Redis 协议解析

Created by 陈宝仪 (@leonchen83)

目录

1. Redis 简介

1.1. 简介

- NoSQL 数据库
- 丰富的数据结构(string, list, set, hash, sortedset, module, stream)
- 丰富的命令 (GEO, Hyperloglog, BitField, Eval script)
- 性能高效 (非pipeline 10k/s, pipeline 400k/s)
- 持久化

1.2. 应用场景

- 缓存
- 分布式session
- 计数器 全局id
- 排行榜 Top N

2. 传输协议 RESP

2.1. RESP

```
./redis/src/redis-cli

127.0.0.1:6379>set key value
OK
```

`*3\r\n$3\r\nset\r\n$3\r\nkey\r\n$5\r\nvalue\r\n`

命令的长度 `*3\r\n`
命令的内容 `$3\r\nset\r\n`, `$3\r\nkey\r\n`, `$5\r\nvalue\r\n` (bulk string)
bulk string表示null: `$-1\r\n`. bulk string表示空串: `$0\r\n\r\n`

RESP array表示null: `*-1\r\n`. RESP array表示空数组: `*0\r\n`

RESP Integers: `:1000\r\n`

RESP Simple Strings: `+OK\r\n`

RESP Errors: `-Error message\r\n`

一个复杂的例子: `*3\r\n$3\r\nset\r\n$3\r\nkey\r\n*2\r\n$-1\r\n:1024\r\n`
[set, key, [nil, 1024]]

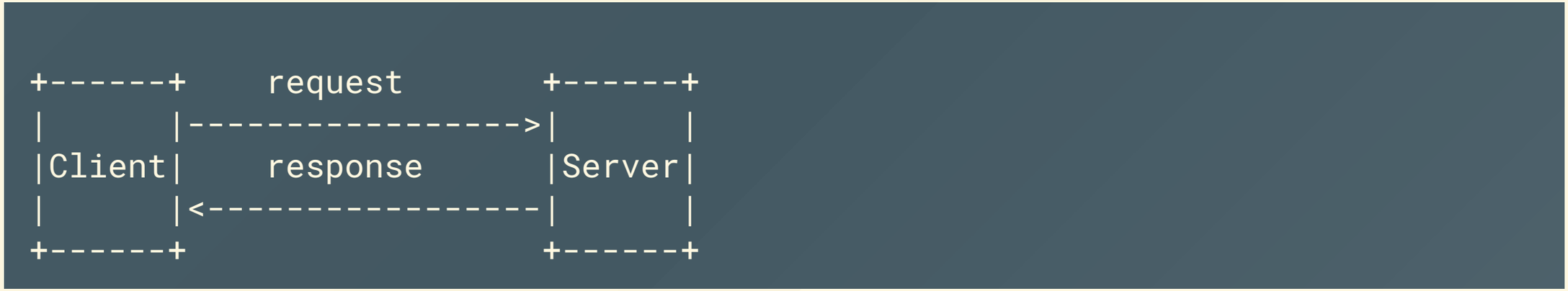
2.2. escape

- `set "a b" "c"` 转义成`set "a\tb" "c"`
- unprintable character (b<=32 || b>=127) 转义成HEX 用`\xFF`表示

```
127.0.0.1:6379> set abc abc
127.0.0.1:6379> dump abc
"\x00\x03abc\b\x00`\x81\xdc\x82\xe8k(\xda"
```

```
[0, 3, 97, 98, 99, 8, 0, 96, -127, -36, -126, -24, 107, 40, -38]
```

