**[redis.conf配置详细解析](https://www.cnblogs.com/kreo/p/4423362.html)**

[复制代码](javascript:void(0);)

# redis 配置文件示例

# 当你需要为某个配置项指定内存大小的时候，必须要带上单位，

# 通常的格式就是 1k 5gb 4m 等酱紫：

#

# 1k => 1000 bytes

# 1kb => 1024 bytes

# 1m => 1000000 bytes

# 1mb => 1024\*1024 bytes

# 1g => 1000000000 bytes

# 1gb => 1024\*1024\*1024 bytes

#

# 单位是不区分大小写的，你写 1K 5GB 4M 也行

################################## INCLUDES ###################################

# 假如说你有一个可用于所有的 redis server 的标准配置模板，

# 但针对某些 server 又需要一些个性化的设置，

# 你可以使用 include 来包含一些其他的配置文件，这对你来说是非常有用的。

#

# 但是要注意哦，include 是不能被 config rewrite 命令改写的

# 由于 redis 总是以最后的加工线作为一个配置指令值，所以你最好是把 include 放在这个文件的最前面，

# 以避免在运行时覆盖配置的改变，相反，你就把它放在后面（外国人真啰嗦）。

#

# include /path/to/local.conf

# include /path/to/other.conf

################################ 常用 #####################################

# 默认情况下 redis 不是作为守护进程运行的，如果你想让它在后台运行，你就把它改成 yes。

# 当redis作为守护进程运行的时候，它会写一个 pid 到 /var/run/redis.pid 文件里面。

daemonize no

# 当redis作为守护进程运行的时候，它会把 pid 默认写到 /var/run/redis.pid 文件里面，

# 但是你可以在这里自己制定它的文件位置。

pidfile /var/run/redis.pid

# 监听端口号，默认为 6379，如果你设为 0 ，redis 将不在 socket 上监听任何客户端连接。

port 6379

# TCP 监听的最大容纳数量

#

# 在高并发的环境下，你需要把这个值调高以避免客户端连接缓慢的问题。

# Linux 内核会一声不响的把这个值缩小成 /proc/sys/net/core/somaxconn 对应的值，

# 所以你要修改这两个值才能达到你的预期。

tcp-backlog 511

# 默认情况下，redis 在 server 上所有有效的网络接口上监听客户端连接。

# 你如果只想让它在一个网络接口上监听，那你就绑定一个IP或者多个IP。

#

# 示例，多个IP用空格隔开:

#

# bind 192.168.1.100 10.0.0.1

# bind 127.0.0.1

# 指定 unix socket 的路径。

#

# unixsocket /tmp/redis.sock

# unixsocketperm 755

# 指定在一个 client 空闲多少秒之后关闭连接（0 就是不管它）

timeout 0

# tcp 心跳包。

#

# 如果设置为非零，则在与客户端缺乏通讯的时候使用 SO\_KEEPALIVE 发送 tcp acks 给客户端。

# 这个之所有有用，主要由两个原因：

#

# 1) 防止死的 peers

# 2) Take the connection alive from the point of view of network

# equipment in the middle.

#

# On Linux, the specified value (in seconds) is the period used to send ACKs.

# Note that to close the connection the double of the time is needed.

# On other kernels the period depends on the kernel configuration.

#

# A reasonable value for this option is 60 seconds.

# 推荐一个合理的值就是60秒

tcp-keepalive 0

# 定义日志级别。

# 可以是下面的这些值：

# debug (适用于开发或测试阶段)

# verbose (many rarely useful info, but not a mess like the debug level)

# notice (适用于生产环境)

# warning (仅仅一些重要的消息被记录)

loglevel notice

# 指定日志文件的位置

logfile ""

