中移在线服务有限公司

Redis缓存集群部署规范

**中移在线服务有限公司**

**二〇一八年九月**

目录

[中移在线服务有限公司 I](#_Toc12096)

[Redis缓存集群部署规范 I](#_Toc20256)

[1 目的 1](#_Toc29533)

[2 范围 1](#_Toc17404)

[3 术语、定义和缩略语 1](#_Toc32584)

[4 软件信息 1](#_Toc22104)

[5 安装步骤 1](#_Toc17660)

[6 集群配置 3](#_Toc12675)

[6.1 配置文件 3](#_Toc18116)

[6.2 配置说明 5](#_Toc10736)

[6.3 调优参数 10](#_Toc15700)

[7 集群部署规范 10](#_Toc7011)

[8 运维流程规范 15](#_Toc2996)

[8.1 接入流程 15](#_Toc25014)

[8.2 垃圾键值清除流程 17](#_Toc22405)

[9 附录 19](#_Toc2182)

[9.1 Info系统状态说明 19](#_Toc4199)

[9.1.1 info Server 模块统计信息 19](#_Toc29065)

[9.1.2 info Clinets模块统计信息 20](#_Toc18308)

[9.1.3 info Memory模块统计信息 20](#_Toc28255)

[9.1.4 info Persistence模块统计信息 20](#_Toc147)

[9.1.5 info Stats模块统计信息 21](#_Toc29210)

[9.1.6 info Replication模块统计信息 22](#_Toc12236)

[9.1.7 info cpu模块统计信息 22](#_Toc24529)

[9.1.8 info Commandstas模块统计信息 23](#_Toc29861)

[9.1.9 info Cluster模块统计信息 23](#_Toc26356)

[9.1.10 info Keyspace模块统计信息 23](#_Toc24401)

[9.2 Redis配置说明 24](#_Toc9036)

[9.2.1 总体配置 24](#_Toc16214)

[9.2.2 最大内存及策略 25](#_Toc9658)

[9.2.3 AOF相关配置 25](#_Toc11190)

[9.2.4 RDB相关配置 26](#_Toc29753)

[9.2.5 慢查询相关配置 27](#_Toc14719)

[9.2.6 数据结构优化配置 27](#_Toc17072)

[9.2.7 复制相关配置 28](#_Toc15295)

[9.2.8 客户端相关配置 29](#_Toc29592)

[9.2.9 安全相关配置 29](#_Toc46)

[9.2.10 cluster相关配置 29](#_Toc9853)

[9.2.11 Sentinel配置说明和分析 30](#_Toc22025)

[10 附件 31](#_Toc3047)

[10.1 Ruby集群创建脚本 31](#_Toc22777)

[10.2 缓存应用场景统计 32](#_Toc12592)

# 目的

为进一步促进IT系统开发过程中Redis缓存数据库使用的标准化、规范化程度，从而确保公司IT系统开发质量，提升系统的访问性能，针对Redis缓存数据库的相关特性，围绕Redis中间件部署约束，特制定本规范。

# 范围

本规范适用于总部、各分公司基于Redis缓存数据库所开发的应用程序。

# 术语、定义和缩略语

本规范采用以下的术语描述：

* 【强制】必须遵从使用的原则。
* 【规则】如无特殊情况，应该遵守的原则。
* 【推荐】建议遵守的原则。
* 【参考】运维人员参考了解的原则。

# 软件信息

|  |  |
| --- | --- |
| 软件名称 | 版本号 |
| Redis | 3.2.6 |
| Zlib | 按照操作系统自带 |
| Ruby | 2.3.3 |
| Rubygem | 2.6.8 |
| Redis.gem | 3.3.1 |

# 安装步骤

1. 在需要执行创建集群脚本所在主机安装ruby rubygem 以及ruby语言redis支持包redis.gem:

cd /home/itframe/software/ruby

tar –xzf ruby-2.3.3.tar.gz

cd ruby-2.3.3

./configure --prefix=/home/ itframe /ruby

make && make install

export PATH=.:/home/ itframe /ruby/bin:$PATH

cd /home/ itframe /software/ruby

tar –xzf rubygems-2.6.8.tgz

cd rubygems-2.6.8

ruby setup.rb

cd /home/ itframe /software/ruby

gem install -l redis-3.3.1.gem

ps:

如果在gem install 时报错，no such file to load – zlib

首先检查主机是否安装zlib以及zlib-dev，若已安装还是报错，可按照以下步骤解决：

进入ruby源码文件夹

安装ruby自身提供的zlib包

cd ext/zlib

ruby ./extconf.rb

make

make install

1. Redis编译安装

cd /home/itframe/software

tar –xzf redis-3.2.6.tar.gz

cd redis-3.2.6

make

1. Redis目录结构要求

bin目录下存储编译后的可执行文件及代码

config目录下存储配置文件

data目录下存储aof，rdb，node.conf等数据文件

log目录下存储日志

配置文件按此目录结构进行修改

示例：

**redis/  
├── bin  
│   ├──redis-server.c**

**├── config  
│   ├── redis-8000.conf  
│   ├── redis-8001.conf  
├── data  
│   ├── appendonly-8000.aof  
│   ├── appendonly-8001.aof**

**│   ├── dump-8000.rdb  
│   ├── dump-8001.rdb  
│   ├── nodes-8000.conf  
│   ├── nodes-8001.conf  
├── log  
│   ├── redis-8000.log  
│   ├── redis-8000.pid  
│   ├── redis-8001.log  
│   ├── redis-8002.pid  
└── CreateCluster.sh**

