OneProxy for Cache

http://www.onexsoft.com

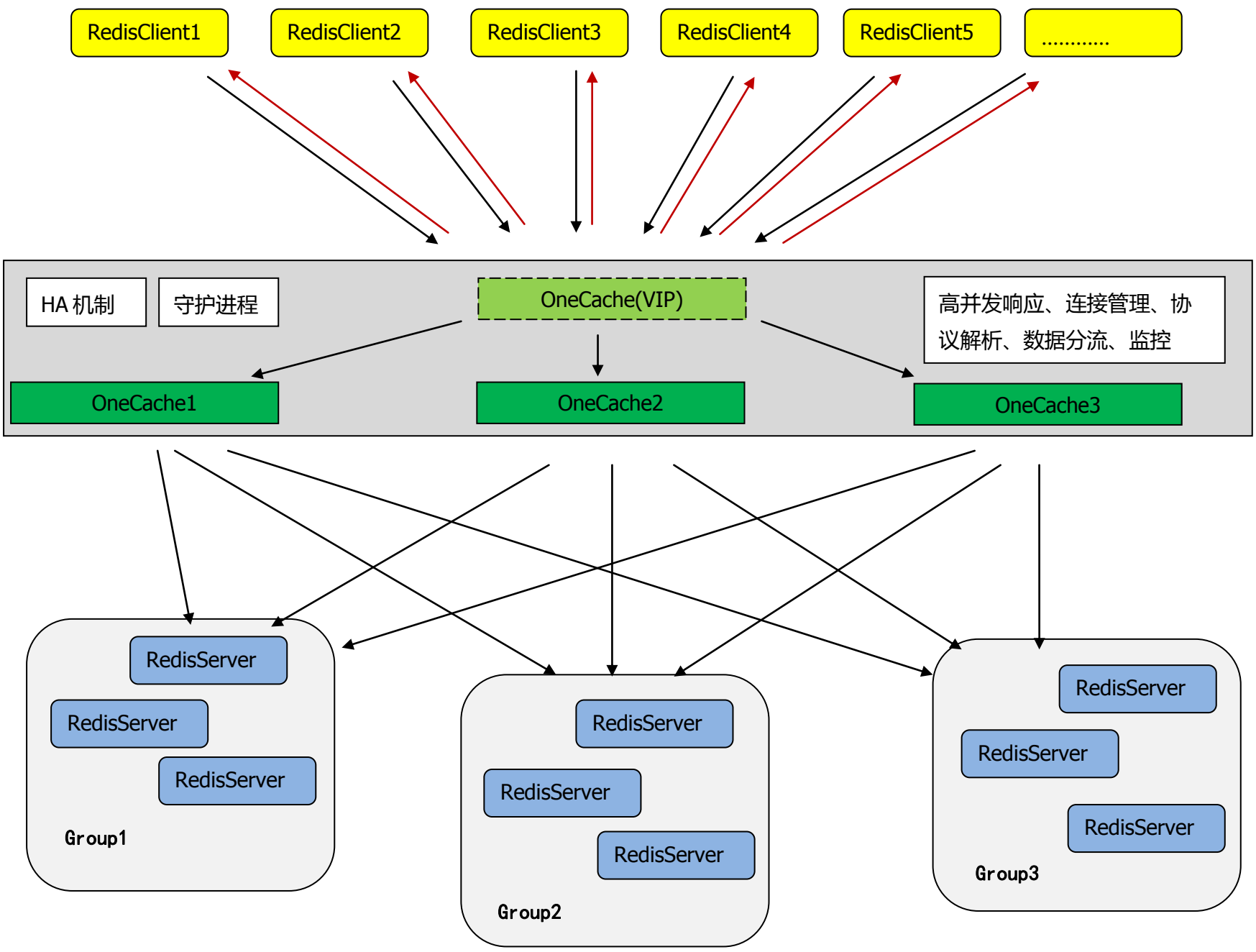
技术交流 QQ 群：324460822

- OneCache — 针对 Redis 的代理服务

功能特性

- **高并发性**
  1. 测试时使用了多个 redis-benchmark 对 onecache 进行测试，每秒并发量可达 40w+/s
- **完整的 Redis 命令支持**
  1. 支持大部分 redis 命令
  2. 自带内置命令。例如查看当前状态、所支持的命令等等
- **兼容 twemproxy**
  1. 原先使用 twemproxy 的应用现在可以迁移到 onecache 上
- **通过代理的方式减少缓存服务器的连接数**
  1. 保持与 Redis 的长连接
  2. 连接可复用性
  3. 可设置代理与每个 Redis 的连接数目
- **RedisServerGroup 模型**
  1. Group 内可配置若干 Master 和 Slave，支持不同的策略模式
  2. 每个 Group 可以动态配置散列范围、可以配置特定的 KEY 路由到该 Group
- **自动分片到后端多个 Redis 实例上**
  1. 使用散列方式将数据以 Key 的值进行散列化
  2. 可以控制后端实例的权重
- **详细的状态监控**
  1. 实时代理运行状态
  2. 实时后端节点运行状态
  3. 实时客户端连接状态
  4. TopKey 统计
  5. 连接池状态查看
- **高可用性**
  1. 可部署多个 OneCache 以防止单台主机故障，使用 VIP 方式实现
  2. Group 不可用时可以自动移除，采用一致性 HASH 算法调整 HASH 槽

软件模型


RedisClient1
RedisClient2
RedisClient3
RedisClient4
RedisClient5
.....
HA 机制
守护进程
OneCache(VIP)
高并发响应、连接管理、协议解析、数据分流、监控
OneCache1
OneCache2
OneCache3
RedisServer
RedisServer
RedisServer
Group1
RedisServer
RedisServer
RedisServer
Group2
RedisServer
RedisServer
RedisServer
Group3

软件安装与配置

1. 下载软件。软件只有一个可执行文件 onecache

2. 启动软件。软件启动只依赖一个配置文件（XML 格式）。命令格式： onecache [cfg-file] 。如果未指定配置文件路径名，默认会以当前目录下的 onecache.xml 做为配置，如果配置参数中有不正确的地方，软件将会报错，具体出错原因可以查看输出的日志信息。

这是一个典型的配置文件格式

```
<onecache port="8221" thread_num="15" hash_value_max="80" daemonize="0" guard="0" log_file="" password="" pid_file="" hash=" fnv1a_64"
    twemproxy_mode="0" debug="0">
    <!--port 运行端口-->
    <!--thread_num 线程数-->
    <!--hash_value_max 哈希槽个数(最大不得超过 1024)-->
    <!--daemonize 表示是否后台运行 1=YES 0=NO-->
    <!--guard 表示是否启用守护进程 1=YES 0=NO-->
    <!--log_file 表示输出的日志文件路径-->
    <!--password onecache 密码，客户端需要 auth 命令进行验证-->
     <!--pid_file pid 文件路径 -->
     <!--hash hash 方法名称，可以为空 -->
     <!—twemproxy_mode 是否按 twemproxy 模式运行 注：只支持 ketama 方式，groupname 对应 servername-->
     <!—debug 是否 debug 模式运行 1=YES 0=NO debug 模式将会得到更详细的运行日志，注：打印日志可能会很多，建议生产线上不要开启-->

    <vip if_alias_name="eth0:0" vip_address="172.30.12.8" enable="0"></vip>
