them*
      into a dynamically allocated RAM-based memory pool. zswap basically trades CPU cycles for potentially reduced swap I/O.
      This trade-off can also result in a significant performance improvement if reads *from the compressed cache are faster than reads from a swap device.*
    - ***Zswap is a new feature as of v3.11 and interacts heavily with memory reclaim. This interaction has not been fully explored on the large set of potential configurations and workloads that exist.***
      For this reason, zswap is a work in progress and should be considered experimental.
    - *Overcommitted guests that share a common I/O resource can dramatically reduce their swap I/O pressure, avoiding heavy handed I/O throttling by the hypervisor.*
      This allows more work to get done with less impact to the guest workload and guests sharing the I/O subsystem.
    - *Users with SSDs as swap devices can extend the life of the device by drastically reducing life-shortening writes.*
  - **Performance Analysis of Compressed Caching Technique**
    - See the CONCLUSION of this paper for insights as to why zswap should not be used on a server with less than 1GB RAM.

  - **For zswap to function correctly it needs swap space equivalent to the maximum uncompressed RAM zswap may contain in its pool**
    - To cater for a zswap compression ratio of 4.0, with the default 20% zswap pool (1/5th),
      one would provision swap space in a server with 10GB RAM installed of 2 GB zswap pool * (4.0-1) = 6.0 GB RAM:
      Server with 10GB RAM + at least 6.0 GB swap space should then be provisioned
    - **One can expect at least a zswap compression ratio of 3.1, so when enabling zswap on a server with 10GB RAM installed, one should provision a minimum of 4.2 GB swap space.**

- **Don't be tempted to increase maximum pool percent from the default setting of 20: this will most likely affect performance adversely.**
- **Command to view zswap info during operation when enabled:**
  - *sudo grep -rRn . /sys/kernel/debug/zswap/*
    may require mounting debugfs :
    sudo mount -t debugfs none /sys/kernel/debug

- **https://www.ibm.com/support/pages/new-linux-zswap-compression-functionality provides that for a server with 10GB RAM and 20% zswap pool size:**

  - *For the x86 runs, the **pool limit was hit** earlier - starting **at the 15.5 GB data point***
  - ***On x86, the average zswap compression ratio was 3.6***

  - Average zswap compression ratio of 3.6 ties in with the 15.5 GB data point a which real disk swap started as follows (as given in **https://www.ibm.com/support/pages/new-linux-zswap-compression-functionality**):
    - 10 GB RAM with 2 GB zswap pool =
      - 8 GB available RAM + 2 GB zswap pool =
        - 8 GB available RAM + (2 GB zswap pool * 3.6 average zswap compression ratio) =
          - 8 GB available RAM + 7.2GB zswap compressed RAM =
            - 15.2 GB RAM

- **Real world example 2021-12-22:**
  - **The following code was placed in /etc/rc.local to use the best zswap options available, as improvements have been made to zswap compression since this article was originally written**

    - **/etc/rc.local**

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# enable zswap:
echo 20 > /sys/module/zswap/parameters/max_pool_percent||:;
echo 1 > /sys/module/zswap/parameters/enabled||:;
# https://linuxreviews.org/Zswap
sh -c 'CMP=/sys/module/zswap/parameters/compressor;echo zstd >"${CMP}"||echo lz4
>"${CMP}"||echo lzo-rle >"${CMP}"||:;' 2>/dev/null
# use the best zpool compressed memory pool allocator available:
sh -c 'ZP=/sys/module/zswap/parameters/zpool;echo z3fold >"${ZP}"||echo zsmalloc
>"${ZP}"||:;' 2>/dev/null
# enable zswap debug info
mount -t debugfs none /sys/kernel/debug 2>/dev/null||:;

exit 0;
```

    - On Debian 9.13 the following zswap parameters were set from the above /etc/rc.local script

```
sudo -i
chmod +x /etc/rc.local
source /etc/rc.local
grep . /sys/module/zswap/parameters/*

/sys/module/zswap/parameters/compressor:lz4
/sys/module/zswap/parameters/enabled:Y
/sys/module/zswap/parameters/max_pool_percent:20
/sys/module/zswap/parameters/zpool:zsmalloc
```

    - When this server was swapping the following zswap stats were achieved:

```
free -h
              total        used        free      shared  buff/cache   available
Mem:           2.0G        1.9G         58M        728K         27M         17M
Swap:          2.7G        1.4G        1.4G

sudo sh -c 'echo;grep . /sys/module/zswap/parameters/*;echo;D=/sys/kernel/debug
/zswap;grep . "${D}/"*;perl -E "say \"\nCompress Ratio: \".$(cat "${D}/stored_pages")
*4096/$(cat "${D}/pool_total_size")" 2>/dev/null'

/sys/module/zswap/parameters/compressor:lz4
/sys/module/zswap/parameters/enabled:Y
/sys/module/zswap/parameters/max_pool_percent:20
/sys/module/zswap/parameters/zpool:zsmalloc

/sys/kernel/debug/zswap/duplicate_entry:0
/sys/kernel/debug/zswap/pool_limit_hit:565978
/sys/kernel/debug/zswap/pool_total_size:287293440
/sys/kernel/debug/zswap/reject_alloc_fail:593
/sys/kernel/debug/zswap/reject_compress_poor:436
/sys/kernel/debug/zswap/reject_kmemcache_fail:0
/sys/kernel/debug/zswap/reject_reclaim_fail:1596
/sys/kernel/debug/zswap/stored_pages:360385
/sys/kernel/debug/zswap/written_back_pages:1145522

Compress Ratio: 5.1370556204920
```

- How to interpret the above stats:
  - 'free -h' command provides that 1.4G of swap was used.
    That stat refers to the uncompressed swap:
    '/sys/kernel/debug/zswap/stored_pages * 4096' = '360385 * 4096' = 1476136960 bytes = 1.37476 GB
  - The 1.37476 GB of swap was compressed to '/sys/kernel/debug/zswap/pool_total_size' = 287293440 bytes = 0.26756 GB of RAM.
  - **Compression Ratio is therefore '1.37476 GB/0.26756 GB' = 5.13706**

    - **Note**: A compression ratio > 5.00 is unusually high, long term:
      - when zpool is **zbud**, the typical c**ompression ratio** will be **1.70**;
      - when zpool is **z3fold**, the typical **compression ratio** will be **2.70**;
      - when zpool is **zsmalloc**, the typical **compression ratio** will be **3.00**,
        as in the following example where swap did not hit the disk,
        even though 9GB memory was allocated when the server had only 7.8GB memory available:

```
free -h
              total        used        free      shared  buff
/cache   available
Mem:           7.8G        7.5G        245M         20K
73M        148M
Swap:          7.7G        1.5G        6.2G

sudo sh -c 'echo;grep . /sys/module/zswap/parameters/*;echo;D=
/sys/kernel/debug/zswap;grep . "${D}/"*;perl -E "say \"
\nCompress Ratio: \".$(cat "${D}/stored_pages")*4096/$(cat "${D}
/pool_total_size")" 2>/dev/null'

/sys/module/zswap/parameters/compressor:lz4
/sys/module/zswap/parameters/enabled:Y
/sys/module/zswap/parameters/max_pool_percent:20
/sys/module/zswap/parameters/zpool:zsmalloc

/sys/kernel/debug/zswap/duplicate_entry:0
/sys/kernel/debug/zswap/pool_limit_hit:0
/sys/kernel/debug/zswap/pool_total_size:387317760
/sys/kernel/debug/zswap/reject_alloc_fail:1507
/sys/kernel/debug/zswap/reject_compress_poor:0
/sys/kernel/debug/zswap/reject_kmemcache_fail:0
/sys/kernel/debug/zswap/reject_reclaim_fail:0
/sys/kernel/debug/zswap/stored_pages:291753
/sys/kernel/debug/zswap/written_back_pages:0