# 集群配置

## 配置文件

结合公司Redis应用场景将Redis集群配置统一为如下模板：

1. 提供RDB持久化机制关闭AOF持久化机制提高集群稳定性与服务器数据访问性能；
2. 增大服务端客户端缓存区大小与复制积压缓冲区大小，防止主从无限复制故障。

该模板适用于对数据高可用有一定要求，能够在异常宕机时容忍少量数据丢失，满足并发量大，单个数据报文较小的应用场景。

#设置为守护进程  
daemonize yes   
#Redis运行的进程pid文件  
pidfile /home/itframe/redis/log/redis-8000.pid   
#Redis服务端口号  
port 8000   
#Redis服务绑定ip  
bind 192.168.100.144   
#最大内存  
maxmemory 8gb  
#开启集群模式  
cluster-enabled yes   
#节点配置文件  
cluster-config-file nodes-8000.conf  
#集群节点超时时间（单位：毫秒)  
cluster-node-timeout 15000  
#集群是否需要所有的slot都分配给在线节点，才能正常访问  
cluster-require-full-coverage no  
#工作目录（aof、rdb、日志文件)  
dir /home/itframe/redis/data  
#tcp-backlog  
tcp-backlog 511  
#客户端闲置多少秒后关闭连接（单位：秒）  
timeout 300  
#检测TCP连接活性的周期（单位：秒）  
tcp-keepalive 60  
#日志级别  
loglevel verbose  
#日志记录目录  
logfile "/home/itframe/redis/log/redis-8000.log"  
#可用的数据库数  
databases 16  
#RDB保存条件  
save 900 1  
save 300 10  
save 60 10000  
#bgsave执行错误，是否停止Redis接受请求  
stop-writes-on-bgsave-error no  
#RDB文件是否压缩  
rdbcompression yes  
#RDB文件是否使用校验和  
rdbchecksum yes  
#RDB文件名  
dbfilename dump-8000.rdb  
#当从节点与主节点连接中断时，如果此参数值设置为“yes”，从节点可以继续处理客户端的#请求。否则除info和slaveof命令之外，拒绝的所有请求并统一回复"SYNC with master in #progress"  
slave-serve-stale-data yes  
#从节点是否开启只读模式，集群架构下从节点默认读写都不可用，需要调用readyonly命令#开启只读模式  
slave-read-only yes  
#是否开启无盘复制  
repl-diskless-sync no  
#开启无盘复制后，需要延迟多少秒后进行创建RDB操作，一般用于同时加入多个从节点时，#保证多个从节点可共享RDB  
repl-diskless-sync-delay 5  
#是否开启主从复制socket的NO\_DELAY选项：yes:Redis会合并小的TCP包来节省带宽，但##是这样增加同步延迟，造成主#从数据不一致；no:主节点会立即发送同步数据，没有延迟  
repl-disable-tcp-nodelay no  
#从节点的优先级  
slave-priority 100  
#是否开启AOF持久化模式  
appendonly no  
#Lua脚本“超时时间”（单位：毫秒）  
lua-time-limit 5000  
#慢查询被记录的阀值（单位微秒）  
slowlog-log-slower-than 10000  
#最多记录慢查询的条数  
slowlog-max-len 1000  
#Redis服务内存延迟监控  
latency-monitor-threshold 0  
notify-keyspace-events ""  
hash-max-ziplist-entries 512  
hash-max-ziplist-value 64  
list-max-ziplist-entries 512  
list-max-ziplist-value 64  
set-max-intset-entries 512  
zset-max-ziplist-entries 128  
zset-max-ziplist-value 64  
hll-sparse-max-bytes 3000  
#是否激活重置哈希  
activerehashing yes  
#客户端输出缓冲区限制  
client-output-buffer-limit normal 0 0 0  
client-output-buffer-limit slave 512mb 128mb 60  
client-output-buffer-limit pubsub 32mb 8mb 60  
#复制积压缓存区大小  
repl-backlog-size 256mb  
#redis server执行后台任务的频率,默认为10  
hz 10  
#最大客户端连接数  
maxclients 15000