    <!--VIP 配置 if_alias_name 表示适配器别名 vip_address 表示虚拟地址 enable 表示是否启用 VIP 功能 1:启用 0:禁用-->
    <top_key enable=" 0"></top_key>
    <!--是否启用 TopKey 统计功能 1:启用 0:禁用-->

    <group_option backend_retry_interval="3" backend_retry_limit="10" auto_eject_group="1" group_retry_time="30"
eject_after_restore="1"></group_option>
    <!--backend_retry_interval 表示后端断开后的重试连接的间隔时间-->
    <!--backend_retry_limit 表示后端重试连接的最大次数-->
    <!--auto_eject_group 表示是否启用 Group 不可用时自动移除 1=YES 0=NO-->
    <!--group_retry_time 表示 Group 的重试时间，超过后将会自动移除-->
    <!--eject_after_restore 表示摘除 Group 后，如果又变为可用状态，将会进行恢复 1=YES 0=NO-->

    <group name="group1" hash_min="0" hash_max="19" policy="master_only">
    <!--组名为 group1 哈希映射的范围为 0~19 (包含 0,19) 使用的策略为 master_only-->
        <host host_name="host1" ip="172.30.12.12" port="6379" master="1" password="" connect_num="50"></host>
        <host host_name="host1" ip="172.30.12.12" port="6380" master="0" password="" connect_num="50"></host>
        <!--host_name 表示主机别名-->
        <!--ip Redis 服务器 IP-->
        <!--port Redis 服务器端口-->
        <!--master 是否主备 1:主 0:备-->
        <!--password 表示 redis 服务器的验证密码-->
        <!—connect_num 表示连接到 redis 服务器的连接池大小-->
    </group>
    <group name="group2" hash_min="20" hash_max="39">
        <host host_name="host1" ip="172.30.12.12" port="6381" master="1"></host>
    </group>
    <group name="group3" hash_min="40" hash_max="59">
        <host host_name="host1" ip="172.30.12.12" port="6382" master="1"></host>
    </group>
    <group name="group4" hash_min="60" hash_max="79">
        <host host_name="host1" ip="172.30.12.12" port="6383" master="1"></host>
    </group>

    <!--配置 HASH 值到 Group 的映射关系-->
    <hash_mapping>
        <hash value="0" group_name="group1"></hash>
        <hash value="1" group_name="group1"></hash>
    </hash_mapping>
    <!--配置指定 KEY 到 Group 的映射关系-->
    <key_mapping>
        <key key_name="key1" group_name="group1"></key>
        <key key_name="key2" group_name="group2"></key>
    </key_mapping>
</onecache>
```

