原理 2：交头接耳 —— 通信协议

Redis 的作者认为数据库系统的瓶颈一般不在于网络流量，而是数据库自身内部逻辑处理上。所以即使 Redis 使用了浪费流量的文本协议，依然可以取得极高的访问性能。Redis 将所有数据都放在内存，用一个单线程对外提供服务，单个节点在跑满一个 CPU 核心的情况下可以达到了 10w/s 的超高 QPS。

RESP(Redis Serialization Protocol)

RESP 是 Redis 序列化协议的简写。它是一种直观的文本协议，优势在于实现异常简单，解析性能极好。

Redis 协议将传输的结构数据分为 5 种最小单元类型，单元结束时统一加上回车换行符号 `\r\n` 。

1. 单行字符串 以 `+` 符号开头。
2. 多行字符串 以 `$` 符号开头，后跟字符串长度。
3. 整数值 以 `:` 符号开头，后跟整数的字符串形式。
4. 错误消息 以 `-` 符号开头。
5. 数组 以 `*` 号开头，后跟数组的长度。

单行字符串 hello world

```
+hello world\r\n
```

多行字符串 hello world

```
$11\r\nhello world\r\n
```

多行字符串当然也可以表示单行字符串。

整数 1024

```
:1024\r\n
```

错误 参数类型错误

```
-WRONGTYPE Operation against a key holding the wrong kind of value
```

数组 [1,2,3]

```
*3\r\n:1\r\n:2\r\n:3\r\n
```

NULL 用多行字符串表示，不过长度要写成-1。

```
$-1\r\n
```

空串 用多行字符串表示，长度填 0。

```
$0\r\n\r\n
```

注意这里有两个 `\r\n` 。为什么是两个?因为两个 `\r\n` 之间,隔的是空串。

客户端 -> 服务器

客户端向服务器发送的指令只有一种格式，多行字符串数组。比如一个简单的 set 指令 `set author codehole` 会被序列化成下面的字符串。

```
*3\r\n$3\r\nset\r\n$6\r\nauthor\r\n$8\r\ncodehole\r\n
```

在控制台输出这个字符串如下，可以看出这是很好阅读的一种格式。

```
*3
$3
set
$6
author
$8
codehole
```

服务器 -> 客户端

服务器向客户端回复的响应要支持多种数据结构，所以消息响应在结构上要复杂不少。不过再复杂的响应消息也是以上 5 中基本类型的组合。

单行字符串响应

```
127.0.0.1:6379> set author codehole
OK
```

这里的 OK 就是单行响应，没有使用引号括起来。

```
+OK
```

错误响应

```
127.0.0.1:6379> incr author
(error) ERR value is not an integer or out of range
```

试图对一个字符串进行自增，服务器抛出一个通用的错误。

```
-ERR value is not an integer or out of range
```

整数响应

```
127.0.0.1:6379> incr books
(integer) 1
```

这里的 1 就是整数响应

```
:1
```

多行字符串响应

```
127.0.0.1:6379> get author
"codehole"
```

这里使用双引号括起来的字符串就是多行字符串响应

```
$8
codehole
```

数组响应

```
127.0.0.1:6379> hset info name laoqian
(integer) 1
127.0.0.1:6379> hset info age 30
(integer) 1
127.0.0.1:6379> hset info sex male
(integer) 1
127.0.0.1:6379> hgetall info
1) "name"
2) "laoqian"
3) "age"
4) "30"
5) "sex"
6) "male"
```

这里的 hgetall 命令返回的就是一个数值，第 0|2|4 位置的字符串是 hash 表的 key，第 1|3|5 位置的字符串是 value，客户端负责将数组组装成字典再返回。

```
*6
$4
name
$6
laoqian
$3
age
$2
30
$3
sex
$4
male
```

嵌套

```
127.0.0.1:6379> scan 0
1) "0"
2) 1) "info"
   2) "books"
   3) "author"
```

scan 命令用来扫描服务器包含的所有 key 列表，它是以游标的形式获取，一次只获取一部分。

scan 命令返回的是一个嵌套数组。数组的第一个值表示游标的值，如果这个值为零，说明已经遍历完毕。如果不为零，使用这个值作为 scan 命令的参数进行下一次遍历。数组的第二个值又是一个数组，这个数组就是 key 列表。

```
*2
$1
0
*3
$4
info
$5
books
$6
author
```

小结

Redis 协议里有大量冗余的回车换行符，但是这不影响它成为互联网技术领域非常受欢迎的一个文本协议。有很多开源项目使用 RESP 作为它的通讯协议。在技术领域性能并不总是一切，还有简单性、易理解性和易实现性，这些都需要进行适当权衡。

扩展阅读

如果你想自己实现一套Redis协议的解码器，请阅读老钱的另一篇文章《基于Netty实现Redis协议的编码解码器》