**INFO [section]**

以一种易于解释（parse）且易于阅读的格式，返回关于 Redis 服务器的各种信息和统计数值。

Here is the description of fields for Redis >= 2.4.

通过给定可选的参数 section ，可以让命令只返回某一部分的信息：

* **server** 部分记录了 **Redis 服务器**的信息，它包含以下域：
  + redis\_version : Redis 服务器版本
  + ~~redis\_git\_sha1 : Git SHA1~~
  + ~~redis\_git\_dirty : Git dirty flag~~
  + ~~redis\_build\_id:974a39162812208c~~
  + redis\_mode:standalone
  + os : Redis 服务器的宿主操作系统
  + arch\_bits : 架构（32 或 **64** 位）
  + multiplexing\_api : Redis 所使用的事件处理机制
  + gcc\_version : 编译 Redis 时所使用的 GCC 版本
  + **process\_id** : 服务器进程的 PID (验证Redis进程是否还活着)
  + run\_id : Redis 服务器的随机标识符（用于 Sentinel 和**集群**）
  + tcp\_port : TCP/IP 监听端口
  + uptime\_in\_seconds : 自 Redis 服务器启动以来，经过的秒数 (验证Redis服务器中途是否重启过)
  + uptime\_in\_days : 自 Redis 服务器启动以来，经过的天数
  + hz:10
  + lru\_clock : 以分钟为单位进行自增的时钟，用于 LRU 管理
  + config\_file:/usr/install/redis/conf/redis.conf
* **clients** 部分记录了已连接**客户端**的信息，它包含以下域：
  + **connected\_clients** : 已连接客户端的数量（不包括通过从属服务器连接的客户端）(查看当前服务器的压力)
  + client\_longest\_output\_list : 当前连接的客户端当中，最长的输出列表
  + client\_longest\_input\_buf : 当前连接的客户端当中，最大输入缓存
  + blocked\_clients : 正在等待**阻塞命令**（BLPOP、BRPOP、BRPOPLPUSH）的客户端的数量
* **memory** 部分记录了**服务器的内存**信息，它包含以下域：

**问题**：如果Redis使用的内存超出了可用的物理内存大小，那么Redis很可能会**被系统OOM Killer杀掉**。

* + ~~used\_memory : 由 Redis 分配器分配的内存总量，以字节（byte）为单位~~
  + **used\_memory\_human** : 以人类可读的格式返回 Redis 分配的内存总量(为**使用内存量**设定**阀值**，并设定相应的**报警机制**)
  + used\_memory\_rss : 从操作系统的角度，返回 Redis 已分配的内存总量（俗称**常驻集大小**）。这个值和 top 、 ps 等命令的输出一致。(最好以人类可读的格式返回)
  + ~~used\_memory\_peak : Redis 的内存消耗峰值（以字节为单位）~~
  + **used\_memory\_peak\_human** : 以人类可读的格式返回 Redis 的内存消耗峰值
  + used\_memory\_lua : Lua 引擎所使用的内存大小（以字节为单位）
  + **mem\_fragmentation\_ratio** : used\_memory\_rss 和 used\_memory 之间的比率
  + mem\_allocator : 在编译时指定的， Redis 所使用的**内存分配**器。可以是 libc 、 **jemalloc** 或者 tcmalloc 。

在理想情况下，used\_memory\_rss 的值应该只比 used\_memory 稍微高一点儿。

当 rss > used ，且两者的值相差较大时，表示存在（内部或外部的）**内存碎片**。

内存碎片的比率可以通过 mem\_fragmentation\_ratio 的值看出。

当 used > rss 时，表示 Redis 的部分内存被操作系统换出到交换空间了，在这种情况下，操作可能会产生明显的延迟。

Because Redis does not have control over how its allocations are mapped to memory pages, high used\_memory\_rss is often the result of a spike in memory usage.