性能测试

测试环境描述

机器类型：物理机(16 Intel(R) Xeon(R) CPU E5620 @ 2.40GHz 32G 内存 千兆网卡)

测试产品：Twemproxy, OneCache

性能测试工具：redis-benchmark

机器分布：Twemproxy 和 OneCache 都运行在同一台机器上，测试机为独立的一台，redis 实例也分布在独立的机器上

测试方案 1：单个 redis-benchmark 进行测试，对比经过代理和直连的差异性

Twemproxy 的启动配置

```
[root@sukeme bin]# cat nutcracker-1.yml
alpha:
  listen: 0.0.0.0:22121
  hash: fnv1a_64
  distribution: ketama
  auto_eject_hosts: true
  redis: true
  server_retry_timeout: 2000
  server_failure_limit: 1
  servers:
   - 172.30.12.12:6379:1 group1
```

OneCache 的启动配置

```
[root@sukeme bin]# cat onecache-1.xml
<onecache port="8221" thread_num="12" hash_value_max="80">
  <group name="group1" hash_min="0" hash_max="79" policy="master_only">
     <host host_name="host1" ip="172.30.12.12" port="6379" master="1"></host>
  </group>
</onecache>
```

直连 Redis 的测试结果

```
[root@osd1 redis]# ./redis-benchmark -h 172.30.12.12 -p 6379 -t set,get -r 1000000 -n 1000000 -q
SET: 94268.48 requests per second
GET: 96655.72 requests per second
```

通过 Twemproxy 的测试结果

```
[root@osd1 redis]# ./redis-benchmark -h 172.30.12.15 -p 22121 -t set,get -r 1000000 -n 1000000 -q
SET: 52394.43 requests per second
GET: 60945.88 requests per second
```

通过 OneCache 的测试结果

```
[root@osd1 redis]# ./redis-benchmark -h 172.30.12.15 -p 8221 -t set,get -r 1000000 -n 1000000 -q
SET: 87168.76 requests per second
GET: 92532.62 requests per second
```

结论： 通过测试结果可以看出，经过 Twemproxy 有一定的损失，但 OneCache 损失却很小。因为 OneCache 内部采用了多线程方式进行网络通信，提高了连接可复用性以及网络 IO 的利用率。

测试方案 2：多个 redis-benchmark 并行测试，测试两者最大并发量。

因为 redis-benchmark 是单线程方式测试的，单个测试并不能反映实际的并发量。

现在 OneCache 和 Twemproxy 后端各挂 8 个 Redis 实例，redis-benchmark 同时启动 10 个进程，通过此方式来对比两者每秒的并发量。

Twemproxy 启动配置

```
[root@sukeme bin]# cat nutcracker.yml
alpha:
  listen: 0.0.0.0:22121
  hash: fnv1a_64
  distribution: ketama
  auto_eject_hosts: true
  redis: true
  server_retry_timeout: 2000
  server_failure_limit: 1
  servers:
   - 172.30.12.12:6379:1 group1
   - 172.30.12.12:6380:1 group2
   - 172.30.12.12:6381:1 group3
   - 172.30.12.12:6382:1 group4
   - 172.30.12.10:6379:1 group5
   - 172.30.12.10:6380:1 group6
   - 172.30.12.10:6381:1 group7
   - 172.30.12.10:6382:1 group8
```