# 要想把日志记录到系统日志，就把它改成 yes，

# 也可以可选择性的更新其他的syslog 参数以达到你的要求

# syslog-enabled no

# 设置 syslog 的 identity。

# syslog-ident redis

# 设置 syslog 的 facility，必须是 USER 或者是 LOCAL0-LOCAL7 之间的值。

# syslog-facility local0

# 设置数据库的数目。

# 默认数据库是 DB 0，你可以在每个连接上使用 select <dbid> 命令选择一个不同的数据库，

# 但是 dbid 必须是一个介于 0 到 databasees - 1 之间的值

databases 16

################################ 快照 ################################

#

# 存 DB 到磁盘：

#

# 格式：save <间隔时间（秒）> <写入次数>

#

# 根据给定的时间间隔和写入次数将数据保存到磁盘

#

# 下面的例子的意思是：

# 900 秒内如果至少有 1 个 key 的值变化，则保存

# 300 秒内如果至少有 10 个 key 的值变化，则保存

# 60 秒内如果至少有 10000 个 key 的值变化，则保存

#

# 注意：你可以注释掉所有的 save 行来停用保存功能。

# 也可以直接一个空字符串来实现停用：

# save ""

save 900 1

save 300 10

save 60 10000

# 默认情况下，如果 redis 最后一次的后台保存失败，redis 将停止接受写操作，

# 这样以一种强硬的方式让用户知道数据不能正确的持久化到磁盘，

# 否则就会没人注意到灾难的发生。

#

# 如果后台保存进程重新启动工作了，redis 也将自动的允许写操作。

#

# 然而你要是安装了靠谱的监控，你可能不希望 redis 这样做，那你就改成 no 好了。

stop-writes-on-bgsave-error yes

# 是否在 dump .rdb 数据库的时候使用 LZF 压缩字符串

# 默认都设为 yes

# 如果你希望保存子进程节省点 cpu ，你就设置它为 no ，

# 不过这个数据集可能就会比较大

rdbcompression yes

# 是否校验rdb文件

rdbchecksum yes

# 设置 dump 的文件位置

dbfilename dump.rdb

# 工作目录

# 例如上面的 dbfilename 只指定了文件名，

# 但是它会写入到这个目录下。这个配置项一定是个目录，而不能是文件名。

dir ./

################################# 主从复制 #################################

# 主从复制。使用 slaveof 来让一个 redis 实例成为另一个reids 实例的副本。

# 注意这个只需要在 slave 上配置。

#

# slaveof <masterip> <masterport>

# 如果 master 需要密码认证，就在这里设置

# masterauth <master-password>

# 当一个 slave 与 master 失去联系，或者复制正在进行的时候，

# slave 可能会有两种表现：

#

# 1) 如果为 yes ，slave 仍然会应答客户端请求，但返回的数据可能是过时，

# 或者数据可能是空的在第一次同步的时候

#

# 2) 如果为 no ，在你执行除了 info he salveof 之外的其他命令时，

# slave 都将返回一个 "SYNC with master in progress" 的错误，

#

slave-serve-stale-data yes

# 你可以配置一个 slave 实体是否接受写入操作。

# 通过写入操作来存储一些短暂的数据对于一个 slave 实例来说可能是有用的，

# 因为相对从 master 重新同步数而言，据数据写入到 slave 会更容易被删除。

# 但是如果客户端因为一个错误的配置写入，也可能会导致一些问题。

#

# 从 redis 2.6 版起，默认 slaves 都是只读的。

#

# Note: read only slaves are not designed to be exposed to untrusted clients

# on the internet. It's just a protection layer against misuse of the instance.

# Still a read only slave exports by default all the administrative commands

# such as CONFIG, DEBUG, and so forth. To a limited extent you can improve

# security of read only slaves using 'rename-command' to shadow all the

# administrative / dangerous commands.

# 注意：只读的 slaves 没有被设计成在 internet 上暴露给不受信任的客户端。

# 它仅仅是一个针对误用实例的一个保护层。

slave-read-only yes

# Slaves 在一个预定义的时间间隔内发送 ping 命令到 server 。

# 你可以改变这个时间间隔。默认为 10 秒。

#

# repl-ping-slave-period 10

# The following option sets the replication timeout for:

# 设置主从复制过期时间

#

# 1) Bulk transfer I/O during SYNC, from the point of view of slave.

# 2) Master timeout from the point of view of slaves (data, pings).

# 3) Slave timeout from the point of view of masters (REPLCONF ACK pings).

#

# It is important to make sure that this value is greater than the value

# specified for repl-ping-slave-period otherwise a timeout will be detected

# every time there is low traffic between the master and the slave.

# 这个值一定要比 repl-ping-slave-period 大

#

# repl-timeout 60

# Disable TCP\_NODELAY on the slave socket after SYNC?

#

# If you select "yes" Redis will use a smaller number of TCP packets and

# less bandwidth to send data to slaves. But this can add a delay for

# the data to appear on the slave side, up to 40 milliseconds with

# Linux kernels using a default configuration.

#

# If you select "no" the delay for data to appear on the slave side will

# be reduced but more bandwidth will be used for replication.

#

# By default we optimize for low latency, but in very high traffic conditions

# or when the master and slaves are many hops away, turning this to "yes" may

# be a good idea.

repl-disable-tcp-nodelay no