当 Redis 释放内存时，分配器可能会，也可能不会，将内存返还给操作系统。

如果 Redis 释放了内存，却没有将内存返还给操作系统，那么 used\_memory 的值可能和操作系统显示的 Redis 内存占用并不一致。

查看 used\_memory\_peak 的值可以验证这种情况是否发生。

* **persistence** 部分记录了跟 **RDB 持久化和 AOF 持久化**有关的信息，它包含以下域：
  + loading : 一个标志值，记录了服务器是否正在载入持久化文件
  + **rdb\_changes\_since\_last\_save** : 距离最近一次成功创建持久化文件之后，经过了多少**秒**(**如果这时候出现故障，会丢失多少数据**)
  + rdb\_bgsave\_in\_progress : 一个标志值，记录了服务器是否正在创建 RDB 文件。
  + **rdb\_last\_save\_time** : 最近一次成功创建 RDB 文件的 UNIX 时间戳
  + rdb\_last\_bgsave\_status : 一个标志值，记录了最近一次创建 RDB 文件的结果是成功还是失败。
  + rdb\_last\_bgsave\_time\_sec : 记录了最近一次创建 RDB 文件耗费的秒数
  + rdb\_current\_bgsave\_time\_sec : 如果服务器正在创建 RDB 文件，那么这个域记录的就是当前的创建操作已经耗费的秒数。
  + aof\_enabled : 一个标志值，记录了 AOF 是否处于打开状态。
  + aof\_rewrite\_in\_progress : 一个标志值，记录了服务器是否正在创建 AOF 文件。
  + aof\_rewrite\_scheduled : 一个标志值，记录了在 RDB 文件创建完毕之后，是否需要执行预约的 AOF 重写操作。
  + aof\_last\_rewrite\_time\_sec : 最近一次创建 AOF 文件耗费的时长
  + aof\_current\_rewrite\_time\_sec : 如果服务器正在创建 AOF 文件，那么这个域记录的就是当前的创建操作已经耗费的秒数。
  + aof\_last\_bgrewrite\_status : 一个标志值，记录了最近一次创建 AOF 文件的结果是成功还是失败。

如果 AOF 持久化功能处于开启状态，那么这个部分还会加上以下域：

* + aof\_current\_size : AOF 文件目前的大小
  + aof\_base\_size : 服务器启动时或者 AOF 重写最近一次执行之后，AOF 文件的大小。
  + aof\_pending\_rewrite : 一个标志值，记录了是否有 AOF 重写操作在等待 RDB 文件创建完毕之后执行。
  + aof\_buffer\_length : AOF 缓冲区的大小
  + aof\_rewrite\_buffer\_length : AOF 重写缓冲区的大小
  + aof\_pending\_bio\_fsync : 后台 I/O 队列里面，等待执行的 fsync 调用数量。
  + aof\_delayed\_fsync : 被延迟的 fsync 调用数量
* **stats** 部分记录了**一般统计信息**，它包含以下域：
  + total\_**connections**\_received : 服务器**已接受的连接**请求数量 (可判断出Client是否使用"长链接"技术)
  + ~~total\_commands\_processed : 服务器已执行的命令数量~~
  + instantaneous\_**ops**\_per\_sec : 服务器每秒钟执行的命令数量 (查看服务器的压力)
  + ~~total\_net\_input\_bytes:32765375~~
  + ~~total\_net\_output\_bytes:497891739~~
  + instantaneous\_**input**\_kbps:0.00
  + instantaneous\_**output**\_kbps:0.00
  + **rejected\_connections** : 因为**最大客户端数量限制**而**被拒绝**的**连接请求**数量 (需要**报警**，是否需要调大'ulimit -n'值)
  + sync\_full:0
  + sync\_partial\_ok:0
  + sync\_partial\_err:0
  + **expired\_keys** : 因为**过期**而被自动删除的**键**数量
  + **evicted\_keys** : 因为**最大内存容量限制**而**被驱逐**（evict）的**键**数量
  + keyspace\_hits : 查找数据库键**成功**的次数
  + keyspace\_misses : 查找数据库键**失败**的次数 (同时，展示键的命中率)
  + **keyspace\_hits/(keyspace\_hits+keyspace\_misses)**:**键的命中率（**XX%**）**
  + pubsub\_channels : 目前被订阅的频道数量
  + pubsub\_patterns : 目前被订阅的模式数量
  + **latest\_fork\_usec** : 最近一次**fork**()操作耗费的**毫秒数**（最近一次fork操作导致了**多少时间的卡顿**）
* **replication** : **主/从复制**信息
  + **role** : 如果**当前服务器**没有在复制任何其他服务器，那么这个域的值就是 master ；否则的话，这个域的值就是 slave 。注意，在创建复制链的时候，一个从服务器也可能是另一个服务器的主服务器。