**PS：标红内容为变更项，在部署过程中请结合实际部署要求进行变更**

## 配置说明

|  |  |
| --- | --- |
| 配置属性 | 配置说明 |
| daemonize yes | 默认redis不是以后台进程的方式启动，如果需要在后台运行，需要将这个值设置成yes ，以后台方式启动的时候，redis会写入默认的进程文/var/run/redis.pid |
| pidfile | redis启动的进程路径 |
| port | 实例端口 (请勿配置为默认端口) |
| bind | 地址IP (务必绑定端口) |
| maxclients 15000 | 限制同时连接的客户数量。当连接数超过这个值时，redis 将不再接收其他连接请求，客户端尝试连接时将收到error 信息。默认为10000，要考虑系统文件描述符限制，不宜过大，浪费文件描述符 |
| maxmemory <bytes> | redis实例分配的最大内存，如果一个物理节点部署一个redis实例，maxmemory应最多设置为物理内存的2/3，预留一定系统内存空间 |
| tcp-backlog 511 | 在高并发环境下需要一个高backlog值来避免慢客户端连接问题，linux操作系统的tcp-backlog需要大于或等于redis实例的backlog数值，否则redis服务端会抛出异常 |
| timeout 300 | 指定socket连接空闲时间(单位：秒).如果connection空闲超时,将会关闭连接(TCP socket选项)如果为0,表示永不超时. |
| tcp-keepalive 60 | 指定TCP连接是否为长连接,"侦探"信号有server端维护,长连接将会额外的增加server端的开支(TCP socket选项) 默认为0.表示禁用,非0值表示开启"长连接";"侦探"信号的发送间隔将有linux系统决定在多次"侦探"后,如果对等端(客户端socket)仍不回复,将会关闭连接,否则连接将会被保持开启.client端socket也可以通过配置keepalive选项,开启"长连接".（单位：秒） |
| loglevel notice (生产环境建议使用此级别) | server日志级别,合法值:debug,verbose,notice,warning 默认为notice debug适合开发环境,客户端操作信息都会输出日志 verbose输出一些相对有用的信息,目前效果不明 notice适合生产环境 warning异常信息 |
| logfile | 日志路径如"/home/rhkf/redis11/log/20011.log" |
| databases 16 | 设定redis所允许的最大"db簇"的个数,默认为16个簇. 客户端可以通过"select"指令指定需要使用的"db簇"索引号,默认为0.redis的顶层数据结构中,所有K-V都潜在的包括了"db簇"索引号,任何一个key都将隶属于一个"db".任何对数据的检索,只会覆盖指定的"db"; 例如数据被插入到"db 10"中,那么在"db 1"中去get,将会返回null.对数据归类到不同的db簇中,可以帮助我们实现一些特定的需求,比如根据不同客户端连接,来指定不同的db索引号. |
| save 900 1 save 300 10 save 60 10000 | snapshot配置,save <seconds> <changes>,用来描述"在多少秒期间至少多少个变更操作"触发snapshot #snapshot最终将生成新的dump.rdb文件 save 900 1 # 900秒内有至少1个键被更改则进行快照 save 300 10 # 300秒内有至少10个键被更改则进行快照 save 60 10000 # 60秒内有至少10000个键被更改则进行快照 |
| stop-writes-on-bgsave-error no | 当持久化出现错误时，是否依然继续进行工作，是否终止所有的客户端write请求。默认设置"yes"表示终止，一旦snapshot数据保存故障，那么此server为只读服务。如果为"no"，那么此次snapshot将失败，但下一次snapshot不会受到影响，不过如果出现故障,数据只能恢复到"最近一个成功点" |
| rdbcompression yes | 在进行数据镜像备份时，是否启用rdb文件压缩手段，默认为yes。压缩可能需要额外的cpu开支，不过这能够有效的减小rdb文件的大，有利于存储/备份/传输/数据恢复 |
| rdbchecksum yes | checksum文件检测，读取写入的时候rdb文件checksum，会损失一些性能 |
| dbfilename dump.rdb | 镜像备份文件的文件名，默认为dump.rdb |
| dir ./ | RDB文件路径 |
| slave-serve-stale-data yes | 当主master服务器挂机或主从复制在进行时，是否依然可以允许客户访问可能过期的数据。在"yes"情况下,slave继续向客户端提供只读服务,有可能此时的数据已经过期；在"no"情况下，任何向此server发送的数据请求服务(包括客户端和此server的slave)都将被告知"error" |
| slave-read-only yes | 如果是slave库，只允许只读，不允许修改（集群模式下失效） |
| repl-diskless-sync no | slave与master的连接,是否禁用TCPnodelay选项。"yes"表示禁用,那么socket通讯中数据将会以packet方式发送(packet大小受到socket buffer限制)。可以提高socket通讯的效率(减小tcp交互次数),但是小数据将会被buffer,不会被立即发送,对于接受者可能存在延迟。"no"表示开启tcp nodelay选项,任何数据都会被立即发送,及时性较好,但是效率较低，建议设为no，在高并发或者主从有大量操作的情况下，设置为yes |
| repl-diskless-sync-delay 5 | repl-diskless-sync-delay |
| repl-disable-tcp-nodelay no |  |
| slave-priority 100 | 适用Sentinel模块(unstable,M-S集群管理和监控),需要额外的配置文件支持。slave的权重值,默认100.当master失效后,Sentinel将会从slave列表中找到权重值最低(>0)的slave,并提升为master。如果权重值为0,表示此slave为"观察者",不参与master选举 |
| appendonly no | 默认情况下，redis 会在后台异步的把数据库镜像备份到磁盘，但是该备份是非常耗时的，而且备份也不能很频繁。所以redis 提供了另外一种更加高效的数据库备份及灾难恢复方式。开启append only 模式之后，redis 会把所接收到的每一次写操作请求都追加到appendonly.aof 文件中，当redis 重新启动时，会从该文件恢复出之前的状态。但是这样会造成appendonly.aof 文件过大，所以redis 还支持了BGREWRITEAOF 指令，对appendonly.aof 进行重新整理。如果不经常进行数据迁移操作，推荐生产环境下的做法为关闭镜像，开启appendonly.aof，同时可以选择在访问较少的时间每天对appendonly.aof 进行重写一次。在并发量大情况下需要将AOF持久化策略关闭，避免频繁操作磁盘造成阻塞 |
| lua-time-limit 5000 | lua脚本执行的最大时间，单位毫秒 |
| slowlog-log-slower-than 10000 | 慢日志记录，单位微妙，10000就是10毫秒值，如果操作时间超过此值,将会把command信息"记录"起来.(内存,非文件)。其中"操作时间"不包括网络IO开支,只包括请求达到server后进行"内存实施"的时间."0"表示记录全部操作 |
| slowlog-max-len 1000 | "慢操作日志"保留的最大条数,"记录"将会被队列化,如果超过了此长度,旧记录将会被移除。可以通过"SLOWLOG<subcommand> args"查看慢记录的信息(SLOWLOG get 10,SLOWLOG reset) |
| latency-monitor-threshold 0 | 延迟监控，用于记录等于或超过了指定时间的操作，默认是关闭状态，即值为0。 |
| notify-keyspace-events "" | notify-keyspace-events 的参数可以是以下字符的任意组合， 它指定了服务器该发送哪些类型的通知： 字符 发送的通知 K 键空间通知，所有通知以 \_\_keyspace@<db>\_\_ 为前缀 E 键事件通知，所有通知以 \_\_keyevent@<db>\_\_ 为前缀 g DEL 、 EXPIRE 、 RENAME 等类型无关的通用命令的通知 $ 字符串命令的通知 l 列表命令的通知 s 集合命令的通知 h 哈希命令的通知 z 有序集合命令的通知 x 过期事件：每当有过期键被删除时发送 e 驱逐(evict)事件：每当有键因为 maxmemory 政策而被删除时发送 A 参数 g$lshzxe 的别名 |
| cluster-enabled yes | redis实例支持集群模式 |
| cluster-config-file | 集群配置文件 |
| cluster-node-timeout 15000 | 单位是ms，节点会因为某些原因发生阻塞(阻塞时间大于clutser-node-timeout），被判断下线。这种failover是没有必要，可以适当增大。  故障转移时间预估：  Failover-time(毫秒)<=cluster-node-timeout+cluster-node-timeout/2+1000 |
| cluster-require-full-coverage no | 表示当负责一个插槽的主库下线且没有相应的从库进行故障恢复时，集群仍然可用 |
| repl-backlog-size 256mb（优化值） | 复制积压缓冲区，默认为1mb，对于复制积压缓冲区整个主节点只有一个，所有从节点共享此缓冲区 |
| aof-rewrite-incremental-fsync yes | aof rewrite过程中,是否采取增量"文件同步"策略,默认为"yes",而且必须为yes.rewrite过程中,每32M数据进行一次文件同步,这样可以减少"aof大文件"写入对磁盘的操作次数. |
| hash-max-ziplist-entries 512 hash-max-ziplist-value 64 list-max-ziplist-entries 512 list-max-ziplist-value 64 | hash类型的数据结构在编码上可以使用ziplist和hashtable。ziplist的特点就是文件存储(以及内存存储)所需的空间较小,在内容较小时,性能和hashtable几乎一样.因此redis对hash类型默认采取ziplist。如果hash中条目的条目个数或者value长度达到阀值,将会被重构为hashtable。 |
| set-max-intset-entries 512 | intset中允许保存的最大条目个数,如果达到阀值,intset将会被重构为hashtable |
| zset-max-ziplist-entries 128 | zset为有序集合,有2中编码类型:ziplist,skiplist。因为"排序"将会消耗额外的性能,当zset中数据较多时,将会被重构为skiplis。 |
| zset-max-ziplist-value 64 | zset中允许条目value值最大字节数，默认为64，建议为1024 |
| hll-sparse-max-bytes 3000 | 稀疏表示字节限制 |
| activerehashing yes | 是否开启顶层数据结构的rehash功能,如果内存允许,请开启。rehash能够很大程度上提高K-V存取的效率 |
| client-output-buffer-limit normal 0 0 0 client-output-buffer-limit slave 512mb 128mb 60 client-output-buffer-limit pubsub 32mb 8mb 60 | 客户端buffer控制。在客户端与server进行的交互中,每个连接都会与一个buffer关联,此buffer用来队列化等待被client接受的响应信息。如果client不能及时的消费响应信息,那么buffer将会被不断积压而给server带来内存压力.如果buffer中积压的数据达到阀值,将会导致连接被关闭,buffer被移除。  buffer控制类型包括:normal -> 普通连接；slave->与slave之间的连接；pubsub ->pub/sub类型连接，此类型的连接，往往会产生此种问题;因为pub端会密集的发布消息,但是sub端可能消费不足.指令格式:client-output-buffer-limit <class> <hard><soft><seconds>",其中hard表示buffer最大值,一旦达到阀值将立即关闭连接;soft表示"容忍值",它和seconds配合,如果buffer值超过soft且持续时间达到了seconds,也将立即关闭连接,如果超过了soft但是在seconds之后，buffer数据小于了soft,连接将会被保留.其中hard和soft都设置为0,则表示禁用buffer控制.通常hard值大于soft. |
| hz 10 | Redis server执行后台任务的频率,默认为10,此值越大表示redis对"间歇性task"的执行次数越频繁(次数/秒) "间歇性task"包括"过期集合"检测、关闭"空闲超时"的连接等,此值必须大于0且小于500.(参见redis.h源码) 此值过小就意味着更多的cpu周期消耗,后台task被轮询的次数更频繁，此值过大意味着"内存敏感"性较差. 建议保持默认值 |

## 调优参数

|  |  |
| --- | --- |
| 配置参数 | 调优说明 |
| save 900 1  save 300 10  save 60 10000 | 在业务系统高并发环境下，每60s触发10000次命令操作就会执行bgsave操作，可以将命令操作以10000每单位进行递增  如save 60 20000  即每60s触发20000次命令操作执行bgsave操作，降低磁盘I/O使用率 |
| appendonly no | 关闭AOF持久化机制提高系统稳定性与数据访问效率 |
| client-output-buffer-limit normal 0 0 0  client-output-buffer-limit slave 512mb 128mb 60  client-output-buffer-limit pubsub 32mb 8mb 60 | 当单实例缓存数据量较大情况下可以将client-output-buffer-limit slave 256mb 128mb 60设置为client-output-buffer-limit slave 0 0 0 |
| repl-backlog-size 256mb | 当单实例缓存数据较大时，可以适当调大复制积压缓冲区大小以100mb为增幅， |

# 集群部署规范

【推荐】集群每一个Redis实例都单独部署在一个物理机，考虑到容错和网络环境尽量实现跨机柜部署。

说明：

Redis集群中每一台应用实例都单独部署在一个物理机上和跨机房部署，能够实现高度的集群容错状态，Redis集群本身提供容错特性。对于三主三从的集群来说，一台主节点宕机响应的从节点会自动生成主节点集群能够继续对外提供服务，同时由于是单独部署一台物理机宕机不会影响缓存集群的其他节点对外提供服务。

【强制】禁止集群的一对主从部署在同一台物理机。

说明：

如果集群中一个master宕机，集群领导选举的过程是集群中所有活跃master参与，如果半数以上活跃的master与master节点通信超过(cluster-node-timeout)，认为当前的master节点挂掉。