# 设置主从复制容量大小。这个 backlog 是一个用来在 slaves 被断开连接时

# 存放 slave 数据的 buffer，所以当一个 slave 想要重新连接，通常不希望全部重新同步，

# 只是部分同步就够了，仅仅传递 slave 在断开连接时丢失的这部分数据。

#

# The biggest the replication backlog, the longer the time the slave can be

# disconnected and later be able to perform a partial resynchronization.

# 这个值越大，salve 可以断开连接的时间就越长。

#

# The backlog is only allocated once there is at least a slave connected.

#

# repl-backlog-size 1mb

# After a master has no longer connected slaves for some time, the backlog

# will be freed. The following option configures the amount of seconds that

# need to elapse, starting from the time the last slave disconnected, for

# the backlog buffer to be freed.

# 在某些时候，master 不再连接 slaves，backlog 将被释放。

#

# A value of 0 means to never release the backlog.

# 如果设置为 0 ，意味着绝不释放 backlog 。

#

# repl-backlog-ttl 3600

# 当 master 不能正常工作的时候，Redis Sentinel 会从 slaves 中选出一个新的 master，

# 这个值越小，就越会被优先选中，但是如果是 0 ， 那是意味着这个 slave 不可能被选中。

#

# 默认优先级为 100。

slave-priority 100

# It is possible for a master to stop accepting writes if there are less than

# N slaves connected, having a lag less or equal than M seconds.

#

# The N slaves need to be in "online" state.

#

# The lag in seconds, that must be <= the specified value, is calculated from

# the last ping received from the slave, that is usually sent every second.

#

# This option does not GUARANTEES that N replicas will accept the write, but

# will limit the window of exposure for lost writes in case not enough slaves

# are available, to the specified number of seconds.

#

# For example to require at least 3 slaves with a lag <= 10 seconds use:

#

# min-slaves-to-write 3

# min-slaves-max-lag 10

#

# Setting one or the other to 0 disables the feature.

#

# By default min-slaves-to-write is set to 0 (feature disabled) and

# min-slaves-max-lag is set to 10.

################################## 安全 ###################################

# Require clients to issue AUTH <PASSWORD> before processing any other

# commands. This might be useful in environments in which you do not trust

# others with access to the host running redis-server.

#

# This should stay commented out for backward compatibility and because most

# people do not need auth (e.g. they run their own servers).

#

# Warning: since Redis is pretty fast an outside user can try up to

# 150k passwords per second against a good box. This means that you should

# use a very strong password otherwise it will be very easy to break.

#

# 设置认证密码

# requirepass foobared

# Command renaming.

#

# It is possible to change the name of dangerous commands in a shared

# environment. For instance the CONFIG command may be renamed into something

# hard to guess so that it will still be available for internal-use tools

# but not available for general clients.

#

# Example:

#

# rename-command CONFIG b840fc02d524045429941cc15f59e41cb7be6c52

#

# It is also possible to completely kill a command by renaming it into

# an empty string:

#

# rename-command CONFIG ""

#

# Please note that changing the name of commands that are logged into the

# AOF file or transmitted to slaves may cause problems.

################################### 限制 ####################################