Twemproxy 测试脚本

```
[root@osd1 redis]# cat 22121.sh
#!/bin/bash
#
for i in {1..10}
do
 nohup ./redis-benchmark -h 172.30.12.15 -p 22121 -t set,get -r 1000000 -n 1000000 -q > redis_test_${i}.log 2>&1 &
done
```

OneCache 启动配置

```
[root@sukeme bin]# cat onecache.xml
<onecache port="8221" thread_num="12" hash_value_max="80">
   <group name="group1" hash_min="0" hash_max="9" policy="master_only">
     <host host_name="host1" ip="172.30.12.12" port="6379" master="1"></host>
  </group>
  <group name="group2" hash_min="10" hash_max="19" policy="master_only">
     <host host_name="host2" ip="172.30.12.12" port="6380" master="1"></host>
  </group>
  <group name="group3" hash_min="20" hash_max="29" policy="master_only">
     <host host_name="host3" ip="172.30.12.12" port="6381" master="1"></host>
  </group>
  <group name="group4" hash_min="30" hash_max="39" policy="master_only">
     <host host_name="host3" ip="172.30.12.12" port="6382" master="1"></host>
  </group>
  <group name="group5" hash_min="40" hash_max="49" policy="master_only">
     <host host_name="host1" ip="172.30.12.10" port="6379" master="1"></host>
  </group>
  <group name="group6" hash_min="50" hash_max="59" policy="master_only">
     <host host_name="host2" ip="172.30.12.10" port="6380" master="1"></host>
  </group>
  <group name="group7" hash_min="60" hash_max="69" policy="master_only">
     <host host_name="host3" ip="172.30.12.10" port="6381" master="1"></host>
  </group>
  <group name="group8" hash_min="70" hash_max="79" policy="master_only">
     <host host_name="host3" ip="172.30.12.10" port="6382" master="1"></host>
  </group>
</onecache>
```

OneCache 测试脚本

```
[root@osd1 redis]# cat 8221.sh
#!/bin/bash
#
for i in {1..10}
do
 nohup ./redis-benchmark -h 172.30.12.15 -p 8221 -t set,get -r 1000000 -n 1000000 -q > redis_test_${i}.log 2>&1 &
done
```

Twemproxy 测试结果

```
[root@osd1 redis]# bash 22121.sh
[root@osd1 redis]# cat *.log
SET: 14814.38 requests per second
GET: 14842.74 requests per second

SET: 14799.69 requests per second
GET: 14853.54 requests per second

SET: 14817.01 requests per second
GET: 14844.28 requests per second

SET: 14820.52 requests per second
GET: 14842.96 requests per second

SET: 14799.03 requests per second
GET: 14853.10 requests per second

SET: 14801.44 requests per second
GET: 14850.46 requests per second

SET: 14815.03 requests per second
GET: 14846.05 requests per second

SET: 14805.60 requests per second
GET: 14851.56 requests per second

SET: 14806.92 requests per second
GET: 14848.91 requests per second

SET: 14798.81 requests per second
GET: 14853.54 requests per second
```

OneCache 测试结果

```
[root@osd1 redis]# bash 8221.sh
[root@osd1 redis]# cat *.log
SET: 49463.32 requests per second
GET: 50599.61 requests per second

SET: 50180.65 requests per second
GET: 51706.31 requests per second

SET: 49207.75 requests per second
GET: 51419.17 requests per second

SET: 49399.79 requests per second
GET: 51284.68 requests per second

SET: 48555.48 requests per second
GET: 48911.71 requests per second

SET: 49212.60 requests per second
GET: 51432.39 requests per second

SET: 50067.59 requests per second
GET: 51077.74 requests per second

SET: 50037.53 requests per second
GET: 49544.20 requests per second

SET: 46377.89 requests per second
GET: 51948.05 requests per second

SET: 48162.60 requests per second
GET: 50311.93 requests per second
```

结论： 通过测试结果可以看出，都采用 10 个进程进行测试， Twemproxy 差不多是 15W/S 左右，而 OneCache 已经到达了 50W/S。由此可以对比出 OneCache 的高并发性能远远超过了 Twemproxy。

软件自带命令

启动软件后，可以使用 Redis 自带的客户端去连接代理，连接成功后跟正常访问 Redis 服务器一样。但 OneCache 并不是完全支持了所有的 Redis 命令（可以使用 SHOWCMD 查看所支持的命令）。下表是目前软件自带的命令。