集群不可用场景：

如果配置文件中cluster-require-full-coverage配置属性设置为yes，当集群中任何一对主从挂机集群进入fail状态。

如果配置文件中cluster-require-full-coverage配置属性设置为no，如果集群超过半数以上的master改掉，无论是否有slave集群进入fail状态。

如果集群中一对主从部署在同一台物理机，该物理机宕机集群分片不完整，部分数据没有对应的slots分片不能进行缓存数据操作，造成大量缓存数据与数据库数据的不一致状态。

【强制】禁止集群中两个或两个以上主节点部署在同一台物理机

说明：

集群中master节点响应客户端请求，salve节点实现集群的高可用机制。如果集群超过半数以上的master改掉，无论是否有slave集群进入fail状态。

集群推荐部署方式示例如下（三主三从）

|  |  |  |  |
| --- | --- | --- | --- |
| **实例信息** | **物理主机** | **主节点** | **从节点** |
| 192.168.100.144 | 物理机A | Master1 | Slave2 |
| 192.168.100.146 | 物理机B | Master2 | Slave3 |
| 192.168.100.152 | 物理机C | Master3 | Slave1 |

集群构建脚本CreateCluster.sh（以redis-trib.rb构建为例）：

#!/bin/bash

export PATH=/home/itframe/ruby/bin:.:$PATH

./bin/redis-trib.rb create-manual G 192.168.100.144:8000 192.168.100.146:8003 G 192.168.100.146:8002 192.168.100.152:8005 G 192.168.100.152:8004 192.168.100.144:8001

echo "success...."

exit 0

redis-trib.rb create-manual G 192.168.100.144:8000 192.168.100.146:8003 G 192.168.100.146:8002 192.168.100.152:8005 G 192.168.100.152:8004 192.168.100.144:8001主从关系为：

192.168.100.146:8003 slaveof 192.168.100.144:8000

192.168.100.152:8005 slaveof 192.168.100.146:8002

192.168.100.144:8001 slaveof 192.168.100.152:8004

PS：

G代表一个分组，G后面第一个节点信息为主节点，第一个节点信息至下一个“G”符号标识的节点信息为改主节点的从节点（一个主节点可以配置多个从节点）

【强制】集群部署的Redis实例主机必须预留集群总线（cluster bus）的通信端口。

说明：

采用集群部署的Redis实例主机必须预留集群总线的通信端口，及Redis实例端口+10000。例如主机A部署的Redis实例端口为7000，则集群的总线通信端口为17000。

【规则】集群部署Redis实例的Linux操作系统内存分配策略vm.overcommit\_memory属性设置为1。

说明：

Linux操作系统对于大部分申请内存的请求都回复yes，以便运行更多程序。在申请内存后，并不会马上应用内存，这种技术叫做overcommit。vm.overcommit\_memory用来设置内存分配策略有三个可选值。

|  |  |
| --- | --- |
| 值 | 含义 |
| 0 | 表示内核将检查是否有足够的可用内存，如果有足够的可用内存，内存申请通过，否则内存申请失败，并把错误返回给应用进程 |
| 1 | 表示内核允许超量使用内存直到用完为止 |
| 2 | 表示内核决不过量的（“never overcommit”）使用内存，即系统整个内存地址空间不能超过swap+50%的RAM值，50%是overcommit\_ratio默认值，此参数同样支持修改。 |

【推荐】部署缓存集群的主机系统建议关闭THP，防止copy-on-write期间过度的内存损耗。

说明：

如Redis子进程内存消耗主要指执行AOF/RDB重写时Redis创建的子进程内存消耗。Redis执行fork操作产生的子进程内存占用量对外表现为与父进程相同，理论上需要一倍的物理内存来完成重写操作。但Linux具备写时复制技术（copy-on-write），父进程会共享相同的物理内存页，当父进程处理写请求是会对需要修改的页复制出一个副本完成写操作，而子进程依然读取fork时整个父进程的内存快照。虽然开启THP可以降低fork子进程的时间，但之后copy-on-write期间复制内存页的单位从4KB变成2MB，如果父进程有大量的写请求，会加重内存拷贝量，从而造成过度的内存损耗。

【推荐】在线上环境建议每个Redis实例内存控制在10GB以内。

说明：

Redis执行RDB和AOF持久化，必不可少的操作就是执行fork操作创建子进程，fork耗时跟内存量成正比。父进程在fork期间不能响应客户端的命令请求。

【推荐】线上环境redis实例的maxmemory应设置物理内存的2/3,保证机器有20%-30%的闲置内存。

说明：

Redis默认无限使用服务器内存，为防止极端情况下导致系统内存耗尽，建议所有的Redis进程都要配置maxmemory。

如果一台物理机部署一个Redis实例，物理机内存的内存配置为6g，则maxmemory应设置为4g。

如果一台物理机部署两个Redis实例，物理机内存的内存配置为6g，则maxmemory应设置为2g。

【推荐】线上环境redis实例的maxclients设置限制为15000

说明：

限制同时连接的客户数量。当连接数超过这个值时，redis 将不再接收其他连接请求，客户端尝试连接时将收到error 信息。默认为10000，要考虑系统文件描述符限制，不宜过大，浪费文件描述符。

【强制】操作系统的tcp-backlog需要大于或等于redis实例的backlog数值

说明：

Redis默认的tcp-backlog值为511，在高并发环境下需要一个高backlog值来避免慢客户端连接问题，如果Linux的tcp-backlog小于Redis设置的tcp-backlog服务端启动是会抛出异常。

【强制】设置操作系统open files至少为maxclients+32

说明：

Redis允许同时有多个客户端通过网络进行连接，可以通过配置maxclients来限制最大客户端连接数。对于linux操作系统来说，每一个网络连接都是文件句柄，除此之外Redis内部会使用最多32个文件描述符，所以操作系统open file需要设置至少maxclients+32。

【规则】线上环境redis实例的tcp-keepalive设置限制为60 ，timeout设置为300

说明：

tcp-keepalive指定TCP连接是否为长连接,"侦探"信号有server端维护,长连接将会额外的增加server端的开支(TCP socket选项)默认为0.表示禁用,非0值表示开启"长连接";"侦探"信号的发送间隔将由linux系统决定在多次"侦探"后,如果对等端(客户端socket)仍不回复,将会关闭连接,否则连接将会被保持开启。timeout设置为5s说明客户端连接空闲300s后将关闭连接。（单位：秒）