# Set the max number of connected clients at the same time. By default

# this limit is set to 10000 clients, however if the Redis server is not

# able to configure the process file limit to allow for the specified limit

# the max number of allowed clients is set to the current file limit

# minus 32 (as Redis reserves a few file descriptors for internal uses).

#

# 一旦达到最大限制，redis 将关闭所有的新连接

# 并发送一个‘max number of clients reached’的错误。

#

# maxclients 10000

# 如果你设置了这个值，当缓存的数据容量达到这个值， redis 将根据你选择的

# eviction 策略来移除一些 keys。

#

# 如果 redis 不能根据策略移除 keys ，或者是策略被设置为 ‘noeviction’，

# redis 将开始响应错误给命令，如 set，lpush 等等，

# 并继续响应只读的命令，如 get

#

# This option is usually useful when using Redis as an LRU cache, or to set

# a hard memory limit for an instance (using the 'noeviction' policy).

#

# WARNING: If you have slaves attached to an instance with maxmemory on,

# the size of the output buffers needed to feed the slaves are subtracted

# from the used memory count, so that network problems / resyncs will

# not trigger a loop where keys are evicted, and in turn the output

# buffer of slaves is full with DELs of keys evicted triggering the deletion

# of more keys, and so forth until the database is completely emptied.

#

# In short... if you have slaves attached it is suggested that you set a lower

# limit for maxmemory so that there is some free RAM on the system for slave

# output buffers (but this is not needed if the policy is 'noeviction').

#

# 最大使用内存

# maxmemory <bytes>

# 最大内存策略，你有 5 个选择。

#

# volatile-lru -> remove the key with an expire set using an LRU algorithm

# volatile-lru -> 使用 LRU 算法移除包含过期设置的 key 。

# allkeys-lru -> remove any key accordingly to the LRU algorithm

# allkeys-lru -> 根据 LRU 算法移除所有的 key 。

# volatile-random -> remove a random key with an expire set

# allkeys-random -> remove a random key, any key

# volatile-ttl -> remove the key with the nearest expire time (minor TTL)

# noeviction -> don't expire at all, just return an error on write operations

# noeviction -> 不让任何 key 过期，只是给写入操作返回一个错误

#

# Note: with any of the above policies, Redis will return an error on write

# operations, when there are not suitable keys for eviction.

#

# At the date of writing this commands are: set setnx setex append

# incr decr rpush lpush rpushx lpushx linsert lset rpoplpush sadd

# sinter sinterstore sunion sunionstore sdiff sdiffstore zadd zincrby

# zunionstore zinterstore hset hsetnx hmset hincrby incrby decrby

# getset mset msetnx exec sort

#

# The default is:

#

# maxmemory-policy noeviction

# LRU and minimal TTL algorithms are not precise algorithms but approximated

# algorithms (in order to save memory), so you can tune it for speed or

# accuracy. For default Redis will check five keys and pick the one that was

# used less recently, you can change the sample size using the following

# configuration directive.

#

# The default of 5 produces good enough results. 10 Approximates very closely

# true LRU but costs a bit more CPU. 3 is very fast but not very accurate.

#

# maxmemory-samples 5

############################## APPEND ONLY MODE ###############################

# By default Redis asynchronously dumps the dataset on disk. This mode is

# good enough in many applications, but an issue with the Redis process or

# a power outage may result into a few minutes of writes lost (depending on

# the configured save points).

#

# The Append Only File is an alternative persistence mode that provides

# much better durability. For instance using the default data fsync policy

# (see later in the config file) Redis can lose just one second of writes in a

# dramatic event like a server power outage, or a single write if something

# wrong with the Redis process itself happens, but the operating system is

# still running correctly.

#

# AOF and RDB persistence can be enabled at the same time without problems.

# If the AOF is enabled on startup Redis will load the AOF, that is the file

# with the better durability guarantees.

#

# Please check http://redis.io/topics/persistence for more information.

appendonly no

# The name of the append only file (default: "appendonly.aof")

appendfilename "appendonly.aof"

