拓展 2：无所不知 —— Info 指令

在使用 Redis 时，时常会遇到很多问题需要诊断，在诊断之前需要了解 Redis 的运行状态，通过强大的 Info 指令，你可以清晰地知道 Redis 内部一系列运行参数。

Info 指令显示的信息非常繁多，分为 8 大块，每个块都有非常多的参数，这 8 个块分别是:

1. Server 服务器运行的环境参数
2. Clients 客户端相关信息
3. Memory 服务器运行内存统计数据
4. Persistence 持久化信息
5. Stats 通用统计数据
6. Replication 主从复制相关信息
7. CPU CPU 使用情况
8. Cluster 集群信息
9. KeySpace 键值对统计数量信息

Info 可以一次性获取所有的信息，也可以按块取信息。

```
# 获取所有信息
> info
# 获取内存相关信息
> info memory
# 获取复制相关信息
> info replication
```

考虑到参数非常繁多，——说明工作量巨大，下面我只挑一些关键性的、非常实用和最常用的参数进行详细讲解。如果读者想要了解所有的参数细节，请参考阅读 Redis 官网文档。

Redis 每秒执行多少次指令?

这个信息在 Stats 块里，可以通过 `info stats` 看到。

```
# ops_per_sec: operations per second，也就是每秒操作数
> redis-cli info stats |grep ops
instantaneous_ops_per_sec:789
```

以上，表示 ops 是 789，也就是所有客户端每秒会发送 789 条指令到服务器执行。极限情况下，Redis 可以每秒执行 10w 次指令，CPU 几乎完全榨干。如果 qps 过高，可以考虑通过 `monitor` 指令快速观察一下究竟是哪些 key 访问比较频繁，从而在相应的业务上进行优化，以减少 IO 次数。 `monitor` 指令会瞬间吐出来巨量的指令文本，所以一般在执行 `monitor` 后立即 `ctrl+c` 中断输出。

```
> redis-cli monitor
```

Redis 连接了多少客户端?

这个信息在 Clients 块里，可以通过 `info clients` 看到。

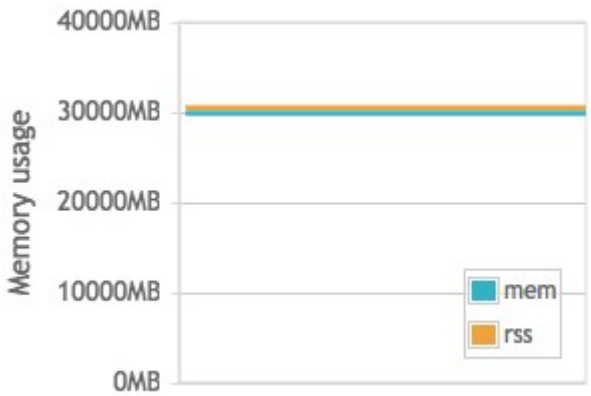
```
> redis-cli info clients
# Clients
connected_clients:124  # 这个就是正在连接的客户端数量
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0
```

这个信息也是比较有用的，通过观察这个数量可以确定是否存在意料之外的连接。如果发现这个数量不对劲，接着就可以使用 `client list` 指令列出所有的客户端链接地址来确定源头。

关于客户端的数量还有个重要的参数需要观察，那就是 `rejected_connections`，它表示因为超出最大连接数限制而被拒绝的客户端连接次数，如果这个数字很大，意味着服务器的最大连接数设置的过低需要调整 `maxclients` 参数。

```
> redis-cli info stats |grep reject
rejected_connections:0
```

Redis 内存占用多大？



这个信息在 Memory 块里，可以通过 `info memory` 看到。

```
> redis-cli info memory | grep used | grep human
used_memory_human:827.46K # 内存分配器 (jemalloc) 从操作系统分配的内存总量
used_memory_rss_human:3.61M  # 操作系统看到的内存占用 ,top 命令看到的内存
used_memory_peak_human:829.41K  # Redis 内存消耗的峰值
used_memory_lua_human:37.00K # lua 脚本引擎占用的内存大小
```

如果单个 Redis 内存占用过大，并且在业务上没有太多压缩的空间的话，可以考虑集群化了。

复制积压缓冲区多大?

这个信息在 Replication 块里，可以通过 `info replication` 看到。

```
> redis-cli info replication |grep backlog
repl_backlog_active:0
repl_backlog_size:1048576  # 这个就是积压缓冲区大小
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
```

复制积压缓冲区大小非常重要，它严重影响到主从复制的效率。当从库因为网络原因临时断开了主库的复制，然后网络恢复了，又重新连上的时候，这段断开的时间内发生在 master 上的修改操作指令都会放在积压缓冲区中，这样从库可以通过积压缓冲区恢复中断的主从同步过程。

积压缓冲区是环形的，后来的指令会覆盖掉前面的内容。如果从库断开的时间过长，或者缓冲区的大小设置的太小，都会导致从库无法快速恢复中断的主从同步过程，因为中间的修改指令被覆盖掉了。这时候从库就会进行全量同步模式，非常耗费 CPU 和网络资源。

如果有多个从库复制，积压缓冲区是共享的，它不会因为从库过多而线性增长。如果实例的修改指令请求很频繁，那就把积压缓冲区调大一些，几十个 M 大小差不多了，如果很闲，那就设置为几个 M。

```
> redis-cli info stats | grep sync
sync_full:0
sync_partial_ok:0
sync_partial_err:0  # 半同步失败次数
```

通过查看 `sync_partial_err` 变量的次数来决定是否需要扩大积压缓冲区，它表示主从半同步复制失败的次数。

思考

平时你们在使用 Redis 时还需要查看哪些重要的信息，能不能直接在 Info 信息里获取?