【规则】Redis不应和高磁盘I/O，高计算任务的应用部署在同一台物理机上

说明：

Redis的持久化消耗大量的硬盘I/O，与其他的高磁盘I/O，高计算任务的应用部署在同一台物理机上会相互印象，降低Redis服务性能。

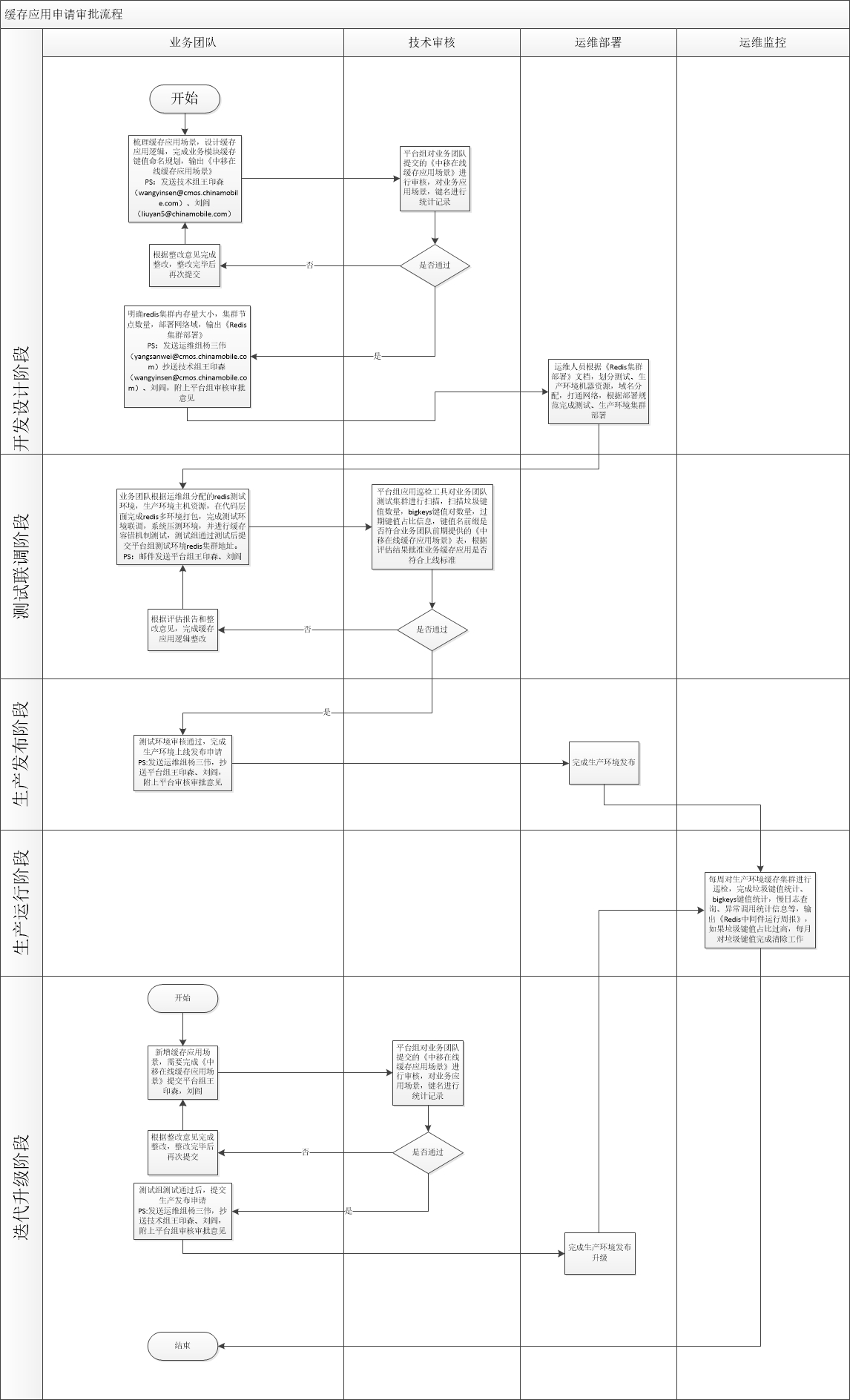
【推荐】建议物理机替换固态硬盘提高Redis持久化磁盘读写性能。

说明：

当开启AOF功能的Redis用于高流量写入场景时，如果应用普通机械硬盘，写入吞吐一般在100MB/s左右，这时Redis实例的瓶颈主要在AOF同步硬盘上，建议替换固态硬盘或放宽AOF文件同步策略。

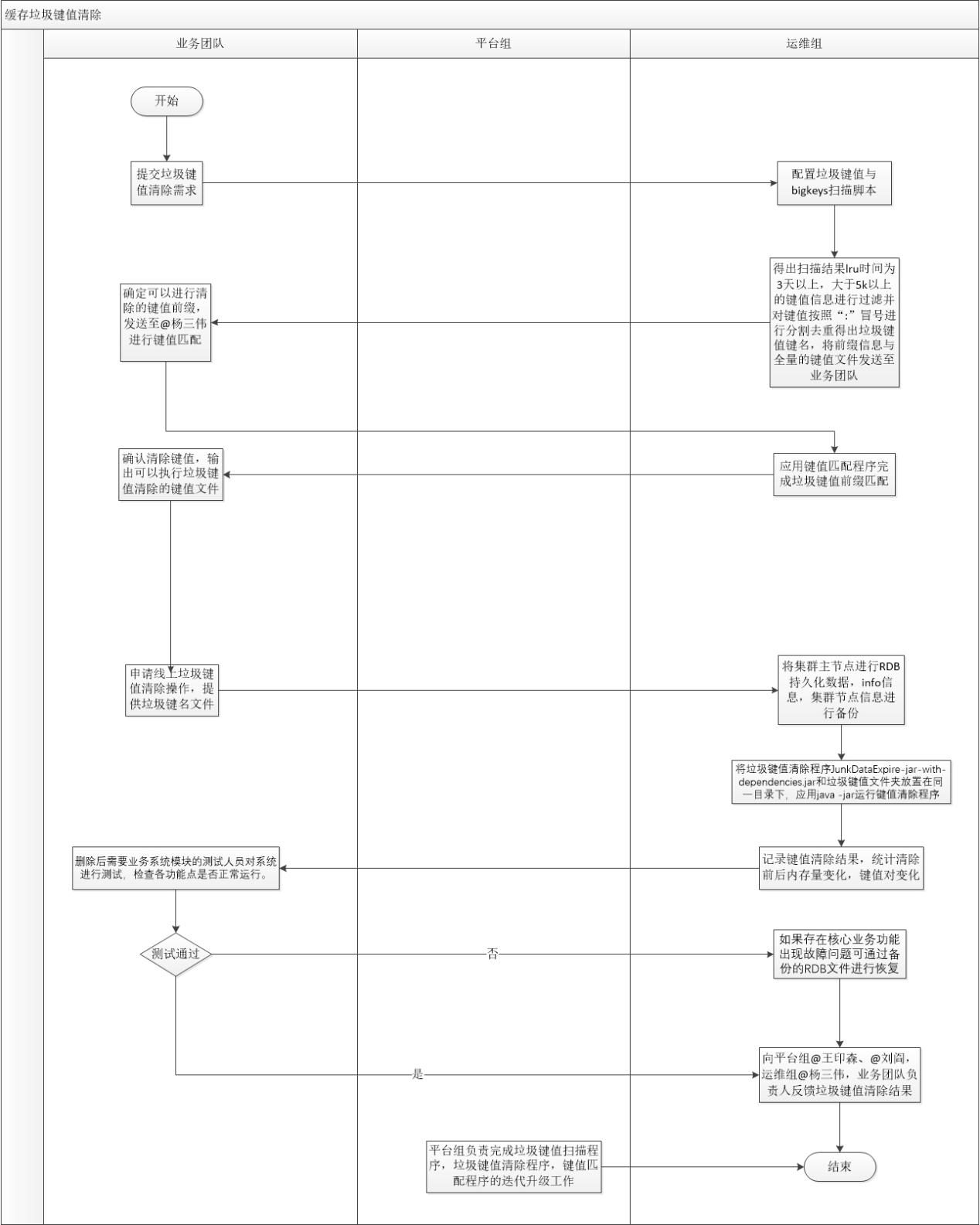
# 运维流程规范

## 接入流程



1. 业务参照《中移在线Redis开发规范》应用附件10.2《中移在线缓存应用场景统计》对缓存应用场景进行梳理，规范缓存应用逻辑，明确缓存键值命名规划，将《中移在线缓存应用场景统计》表发送邮件至王印森（wangyinsen@cmos.chinamobile.com）、刘阎（liuyan5@cmos.chinamobile.com）进行评审。
2. 平台组评审通过后，业务侧需明确Redis集群内存量大小，集群节点数量，部署网络域信息，输出《Redis集群部署信息》表发送至运维组杨三伟（yangsanwei@cmos.chinamobile.com），附上平台组审核意见；如果评审未通过则按照要求进行整改。
3. 运维人员依据《Redis集群部署信息》划分测试，生产环境机器资源，域名分配，密码分配，打通网络，完成开发，测试，生产环境集群部署并回复邮件。
4. 测试组需要上线发布环境对业务测试环境缓存集群进行缓存容错机制测试，即对缓存集群数据清空业务主流程能够正常运行。通过测试后测试组将测试环境Redis集群地址发送至平台组王印森、刘阎，完成缓存运行态键值扫描工作。
5. 平台组应用缓存键值巡检工具对业务团队测试集群进行扫描，内容包括：垃圾键值数、bigkeys键值数、过期键值占比信息进行扫描并反馈是否符合缓存应用上线标准。（垃圾键值数占比不超过5%，bigkeys键值数占比为0，过期键值占比不低于95%）
6. 测试环境审核通过，完成生产环境上线发布申请，附上平台组审核意见
7. 运维组将生产环境集群录入缓存集群进行纳管
8. 平台组每周对生产环境集群进行健康巡检，输出《Redis缓存中间件运行周报》
9. 迭代升级工作中，业务侧需要新增《中移在线缓存应用场景》，平台组完成审核，测试完成宕机容错测试后提交运维完成生产上线发布。