# The fsync() call tells the Operating System to actually write data on disk

# instead to wait for more data in the output buffer. Some OS will really flush

# data on disk, some other OS will just try to do it ASAP.

#

# Redis supports three different modes:

#

# no: don't fsync, just let the OS flush the data when it wants. Faster.

# always: fsync after every write to the append only log . Slow, Safest.

# everysec: fsync only one time every second. Compromise.

#

# The default is "everysec", as that's usually the right compromise between

# speed and data safety. It's up to you to understand if you can relax this to

# "no" that will let the operating system flush the output buffer when

# it wants, for better performances (but if you can live with the idea of

# some data loss consider the default persistence mode that's snapshotting),

# or on the contrary, use "always" that's very slow but a bit safer than

# everysec.

#

# More details please check the following article:

# http://antirez.com/post/redis-persistence-demystified.html

#

# If unsure, use "everysec".

# appendfsync always

appendfsync everysec

# appendfsync no

# When the AOF fsync policy is set to always or everysec, and a background

# saving process (a background save or AOF log background rewriting) is

# performing a lot of I/O against the disk, in some Linux configurations

# Redis may block too long on the fsync() call. Note that there is no fix for

# this currently, as even performing fsync in a different thread will block

# our synchronous write(2) call.

#

# In order to mitigate this problem it's possible to use the following option

# that will prevent fsync() from being called in the main process while a

# BGSAVE or BGREWRITEAOF is in progress.

#

# This means that while another child is saving, the durability of Redis is

# the same as "appendfsync none". In practical terms, this means that it is

# possible to lose up to 30 seconds of log in the worst scenario (with the

# default Linux settings).

#

# If you have latency problems turn this to "yes". Otherwise leave it as

# "no" that is the safest pick from the point of view of durability.

no-appendfsync-on-rewrite no

# Automatic rewrite of the append only file.

# Redis is able to automatically rewrite the log file implicitly calling

# BGREWRITEAOF when the AOF log size grows by the specified percentage.

#

# This is how it works: Redis remembers the size of the AOF file after the

# latest rewrite (if no rewrite has happened since the restart, the size of

# the AOF at startup is used).

#

# This base size is compared to the current size. If the current size is

# bigger than the specified percentage, the rewrite is triggered. Also

# you need to specify a minimal size for the AOF file to be rewritten, this

# is useful to avoid rewriting the AOF file even if the percentage increase

# is reached but it is still pretty small.

#

# Specify a percentage of zero in order to disable the automatic AOF

# rewrite feature.

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

################################ LUA SCRIPTING ###############################

# Max execution time of a Lua script in milliseconds.

#

# If the maximum execution time is reached Redis will log that a script is

# still in execution after the maximum allowed time and will start to

# reply to queries with an error.

#

# When a long running script exceed the maximum execution time only the

# SCRIPT KILL and SHUTDOWN NOSAVE commands are available. The first can be

# used to stop a script that did not yet called write commands. The second

# is the only way to shut down the server in the case a write commands was

# already issue by the script but the user don't want to wait for the natural

# termination of the script.

#

# Set it to 0 or a negative value for unlimited execution without warnings.

lua-time-limit 5000

################################ REDIS 集群 ###############################

#

# 启用或停用集群

# cluster-enabled yes

# Every cluster node has a cluster configuration file. This file is not

# intended to be edited by hand. It is created and updated by Redis nodes.

# Every Redis Cluster node requires a different cluster configuration file.

# Make sure that instances running in the same system does not have

# overlapping cluster configuration file names.

#

# cluster-config-file nodes-6379.conf

# Cluster node timeout is the amount of milliseconds a node must be unreachable

# for it to be considered in failure state.

# Most other internal time limits are multiple of the node timeout.

#

# cluster-node-timeout 15000

# A slave of a failing master will avoid to start a failover if its data

# looks too old.

#

# There is no simple way for a slave to actually have a exact measure of

# its "data age", so the following two checks are performed:

#

# 1) If there are multiple slaves able to failover, they exchange messages

# in order to try to give an advantage to the slave with the best

# replication offset (more data from the master processed).

# Slaves will try to get their rank by offset, and apply to the start

# of the failover a delay proportional to their rank.

#

# 2) Every single slave computes the time of the last interaction with

# its master. This can be the last ping or command received (if the master

# is still in the "connected" state), or the time that elapsed since the

# disconnection with the master (if the replication link is currently down).