如果当前服务器是一个**从服务器**的话，那么这个部分还会加上以下域：

* + master\_host : 主服务器的 IP 地址
  + master\_port : 主服务器的 TCP 监听端口号
  + **master\_link\_status** : 复制**连接当前的状态**，**up** 表示连接**正常**，down 表示连接**断开**。
  + master\_last\_io\_seconds\_ago : 距离最近一次与主服务器进行通信已经过去了多少**秒钟**
  + master\_sync\_in\_progress : 一个标志值，记录了主服务器是否正在与这个从服务器进行同步。

如果同步操作正在进行，那么这个部分还会加上以下域：

* + master\_sync\_left\_bytes : 距离同步完成还缺少多少字节数据。
  + master\_sync\_last\_io\_seconds\_ago : 距离最近一次因为 SYNC 操作而进行 I/O 已经过去了多少秒。

如果主从服务器之间的连接处于断线状态，那么这个部分还会加上以下域：

* + master\_link\_down\_since\_seconds : 主从服务器连接断开了多少秒

以下是一些总会出现的域：

* + **connected\_slaves** : 已连接的**从服务器**数量
  + master\_repl\_offset:0
  + **repl\_backlog\_active**:0 (请求是否有堆积)
  + repl\_backlog\_size:1048576
  + repl\_backlog\_first\_byte\_offset:0
  + repl\_backlog\_histlen:0

对于每个**从服务器**，都会添加以下一行信息：

* + slaveXXX : ID、IP 地址、端口号、连接状态
* **cpu** 部分记录了 **CPU 的计算量统计**信息，它包含以下域：
  + ~~used\_cpu\_sys : Redis 服务器耗费的系统 CPU~~ (累加值，无多少参考价值)
  + ~~used\_cpu\_user : Redis 服务器耗费的用户 CPU~~
  + ~~used\_cpu\_sys\_children : 后台进程耗费的系统 CPU~~
  + ~~used\_cpu\_user\_children : 后台进程耗费的用户 CPU~~
* **commandstats** 部分记录了**各种不同类型的命令的执行统计信息**，比如命令执行的次数、命令耗费的 CPU 时间、执行每条命令耗费的平均 CPU 时间等等。对于每种类型的命令，这个部分都会添加一行以下格式的信息：
  + **cmdstat**\_XXX:calls=XXX,usec=XXX, usec\_per\_call=XXX
* **cluster** 部分记录了**和集群有关的信息**，它包含以下域：
  + **cluster\_enabled** : 一个标志值，记录**集群功能**是否已经开启
* **keyspace** 部分记录了**数据库相关的统计信息**，比如数据库的键数量、数据库已经被删除的过期键数量等。对于每个数据库，这个部分都会添加一行以下格式的信息：
  + **db**XXX:keys=XXX,expires=XXX ,avg\_ttl=XXX

除上面给出的这些值以外， section 参数的值还可以是下面这两个：

* **all** : 返回所有信息
* default : 返回默认选择的信息

当不带参数直接调用 [INFO](http://redisdoc.com/server/info.html#info) 命令时，使用 default 作为默认参数。

不同版本的 Redis 可能对返回的一些域进行了增加或删减。

因此，一个健壮的客户端程序在对 [*INFO*](http://redisdoc.com/server/info.html#info) 命令的输出进行分析时，应该能够跳过不认识的域，并且妥善地处理丢失不见的域。

**可用版本：**

>= 1.0.0

**时间复杂度：**

O(1)

**返回值：**

具体请参见下面的测试代码。

参考资料

1. [INFO - 返回关于 Redis 服务器的各种信息和统计数值](http://redisdoc.com/server/info.html)
2. [INFO - Get information and statistics about the server](http://redis.io/commands/info)