## 垃圾键值清除流程



1. 业务侧提交垃圾键值清除需求，发送邮件至@杨三伟（yangsanwei@cmos.chinamobile.com）明确集群地址
2. 运维侧配置垃圾键值与bigkeys扫描脚本，得出扫描结果（lru时间为3天以上，大于5k以上的键值信息进行过滤并对键值按照“:”冒号进行分割去重得出垃圾键值键名），将前缀信息与全量的键值文件发送至业务团队。
3. 业务侧确定可以进行清除的键值前缀发送至@杨三伟进行键值匹配
4. 运维人员应用键值匹配程序完成垃圾键值前缀匹配
5. 业务侧确认清除键值，输出可以执行垃圾键值清除的键值文件。
6. 业务侧申请线上垃圾键值清除操作，提交垃圾键值名文件
7. 运维人员将集群主节点进行RDB持久化数据，info信息，集群节点信息进行备份
8. 运维人员将垃圾键值清除程序JunkDataExpire-jar-with-dependencies.jar和垃圾键值文件夹放置在同一目录下，应用java -jar运行键值清除程序
9. 运维人员记录键值清除结果，统计清除前后内存量变化，键值对变化
10. 删除后需要业务系统模块的测试人员对系统进行测试，检查各功能点是否正常运行。
11. 如果测试通过则有运维操作人员反馈本次键值键值清除结果；如果测试不通过则应用RDB持久化文件进行恢复。
12. 平台组负责完成垃圾键值扫描程序，垃圾键值清除程序，键值匹配程序的迭代升级工作。

# 附录

## Info系统状态说明

|  |  |
| --- | --- |
| 模块名 | 模块含义 |
| Server | 服务端信息 |
| Clients | 客户端信息 |
| Memory | 内存信息 |
| Persistence | 持久化信息 |
| Stats | 全局统计信息 |
| Replication | 复制信息 |
| CPU | CPU消耗信息 |
| Comandstats | 命令统计信息 |
| Cluster | 集群信息 |
| Keyspace | 数据库键统计信息 |

### info Server 模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| redis\_version | 3.0.7 | Redis服务版本 |
| redis\_git\_shal | 00000000 | Git SHA 1 |
| redis\_git\_dirty | 0 | Git dirty flag |
| redis\_build\_id | 186eba9451cf9390 | Redis build id |
| redis\_mode | cluster | 运行模式，分为：Cluster、Sentinel、Standalone |
| os | Linux 2.6.18-274.e15 x86-64 | Redis所在机器的操作系统 |
| arch\_bits | 64 | 架构（32或64位） |
| multiplexing\_api | epoll | Redis所使用的事件处理机制 |
| gcc\_version | 4.1.2 | 编译Redis时所使用的GCC版本 |
| process\_id | 31524 | Redis服务进程的PID |
| run\_id | fd8b97739c469526f640b8895a5084d669ed151f | Redis服务的标识符 |
| tcp\_port | 6384 | 监听端口 |
| uptime\_in\_seconds | 9753347 | 自Redis服务启动以来，运行的秒数 |
| uptime\_in\_days | 112 | 自Redis服务启动以来，运行的天数 |
| hz | 10 | serverCron每秒运行次数 |
| lru\_clock | 16388503 | 以分钟为单位进行自增的时钟，用于LRU管理 |
| config\_file | /opt/cachecloud/conf/redis-cluster-6384.conf | Redis的配置文件 |

### info Clinets模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | 属性描述 |
| connected\_clients | 262 | 当前客户端连接数 |
| client\_longest\_output\_list | 0 | 当前所有输出缓冲区中队列对象个数的最大值 |
| client\_biggest\_input\_buf | 0 | 当前所有输入缓冲区中占用的最大容量 |
| blocked\_clients | 0 | 正在等待阻塞命令（例如BLPOP等）的客户端数量 |

### info Memory模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| used\_memory | 183150904 | Redis分配器分配的内存总量，也就是内部存储的所有数据内存占用量 |
| used\_memory\_human | 174.67M | 以可读的格式返回used\_memory |
| used\_memory\_res | 428621824 | 从操作系统的角度，Redis进程占用的物理内存总量 |
| used\_memory\_peak | 522768352 | 内存使用的最大值，表示used\_memory的峰值 |
| used\_memory\_peak\_human | 498.55M | 以可读的格式返回used\_memory\_peak |
| used\_memory\_lua | 35840 | Lua引擎锁消耗的内存大小 |
| mem\_fragmentation\_ratio | 2.34 | used\_memory\_rss/used\_memory比值，表示内存碎片率 |
| mem\_allocator | jemalloc-3.6.0 | Redis所使用的内存分配器。默认为：jemalloc |

### info Persistence模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| loading | 0 | 是否在加载持久化文件。0否，1是 |
| rdb\_changes\_since\_last\_save | 53308858 | 自上次RDB后，Redis数据改动条数 |
| rdb\_bgsave\_in\_progress | 0 | 标识RDB的bgsave操作是否进行中。0否，1是 |
| rdb\_last\_save\_time | 1456376460 | 上次bgsave操作的时间戳 |
| rdb\_last\_bgsave\_status | ok | 上次bgsave操作状态 |
| rdb\_last\_bgsave\_time\_sec | 3 | 上次bgsave操作使用的时间（单位是秒） |
| rdb\_current\_bgsave\_time\_rec | -1 | 如果bgsave操作正在进行，则记录当前bgsave操作使用的时间（单位是秒） |
| aof\_enabled | 1 | 是否开启了AOF功能。0否，1是 |
| aof\_rewrite\_in\_progress | 0 | 标识AOF的rewrite操作是否在进行中。0否，1是 |
| aof\_rewrite\_scheduled | 0 | 标识是否将要在RDB的bgsave操作结束后执行AOF rewrite操作 |
| aof\_last\_rewrite\_time\_sec | 0 | 上次AOF rewrite操作使用的时间（单位是秒） |
| aof\_current\_rewrite\_time\_sec | -1 | 如果rewrite操作正在进行，则记录当前AOF rewrite所使用的时间（单位是秒） |
| aof\_last\_bgrewrite\_status | ok | 上次AOF重写操作的状态 |
| aof\_last\_write\_status | ok | 上次AOF写磁盘的结果 |
| aof\_current\_size | 186702421 | AOF当前尺寸（单位是字节） |
| aof\_base\_size | 134279710 | AOF上次启动或rewrite的尺寸（单位是字节） |
| aof\_buffer\_length | 0 | AOF buffer的大小 |
| aof\_rewrite\_buffer\_length | 0 | AOF rewrite buffer的大小 |
| aof\_pending\_bio\_fsync | 0 | 后台IO队列中等待fsync任务的个数 |
| aof\_delayed\_fsync | 64 | 延迟的fsync计数器 |