命令类型	命令解释
SHOWCMD	显示 onecache 支持的命令，上方是支持的 redis 命令，下方是自带命令 172.30.12.15:8221> showcmd *** Support the command *** [Redis Command] AUTH APPEND BITCOUNT BITPOS DUMP DEL DECR DECRBY EXPIREAT EXISTS EXPIRE GET GETBIT GETRANGE GETSET HSET HSETNX HMSET HGET HMGET HINCRBY HEXISTS HLEN HDEL HKEYS HVALS HGETALL HINCRBYFLOAT INCR INCRBY INCRBYFLOAT LPUSH LPUSHX LPOP LRANGE LREM LINDEX LINSERT LLEN LSET LTRIM MGET MSET PSETEX PERSIST PEXPIRE PEXPIREAT PTTL PING PFADD PFCOUNT PFMERGE RESTORE RPOP RPUSH RPUSHX SADD SMEMBERS SREM SPOP SCARD SISMEMBER SRANDMEMBER SETBIT SETRANGE STRLEN SET SETEX SETNX TTL TYPE ZADD ZRANGE ZREM ZINCRBY ZRANK ZREVRANK ZREVRANGE ZRANGEBYSCORE ZCOUNT ZCARD ZREMRANGEBYRANK ZREMRANGEBYSCORE [OneCache Command] SHOWCMD STATUS OUTPUTSTATUS TOPKEY TOPVALUE HASHMAPPING ADDKEYMAPPING DELKEYMAPPING SHOWMAPPING POOLINFO SHUTDOWN
STATUS	状态监控
OUTPUTSTATUS	将状态监控输出到文件 onecache_monitor.log
TOPKEY	查看当前 TopKey 状态，根据 KEY 的请求次数。使用方式 TOPKEY [nums] nums 表示要查看前多少个(默认 20 个)。
TOPVALUE	查看当前 TopKey 状态，根据对应 KEY 的 VALUE 大小。使用方式 TOPVALUE [nums] nums 表示要查看前多少个(默认 20 个)。
HASHMAPPING	配置 HASH 值路由到特定 Group。使用方式 HASHMAPPING [hash value] [group name]
ADDKEYMAPPING	配置特定 KEY 路由到特定 Group。 使用方式 ADDKEYMAPPING [group name] [key1] [key2]...
DELKEYMAPPING	删除特定 KEY 到 Group 的映射。 使用方式 DELKEYMAPPING [key1] [key2]...
SHOWMAPPING	显示 Group 的映射关系。
POOLINFO	查看连接池状态信息
SHUTDOWN	关闭 OneCache，可以带参数 force 强制退出（适用于开启了守护进程的场景）

提示：

1. HASHMAPPING，ADDKEYMAPPING， DELKEYMAPPING 这些操作会将当前的配置以及当前的改动信息回写到新的配置文件中，以免程序意外退出造成配置的丢失。配置文件生成的路径在当前可执行文件目录下，文件名格式为：onecache+时间戳.xml。

2. TOPKEY，TOPVALUE 需要启用 topkey 选项，启用此功能会带来一定性能损耗，如果不需要可以在配置文件中指定该功能不启用。

STATUS 命令输出的各个字段含义

字段名称	含义
[OneCache]	OneCache 全局信息
Version	软件版本
Port	软件启动所使用的端口
StartTime	OneCache 的启动时间
UpTime	OneCache 运行时间
GroupCount	后端组的个数
VipEnabled	是否启用 VIP
VipName	VIP 所使用的接口别名
VipAddress	VIP 地址
[Backends]	记录所有后端节点
GROUP	后端分组名
IP	后端 IP 地址
PORT	后端端口
MASTER	Y:主机　N:备机
ACTIVE	Y:活动状态　N:非活动状态
REQUESTS	请求数量
RECV SIZE	收到的字节数
SEND SIZE	发送的字节数
SEND>1KB	后端发送出去的包大小超过 1KB 的数量
SEND>1MB	后端发送出去的包大小超过 1MB 的数量
COMMANDS	所有的操作命令记录
[Clients]	记录所有客户端信息
NUM	客户端编号
IP	客户端 IP 地址
CONNECTS	客户端连接次数
REQUESTS	客户端请求次数
RECV>1KB	客户端接收的回复包大小超过 1KB 的数量
RECV>1MB	客户端接收的回复包大小超过 1MB 的数量
LAST CONNECT	客户端最后一次连接时间
COMMANDS	客户端所有命令记录