# If the last interaction is too old, the slave will not try to failover

# at all.

#

# The point "2" can be tuned by user. Specifically a slave will not perform

# the failover if, since the last interaction with the master, the time

# elapsed is greater than:

#

# (node-timeout \* slave-validity-factor) + repl-ping-slave-period

#

# So for example if node-timeout is 30 seconds, and the slave-validity-factor

# is 10, and assuming a default repl-ping-slave-period of 10 seconds, the

# slave will not try to failover if it was not able to talk with the master

# for longer than 310 seconds.

#

# A large slave-validity-factor may allow slaves with too old data to failover

# a master, while a too small value may prevent the cluster from being able to

# elect a slave at all.

#

# For maximum availability, it is possible to set the slave-validity-factor

# to a value of 0, which means, that slaves will always try to failover the

# master regardless of the last time they interacted with the master.

# (However they'll always try to apply a delay proportional to their

# offset rank).

#

# Zero is the only value able to guarantee that when all the partitions heal

# the cluster will always be able to continue.

#

# cluster-slave-validity-factor 10

# Cluster slaves are able to migrate to orphaned masters, that are masters

# that are left without working slaves. This improves the cluster ability

# to resist to failures as otherwise an orphaned master can't be failed over

# in case of failure if it has no working slaves.

#

# Slaves migrate to orphaned masters only if there are still at least a

# given number of other working slaves for their old master. This number

# is the "migration barrier". A migration barrier of 1 means that a slave

# will migrate only if there is at least 1 other working slave for its master

# and so forth. It usually reflects the number of slaves you want for every

# master in your cluster.

#

# Default is 1 (slaves migrate only if their masters remain with at least

# one slave). To disable migration just set it to a very large value.

# A value of 0 can be set but is useful only for debugging and dangerous

# in production.

#

# cluster-migration-barrier 1

# In order to setup your cluster make sure to read the documentation

# available at http://redis.io web site.

################################## SLOW LOG ###################################

# The Redis Slow Log is a system to log queries that exceeded a specified

# execution time. The execution time does not include the I/O operations

# like talking with the client, sending the reply and so forth,

# but just the time needed to actually execute the command (this is the only

# stage of command execution where the thread is blocked and can not serve

# other requests in the meantime).

#

# You can configure the slow log with two parameters: one tells Redis

# what is the execution time, in microseconds, to exceed in order for the

# command to get logged, and the other parameter is the length of the

# slow log. When a new command is logged the oldest one is removed from the

# queue of logged commands.

# The following time is expressed in microseconds, so 1000000 is equivalent

# to one second. Note that a negative number disables the slow log, while

# a value of zero forces the logging of every command.

slowlog-log-slower-than 10000

# There is no limit to this length. Just be aware that it will consume memory.

# You can reclaim memory used by the slow log with SLOWLOG RESET.

slowlog-max-len 128

############################# Event notification ##############################

# Redis can notify Pub/Sub clients about events happening in the key space.

# This feature is documented at http://redis.io/topics/keyspace-events

#

# For instance if keyspace events notification is enabled, and a client

# performs a DEL operation on key "foo" stored in the Database 0, two

# messages will be published via Pub/Sub:

#

# PUBLISH \_\_keyspace@0\_\_:foo del

# PUBLISH \_\_keyevent@0\_\_:del foo

#

# It is possible to select the events that Redis will notify among a set

# of classes. Every class is identified by a single character:

#

# K Keyspace events, published with \_\_keyspace@<db>\_\_ prefix.

# E Keyevent events, published with \_\_keyevent@<db>\_\_ prefix.

# g Generic commands (non-type specific) like DEL, EXPIRE, RENAME, ...

# $ String commands

# l List commands

# s Set commands

# h Hash commands

# z Sorted set commands

# x Expired events (events generated every time a key expires)

# e Evicted events (events generated when a key is evicted for maxmemory)

# A Alias for g$lshzxe, so that the "AKE" string means all the events.

#

# The "notify-keyspace-events" takes as argument a string that is composed

# by zero or multiple characters. The empty string means that notifications

# are disabled at all.

#

# Example: to enable list and generic events, from the point of view of the

# event name, use:

#

# notify-keyspace-events Elg

#

# Example 2: to get the stream of the expired keys subscribing to channel

# name \_\_keyevent@0\_\_:expired use:

#

# notify-keyspace-events Ex

#

# By default all notifications are disabled because most users don't need

# this feature and the feature has some overhead. Note that if you don't

# specify at least one of K or E, no events will be delivered.

notify-keyspace-events ""