### info Stats模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| total\_connections\_received | 495967 | 连接过的客户端总数 |
| total\_commands\_processed | 5139857171 | 执行过的命令总数 |
| instantaneous\_ops\_per\_sec | 511 | 每秒处理命令条数 |
| total\_net\_input\_bytes | 2.82961E+11 | 输入总网络流量（以字节为单位） |
| total\_net\_output\_bytes | 1.7605E+12 | 输出总网络流量（以字节为单位） |
| instantaneous\_input\_kbps | 28.24 | 每秒输入字节数 |
| instantaneous\_output\_kbps | 234.9 | 每秒输出字节数 |
| rejected\_connections | 0 | 拒绝连接的个数 |
| sync\_full | 4 | 主从完全同步成功次数 |
| sync\_partial\_ok | 0 | 主从部分同步成功次数 |
| sync\_partial\_err | 0 | 主从部分同步失败次数 |
| expired\_keys | 45534039 | 过期的key数量 |
| evicted\_keys | 0 | 剔除（超过了maxmemory后）的key数量 |
| keyspace\_hits | 3923837939 | 命中次数 |
| keyspace\_misses | 1078922155 | 不命中次数 |
| pubsub\_channels | 0 | 当前使用中的频道数量 |
| pubsub\_patterns | 0 | 当前使用中的模式数量 |
| latest\_fork\_usec | 16194 | 最近一次fork操作消耗的时间（微秒） |
| migrate\_cached\_sockets | 0 | 记录当前Redis正在进行migrate操作的目标Redis个数。例如Redis A分别向Redis B和C执行migrate操作，那么这个值就是2 |

### info Replication模块统计信息

|  |  |  |
| --- | --- | --- |
| **角色** | **属性名** | **属性值** |
| 通用配置 | role | master|slave |
| 主节点 | connected\_slaves | 1 |
| slave0 | slave0:ip=10.10.xx.160,port=6382,state=online,offset=426978948465,lag=1 |
| master\_repl\_offset | 4.26979E+11 |
| 从节点 | master\_host | 10.10.xx.63 |
| master\_port | 6387 |
| master\_link\_status | up |
| master\_last\_io\_seconds\_ago | 0 |
| master\_sync\_in\_progress | 0 |
| slave\_repl\_offset | 4.26979E+11 |
| slave\_priority | 100 |
| slave\_read\_only | 1 |
| connected\_slaves | 0 |
| master\_repl\_offset | 0 |
| 通用配置 | repl\_backlog\_active | 1 |
| repl\_backlog\_size | 10000000 |
| repl\_backlog\_first\_byte\_offset | 4.26969E+11 |
| repl\_backlog\_histlen | 10000000 |

### info cpu模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| used\_cpu\_sys | 31957.3 | Redis主进程在内核状态所占用的CPU时钟总和 |
| used\_cpu\_user | 72484.27 | Redis主进程在用户状态所占用的CPU时钟总和 |
| used\_cpu\_sys\_children | 121.49 | Redis子进程在内核状态所占用的CPU时钟总和 |
| used\_cpu\_user\_children | 195.13 | Redis子进程在用户状态所占用的CPU时钟总和 |

### info Commandstas模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| cmdstat\_get | calls=3738730699,usec=11054972404,usec\_per\_call=2.96 | get命令调用总次数、总耗时、平均耗时（单位：毫秒） |
| cmdstat\_set | calls=50174458,usec=323143686,usec\_per\_call=6.44 | set命令调用总次数、总耗时、平均耗时（单位：毫秒） |

### info Cluster模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| cluster\_enabled | 1 | 节点是否为cluster模式。1是，0否 |

### info Keyspace模块统计信息

|  |  |  |
| --- | --- | --- |
| **属性名** | **属性值** | **属性描述** |
| db0 | dbo:keys=106430,expires=56107,avg\_ttl=60283952 | 当前数据库key总数，带有过期时间的key总数，平均存活时间 |

## Redis配置说明

### 总体配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| daemonize | 是否是守护进程 | no | yes|no | 不可以 |
| port | 端口号 | 6379 | 整数 | 不可以 |
| loglevel | 日志级别 | notice | debug|verbose|notice|warning | 可以 |
| logfile | 日志文件名 | 空 | 自定义，建议以端口号为名 | 不可以 |
| databases | 可用的数据库数 | 16 | 整数 | 不可以 |
| unixsocket | unix套接字 | 空（不通过unix套接字来监听） | 指定套接字文件 | 不可以 |
| unixsocketperm | unix套接字权限 | 0 | Linux三位数权限 | 不可以 |
| pidfile | Redis运行的进程pid文件 | /var/run/redis.pid | /var/run/redis-{port}.id | 不可以 |
| lua-time-limit | Lua脚本“超时时间”（单位：毫秒） | 5000 | 整数，但是此超时不会真正停止脚本运行，具体参考第3章 | 可以 |
| tcp-backlog | tcp-backlog | 511 | 整数 | 不可以 |
| watchdog-period | 看门狗，用于诊断Redis的延迟问题，此参数是检查周期。（此参数需要在运行时配置才能生效） | 0 | 整数 | 可以 |
| activerehashing | 制定是否激活重置哈希 | yes | yes|no | 可以 |
| dir | 工作目录（aof、rdb、日志文件都存放在此目录） | ./(当前目录) | 自定义 | 可以 |

### 最大内存及策略

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| maxmemory | 最大可用内存（单位字节） | 0（没有限制） | 整数 | 可以 |
| maxmemory-policy | 内存不够时，淘汰策略 | noeviction | volatile-lru->用lru算法删除过期的键值  allkeys-lru->用lru算法删除所有键值 volatile-random->随机删除过期的键值  allkeys-random->随机删除任何键值 volatile-ttl->删除最近要到期的键值 noeviction->不删除键 | 可以 |
| maxmemory-samples | 检测LRU采样数 | 5 | 整数 | 可以 |

### AOF相关配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| appendonly | 是否开启AOF持久化模式 | no | no|yes | 可以 |
| appendfsync | AOF同步磁盘频率 | everysec | always|everysec|no | 可以 |
| appendfilename | AOF文件名 | appendonly.aof | appendonly-{port}.aof | 不可以 |
| aof-load-truncated | 加载AOF文件时，是否忽略AOF文件不完整的情况，让Redis正常启动 | yes | yes|no | 可以 |
| no-appendfsync-on-rewrite | 设置为yes表示rewrite期间对新写操作不fsync，暂时存在缓冲区中，等rewrite完成后再写入 | no | no|yes | 可以 |
| auto-aof-rewrite-min-size | 触发rewrite的AOF文件最小阀值（单位：兆） | 64m | 整数+m(代表兆） | 可以 |
| auto-aof-rewrite-percentage | 触发rewrite的AOF文件的增长比例条件 | 100 | 整数 | 可以 |
| aof-rewrite-incremen-tal-fsync | AOF重写过程中，是否采取增量文件同步策略 | yes | yes|no | 可以 |

### RDB相关配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| save | RDB保存条件 | save 900 1  save 300 10  save 60 10000 | 如果没有该配置，代表不使用自动RDB策略 | 可以 |
| dbfilename | RDB文件名 | dump.rdb | dump-{port}.rdb | 可以 |
| rdbcompression | RDB文件是否压缩 | yes | yes|no | 可以 |
| rdbchecksum | RDB文件是否使用校验和 | yes | yes|no | 可以 |
| stop-writes-on-bgsave-error | bgsave执行错误，是否停止Redis接受请求 | yes | yes|no | 可以 |