2.3. pipeline



send `+PING\r\n+PING\r\n+PING\r\n+PING\r\n+PING\r\n+PING\r\n`

fetch `+PONG\r\n+PONG\r\n+PONG\r\n+PONG\r\n+PONG\r\n+PONG\r\n`

一次发送多条命令以提高redis的吞吐量

2.4. Jedis 与 Redis 连接暴涨

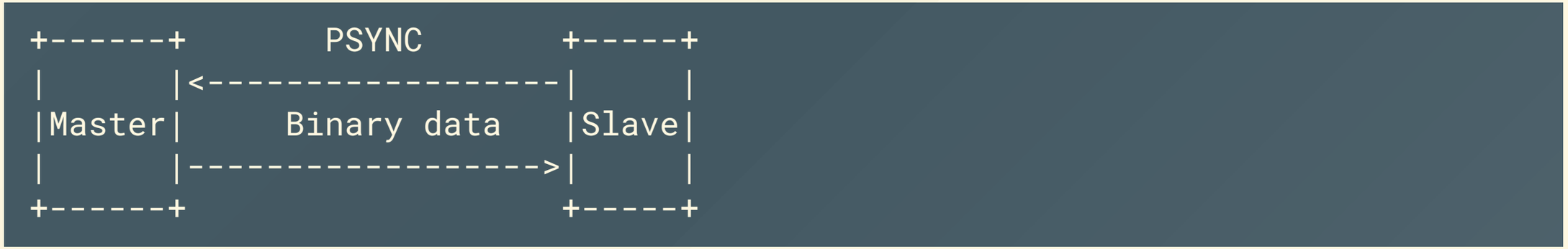
```
JedisPool pool = new JedisPool("127.0.0.1", 6379);
Jedis redis = null;
try {
    redis = pool.getResource();
    String value = redis.get("key");
    System.out.println(value);
} catch (JedisConnectionException e) {
    if (redis != null) {
        pool.returnBrokenResource(redis);
        redis = null;
    }
    throw e;
} finally {
    if (redis != null) {
        pool.returnResource(redis);
    }
}
```

2.5. 正确的做法

```
JedisPool pool = new JedisPool("127.0.0.1", 6379);
Jedis redis = null;
try {
    redis = pool.getResource();
    String value = redis.get("key");
    System.out.println(value);
} finally {
    if (redis != null) {
        redis.close();
    }
}
```

3. 同步协议 (Replication Protocol)

3.1. 同步过程



3.2. 与 Master 交互

- 发送 AUTH password
- 发送 REPLCONF listening-port port
- 发送 PSYNC repl_id repl_offset
- 发送 REPLCONF ip-address address
- 发送 REPLCONF capa eof (diskless-replication)
- 发送 REPLCONF capa psync2
- 接收 Binary data

3.3. Binary data 的格式

- 服务器首先返回 `+FULLRESYNC repl_id repl_offset\r\n` (`+CONTINUE\r\n`)
- 然后返回 `$payload\r\nRDB`
- 再然后返回积压在backlog里的命令如 `*2\r\n$6\r\nSELECT\r\n$1\r\n0\r\n` (select 0)
- 同步完RDB之后Slave定期向Master发送`REPLCONF ACK offset`报告自己同步的位置
- 同步完RDB没有其他命令需要同步时， master会定期给slave发送PING命令以维持长连接

3.4. RDB 格式

```
RDB         =     'REDIS', $version,
                  {DBSELECT, {RECORD}}, '0xFF', [$checksum];
RECORD      =     KEY, VALUE;
DBSELECT    =     '0xFE', $length;
KEY         =     $string;
VALUE       =     $value-type, ( $string
                               | $list
                               | $set
                               | $zset
                               | ...
```

```
|Magic|Version|DB 0    | Key1| Value1| Key2..| Value2..|DB 1   |..|EOF |CRC    |
|REDIS|0006   |0xFE 0 | abc | cde    | ...    | ...      |0xFE 1|..|0xFF|8bytes|
```

[RDB dump data format]

$length

```
|11xxxxxx| 余下的6bit表示长度，并表示这是个encoded类型
|00xxxxxx| 余下的6bit表示长度 2^6
|01xxxxxx|xxxxxxxx| 14bit表示长度 2^14
|10000000|xxxxxxxx|xxxxxxxx|xxxxxxxx|xxxxxxxx| 4字节表示长度 2^32
|10000001|xxxxxxxx|xxxxxxxx|xxxxxxxx|xxxxxxxx|....| 8字节表示长度 2^64
```