Compress Ratio: 3.08537436548223
```

- With regards to the VM, with Compress Ratio: 5.1370556204920 example further above,
  swapsize was set knowing the compression ratios zswap achieved on this VM.

  To allow zswap to achieve a compression ratio of **5.13706**,
  zswap needs swap space equivalent to the maximum uncompressed RAM zswap may contain in its pool when compressing at compression ratio of **5.13706**.

  Consequently, the minimum swap space this VM should then have is:
  - **'2 GB RAM * /sys/module/zswap/parameters/max_pool_percent * Compression Ratio'**
    = '2 GB RAM * 20% * 5.13706'
    = 2.055 GB swap

- **However, the reason why the swap on this VM was actually set at 2.75 GB is that the command**
  watch free -h
  showed that this VMs' zswap compression ratio was fluctuating between 4.8 and 6.1

  Consequently the decision was made to set
  '2GB RAM * 20% * 6.875'
  = 2.75 GB swap on this VM.

- **Use a swapfile for swap as this provides better flexibility in sizing swap than a swap partition**.
  To resize an existing **/swapfile** to 2.75 GB (2883584 kilobytes):

```
SWAPFILE=/swapfile;
sudo swapoff "${SWAPFILE}";
# count in kilobytes
sudo dd if=/dev/zero of="${SWAPFILE}" bs=1024 count=2883584;
sudo chmod 0600 "${SWAPFILE}";
sudo mkswap "${SWAPFILE}";
sudo swapon "${SWAPFILE}";
sudo swapon --show;
NAME      TYPE SIZE USED PRIO
/swapfile file  12G 512K   -2
```

- To make a new /swapfile persist after a reboot append this entry to **/etc/fstab**:

```
/swapfile swap swap defaults 0 0
```

Setting **cacheSizeGB** appropriately when **MongoDB table type is wiredtiger (the default)**

If your MongoDB installation is using table type wiredtiger, please read
https://docs.mongodb.com/manual/reference/configuration-options/#storage.wiredTiger.engineConfig.cacheSizeGB

- Consider setting **cacheSizeGB** appropriately taking into consideration the memory requirements of other processes running on this box. With default mongod settings, a box with 16GB memory will most probably be currently set to use more than 7GB of memory for cacheSizeGB.

  This may be too much considering the box may be running NMIS, OMK and other applications too.

  - **Check how much memory mongod is using for this cache at /var/log/mongodb /mongod.log: search for cache_size where it is provided in MB, for example:**
    - STORAGE [initandlisten] wiredtiger_open config: create,**cache_size=7414M**
      cacheSizeGB is set in GB, so a cache_size of 7414M is equivalent to the following setting in /etc/mongod.conf:
      - **storage:**
        - **wiredTiger:**
          - **engineConfig:**
            - **cacheSizeGB: 7.24**

  - **Example where 60% was decided as the ratio that should be used to compute MongoDB cacheSizeGB on a VM with 8GB memory**

    - MONGO_USAGE=0.6;echo "$MONGO_USAGE";
    - TOTAL_MEM=$(cat /proc/meminfo| grep -iF "memtotal" | awk '{print $2}');echo "$TOTAL_MEM";
      - returns (in KB): TOTAL_MEM=7673702.4
    - MONGO_MEM=$(echo "(${TOTAL_MEM} / (1024 * 1024)) * ${MONGO_USAGE}" | bc -l);echo "$MONGO_MEM";
      - returns (in GB): MONGO_MEM=4.390927734375
    - MONGO_MEM=$(printf "%.2f" "${MONGO_MEM}");echo "$MONGO_MEM";

    - returns (in GB): MONGO_MEM=4.39
- CACHE_MEM_FINAL=$(echo "(${MONGO_MEM} -1) * 0.5) / 1" | bc -l);echo "$CACHE_MEM_FINAL";
    - returns (in GB): CACHE_MEM_FINAL=1.695
- CACHE_MEM_FINAL=$(printf "%.2f" "${CACHE_MEM_FINAL}");echo "$CACHE_MEM_FINAL";
    - returns (in GB): CACHE_MEM_FINAL=1.70
- https://docs.mongodb.com/manual/reference/configuration-options/#storage-options
  IF CACHE_MEM_FINAL < 0.25 (GB) THEN CACHE_MEM_FINAL = 0.25 (GB)
- echo $CACHE_MEM_FINAL
    - **1.7**
- Set **cacheSizeGB** to **1.7**

Related Topics

- OMK Common Settings to Consider Changing
- Configuration Options for Server Performance Tuning