### 慢查询相关配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| slowlog-log-slower-than | 慢查询被记录的阀值（单位微秒 | 10000 | 整数 | 可以 |
| slowlog-max-len | 最多记录慢查询的条数 | 128 | 整数 | 可以 |
| latency-monitor-threshold | Redis服务内存延迟监控 | 0（关闭） | 整数 | 可以 |

### 数据结构优化配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| hash-max-ziplist-entries | hash数据结构优化参数 | 512 | 整数 | 可以 |
| hash-max-ziplist-value | hash数据结构优化参数 | 64 | 整数 | 可以 |
| list-max-ziplist-entries | list数据结构优化参数 | 512 | 整数 | 可以 |
| list-max-ziplist-value | list数据结构优化参数 | 64 | 整数 | 可以 |
| set-max-intset-entries | set数据结构优化参数 | 512 | 整数 | 可以 |
| zset-max-ziplist-entries | zset数据结构优化参数 | 128 | 整数 | 可以 |
| zset-max-ziplist-value | zset数据结构优化参数 | 64 | 整数 | 可以 |
| hll-sparse-max-bytes | HyperLogLog数据结构优化参数 | 3000 | 整数 | 可以 |

### 复制相关配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| slaveof | 指定当前从节点复制哪个主节点，参数：主节点的ip和port | 空 | ip和端口 | 不可以，但可以用slaveof命令设置 |
| repl-ping-slave-period | 主节点定期向从节点发送ping命令的周期，用于判定从节点是否存活。（单位：秒） | 10 | 整数 | 可以 |
| repl-timeout | 主从节点复制超时时间（单位：秒） | 60 | 整数 | 可以 |
| repl-backlog-size | 复制积压缓存区大小 | 1M | 整数 | 可以 |
| repl-backlog-ttl | 主节点在没有从节点的情况下多长时间后释放复制积压缓存区空间 | 3600 | 整数 | 可以 |
| slave-priority | 从节点的优先级 | 100 | 0-100 | 可以 |
| min-slaves-to-write | 当主节点发现从节点数量小于min-slaves-to-write且延迟小于等于min-slaves-max-lag时，master停止写入操作 | 0 | 整数 | 可以 |
| min-slaves-max-lag | 10 | 整数 | 可以 |
| slave-serve-stale-data | 当从节点与主节点连接中断时，如果此参数值设置为“yes”，从节点可以继续处理客户端的请求。否则除info和slaveof命令之外，拒绝的所有请求并统一回复"SYNC with master in progress" | yes | yes|no | 可以 |
| slave-read-only | 从节点是否开启只读模式，集群架构下从节点默认读写都不可用，需要调用readyonly命令开启只读模式 | yes | yes|no | 可以 |
| repl-disable-tcp-nodelay | 是否开启主从复制socket的NO\_DELAY选项： yes:Redis会合并小的TCP包来节省带宽，但是这样增加同步延迟，造成主从数据不一致 no:主节点会立即发送同步数据，没有延迟 | no | yes|no | 可以 |
| repl-diskless-sync | 是否开启无盘复制 | no | yes|no | 可以 |
| repl-diskless-sync-delay | 开启无盘复制后，需要延迟多少秒后进行创建RDB操作，一般用于同时加入多个从节点时，保证多个从节点可共享RDB | 5 | 整数 | 可以 |

### 客户端相关配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| maxclients | 最大客户端连接数 | 10000 | 整数 | 可以 |
| client-output-buffer-limit | 客户端输出缓冲区限制 | normal 0 0 0  slave 268435456 67108864 60  pubsub 33554432 8388608 60 | 整数 | 可以 |
| timeout | 客户端闲置多少秒后关闭连接（单位：秒） | 0（永不关闭） | 整数 | 可以 |
| tcp-keepalive | 检测TCP连接活性的周期（单位：秒） | 0（不检测） | 整数 | 可以 |

### 安全相关配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| requirepass | 密码 | 空 | 自定义 | 可以 |
| bind | 绑定IP | 空 | 自定义 | 不可以 |
| masterauth | 从节点需要配置的主节点密码 | 空 | 主节点的密码 | 可以 |

### cluster相关配置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **参数名** | **含义** | **默认值** | **可选值** | **可否支持config set配置热生效** |
| cluster-node-timeout | 集群节点超时时间（单位：毫秒) | 15000 | 整数 | 可以 |
| cluster-migration-barrier | 主从节点切换需要的从节点数最小个数 | 1 | 整数 | 可以 |
| cluster-slave-validity-factor | 从节点有效性判断因子，当从节点与主节点最后通信时间超过（cluster-node-timeout\*slave-validity-factor)+repl-ping-slave-period时，对应从节点不具备故障转移资格，防止断线时间过长的从节点进行故障转移。设置为0标识从节点永不过期 | 10 | 整数 | 可以 |
| cluster-require-full-converage | 集群是否需要所有的slot都分配给在线节点，才能正常访问 | yes | yes|no | 可以 |
| cluster-enabled | 是否开启集群模式 | yes | yes|no | 不可以 |
| cluster-config-file | 集群配置文件名称 | nodes.conf | nodes-{port}.conf | 不可以 |

### Sentinel配置说明和分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **参数名** | **含义** | **默认值** | **可选值** | **可否支持sentinel set配置热生效** |
| sentinel monitor <master-name> <ip> <port> <quorum> | 定义监控的主节点名、ip、port、主观下线票数 | sentinel monitor mymaster 127.0.0.1.6379 2 | 自定义 masterName实际的ip：port票数 | 支持<quorum> |
| sentinel down-after-milliseconds <master-name> <times> | Sentinel判定节点不可达的毫秒数 | sentinel down-after-milliseconds mymaster 30 000 | 整数 | 支持 |
| sentinel parallel-syncs <master-name> <nums> | 在执行故障转移时，最多有多少个从服务器同时对新的主服务器进行同步 | sentinel parallel-syncs mymaster 1 | 大于0，不超过从服务器个数 | 支持 |
| sentinel failover-timeout <master-name> <times> | 故障迁移超时时间 | sentinel failover-timeout mymaster 180 000 | 整数 | 支持 |
| sentinel auth-pass <mas-ter-name> <script-path> | 主节点密码 | 空 | 主节点密码 | 支持 |
| sentinel notifi-cation-script <mas-ter-name> <script-path> | 故障转移期间脚本通知 | 空 | 脚本文件路径 | 支持 |
| sentinel client-reconfig-script <mas-ter-name> <script-path> | 故障转移成功后脚本通知 | 空 | 脚本文件路径 | 支持 |

# 附件

## Ruby集群创建脚本



使用方法：将脚本copy至redis/bin目录下替换redis原生的redis集群创建脚本。

操作示例：

#!/bin/bash

export PATH=/home/itframe/ruby/bin:.:$PATH

./bin/redis-trib.rb create-manual G 192.168.100.144:8000 192.168.100.146:8003 G 192.168.100.146:8002 192.168.100.152:8005 G 192.168.100.152:8004 192.168.100.144:8001

echo "success...."

exit 0

redis-trib.rb create-manual G 192.168.100.144:8000 192.168.100.146:8003 G 192.168.100.146:8002 192.168.100.152:8005 G 192.168.100.152:8004 192.168.100.144:8001主从关系为：

192.168.100.146:8003 slaveof 192.168.100.144:8000

192.168.100.152:8005 slaveof 192.168.100.146:8002

192.168.100.144:8001 slaveof 192.168.100.152:8004

PS：

G代表一个分组，G后面第一个节点信息为主节点，第一个节点信息至下一个“G”符号标识的节点信息为改主节点的从节点（一个主节点可以配置多个从节点）

## 缓存应用场景统计