$value-type

```
0 = $string, 1 = $list, 2 = $set, 3 = $zset, 4 = $hash, 5 = $zset2
6 = $module, 7 = $module2, 9 = $hashzipmap, 10 = $listziplist
11 = $setintset, 12 = $zsetziplist, 13 = $hashziplist, 14 = $listquicklist
15 = $streamlistpacks
```

$string

```
$length content
计算content
根据$length读取相应的字节返回字节数组
当$length为
|11xxxxxx| 余下的6bit表示长度，并表示这是个encoded类型
0, 读取1个字节
1，读取2个字节
2，读取4个字节
3，表明这个string是lzf压缩格式，  需要解压缩
```

$checksum

```
8 个字节的CRC64验证
```

3.5. 同步协议的其他实现细节

- diskless-replication

  ```
  $EOF:<40 bytes delimiter>\r\nRDB<40 bytes delimiter>
  ```

- 传输巨大RDB的细节

  ```
  +FULLRESYNC repl_id repl_offset\r\n\n\n\n\n\n\n\n
  ```

- PSYNC2: 将repl_id, repl_offset保存在rdb中，恢复服务的时候不用全量同步
- 最终一致性
- 模拟Slave协议的一些工具

3.6. 同步需要注意的问题

- master请求过于频繁，迅速填满backlog，导致slave无限同步
- 由最终一致性导致潜在数据不一致
- 大实例与同步大key导致的问题

4. 未来 Redis 的传输协议RESP3

4.1. RESP2的缺点

缺少浮点数， boolean类型， 暂时全由string表示， null值多重表示， 内部滥用RESP2.

4.2. RESP3

- Blob string: `$11<LF>helloworld<LF>` , or escape blob string: `"$11\nhelloworld\n"`
- Simple string: `+hello world<LF>` , or escape simple string: `"+hello world\n"`
- Simple error: `-ERR this is the error description<LF>` , or escape simple error: `"-ERR this is the error description\n"`
- Number: `:1234<LF>`, or escape number: `":1234\n"`
- Null: `_<LF>` or escape null: `"_\n"`

- Double: `,1.23<LF>` or escape doulbe: `",1.23\n"`
- `:10<LF>`与`,10<LF>`的区别
- Boolean: `#t<LF>` or `#f<LF>`
- Blob error: `!21<LF>SYNTAX invalid syntax<LF>`
- Verbatim string: `=15<LF>txt<LF>Some string<LF>`
- Big number: `(3492890328409238509324850943850943825024385<LF>`

- Array type:

```
*3<LF>
:1<LF>
:2<LF>
:3<LF>
// Array[1, 2, 3]
*2<LF>
    *3<LF>
        :1<LF>
        $5<LF>
        hello<LF>
        :2<LF>
    #f<LF>
// Array[[1, "hello", 2], false]
```

- Map type:

```
{
    "first":1,
    "second":2
}

%2<LF>
    +first<LF>:1<LF>
    +second<LF>:2<LF>
```

- Set type:

```
~5<LF>
    +orange<LF>
    +apple<LF>
    #t<LF>
    :100<LF>
    :999<LF>
// Set["apple", 100, 999, true, "orange"]
```

- Attribute type

```
|1<LF>
    +key-popularity<LF>
    %2<LF>
        +a<LF>,0.1923<LF>
        +b<LF>,0.0012<LF>
*2<LF>
    :2039123<LF>:9543892<LF>
// Array[2039123, 9543892]
// return object with attribute
// key-popularity = {"a" : 0.1923, "b" : 0.0012}
*3<LF>
    :1<LF>:2<LF>
    |1<LF>
        +ttl<LF>:3600<LF>
    :3<LF>
// Array[1, 2, 3]
// attribute: ttl=3600
```

5. Redis 新特性简介

- Redis-4.0.x module. redis-modules-hub
- Redis-4.0.x async delete `FLUSHALL ASYNC`, `UNLINK key`, `FLUSHDB ASYNC`
- Redis-4.0.x Cluster RCmb v1
- Redis-4.0.x aof-use-rdb-preamble
- Redis-5.0.x Stream