############################### ADVANCED CONFIG ###############################

# Hashes are encoded using a memory efficient data structure when they have a

# small number of entries, and the biggest entry does not exceed a given

# threshold. These thresholds can be configured using the following directives.

hash-max-ziplist-entries 512

hash-max-ziplist-value 64

# Similarly to hashes, small lists are also encoded in a special way in order

# to save a lot of space. The special representation is only used when

# you are under the following limits:

list-max-ziplist-entries 512

list-max-ziplist-value 64

# Sets have a special encoding in just one case: when a set is composed

# of just strings that happens to be integers in radix 10 in the range

# of 64 bit signed integers.

# The following configuration setting sets the limit in the size of the

# set in order to use this special memory saving encoding.

set-max-intset-entries 512

# Similarly to hashes and lists, sorted sets are also specially encoded in

# order to save a lot of space. This encoding is only used when the length and

# elements of a sorted set are below the following limits:

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

# HyperLogLog sparse representation bytes limit. The limit includes the

# 16 bytes header. When an HyperLogLog using the sparse representation crosses

# this limit, it is converted into the dense representation.

#

# A value greater than 16000 is totally useless, since at that point the

# dense representation is more memory efficient.

#

# The suggested value is ~ 3000 in order to have the benefits of

# the space efficient encoding without slowing down too much PFADD,

# which is O(N) with the sparse encoding. The value can be raised to

# ~ 10000 when CPU is not a concern, but space is, and the data set is

# composed of many HyperLogLogs with cardinality in the 0 - 15000 range.

hll-sparse-max-bytes 3000

# Active rehashing uses 1 millisecond every 100 milliseconds of CPU time in

# order to help rehashing the main Redis hash table (the one mapping top-level

# keys to values). The hash table implementation Redis uses (see dict.c)

# performs a lazy rehashing: the more operation you run into a hash table

# that is rehashing, the more rehashing "steps" are performed, so if the

# server is idle the rehashing is never complete and some more memory is used

# by the hash table.

#

# The default is to use this millisecond 10 times every second in order to

# active rehashing the main dictionaries, freeing memory when possible.

#

# If unsure:

# use "activerehashing no" if you have hard latency requirements and it is

# not a good thing in your environment that Redis can reply form time to time

# to queries with 2 milliseconds delay.

#

# use "activerehashing yes" if you don't have such hard requirements but

# want to free memory asap when possible.

activerehashing yes

# The client output buffer limits can be used to force disconnection of clients

# that are not reading data from the server fast enough for some reason (a

# common reason is that a Pub/Sub client can't consume messages as fast as the

# publisher can produce them).

#

# The limit can be set differently for the three different classes of clients:

#

# normal -> normal clients

# slave -> slave clients and MONITOR clients

# pubsub -> clients subscribed to at least one pubsub channel or pattern

#

# The syntax of every client-output-buffer-limit directive is the following:

#

# client-output-buffer-limit <class> <hard limit> <soft limit> <soft seconds>

#

# A client is immediately disconnected once the hard limit is reached, or if

# the soft limit is reached and remains reached for the specified number of

# seconds (continuously).

# So for instance if the hard limit is 32 megabytes and the soft limit is

# 16 megabytes / 10 seconds, the client will get disconnected immediately

# if the size of the output buffers reach 32 megabytes, but will also get

# disconnected if the client reaches 16 megabytes and continuously overcomes

# the limit for 10 seconds.

#

# By default normal clients are not limited because they don't receive data

# without asking (in a push way), but just after a request, so only

# asynchronous clients may create a scenario where data is requested faster

# than it can read.

#

# Instead there is a default limit for pubsub and slave clients, since

# subscribers and slaves receive data in a push fashion.

#

# Both the hard or the soft limit can be disabled by setting them to zero.

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

client-output-buffer-limit pubsub 32mb 8mb 60

# Redis calls an internal function to perform many background tasks, like

# closing connections of clients in timeout, purging expired keys that are

# never requested, and so forth.

#

# Not all tasks are performed with the same frequency, but Redis checks for

# tasks to perform accordingly to the specified "hz" value.

#

# By default "hz" is set to 10. Raising the value will use more CPU when

# Redis is idle, but at the same time will make Redis more responsive when

# there are many keys expiring at the same time, and timeouts may be

# handled with more precision.

#

# The range is between 1 and 500, however a value over 100 is usually not

# a good idea. Most users should use the default of 10 and raise this up to

# 100 only in environments where very low latency is required.

hz 10

# When a child rewrites the AOF file, if the following option is enabled

# the file will be fsync-ed every 32 MB of data generated. This is useful

# in order to commit the file to the disk more incrementally and avoid

# big latency spikes.

aof-rewrite-incremental-fsync yes

[复制代码](javascript:void(0);)