Lesson2 Redis持久化

AOF & RDB

Bo.Xiao

Agenda

- 通信协议
- 持久化简介
- AOF详解
- RDB详解

Redis通信协议

Redis通信协议

- 易于实现
- 可以高效地被计算机分析
- 可以很容易地被人类读懂

通讯协议

*<参数数量> CR LF

$<参数 1 的字节数量> CR LF

<参数 1 的数据> CR LF

...

$<参数 N 的字节数量> CR

*3

$3

SET

$5

mykey

$7

myvalue

通讯协议

状态回复（status reply）的第一个字节是 "+"

错误回复（error reply）的第一个字节是 "-"

整数回复（integer reply）的第一个字节是 ":"

批量回复（bulk reply）的第一个字节是 "$"

多条批量回复（multi bulk reply）的第一个字节是 "*"

客户端：GET mykey

服务器：foobar

"$6\r\nfoobar\r\n"

持久化简介

持久化简介

- 数据持久化就是将数据存储到断电后数据不丢失的设备中
- Redis的持久化包括RDB和AOF两种方式

- 客户端向服务发送写请求
- 服务端接受写请求
- 服务端调用 Write 系统调用，将数据写往磁盘
- 操作系统将缓冲区中的数据转移到磁盘
- 磁盘控制器将数据写到磁盘的物理介质中

AOF详解

AOF是什么

AOF（Append Only File ）是基于变更日志的持久化

```
[root@LOR-Platform-Fraudstore-18 data]# tail -n 20 appendonly.aof
$4
ZADD
$42
orderId//234626581//compare 1-1100-me-cNum
$14
1.499425695E12
$30
{"data0":"0","id":"532154787"}
*3
$9
PEXPIREAT
$42
orderId//234626581//compare 1-1100-me-cNum
$13
1500030510401
*2
$3
DEL
$42
member//190000000045796995//compare member
[root@LOR-Platform-Fraudstore-18 data]#
```

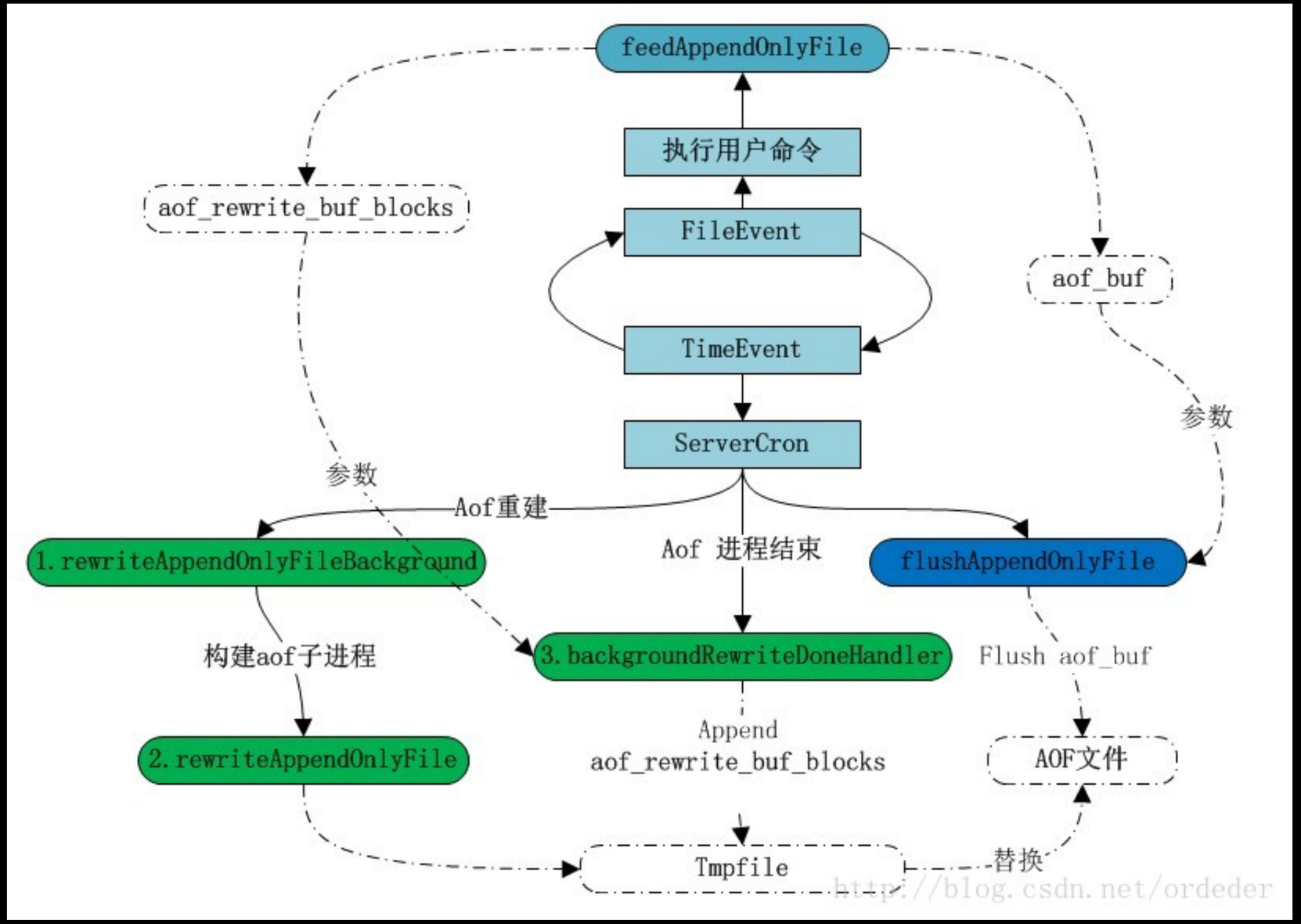
AOF持久化策略

- AOF_FSYNC_NO:不保存
- AOF_FSYNC_EVERYSEC:每秒钟保存一次
- AOF_FSYNC_ALWAYS:每次执行数据变更命令

AOF Rewrite

```
struct redisServer {
        ......
    int aof_state;                  /* AOF状态
REDIS_AOF_(ON|OFF|WAIT_REWRITE) */
    int aof_fsync;                  /* fsync()策略 */
    char *aof_filename;             /* AOF文件名 */
    int aof_no_fsync_on_rewrite;    /* 在rewrite期间是否fsync */
    int aof_rewrite_perc;           /* M 当AOF文件达到上次rewrite后文件大小的
M倍后触发rewrite */
    off_t aof_rewrite_min_size;     /* AOF文件rewrite最小大小 */
    off_t aof_rewrite_base_size;    /* 上一次rewrite后的AOF文件大小 */
    off_t aof_current_size;         /* 现在AOF文件大小 */
    int aof_rewrite_scheduled;      /* 当bgsave结束后开始rewrite */
    pid_t aof_child_pid;            /* rewrite进程的pid */
    list *aof_rewrite_buf_blocks;   /* rewrite期间的AOF缓冲 */
    sds aof_buf;      /* AOF缓冲，需要在事件循环中被fsync同步到硬盘上 */
    int aof_fd;       /* 现在AOF文件的fd */
    int aof_selected_db; /* AOF文件现在指定的DB编号 */
    time_t aof_flush_postponed_start; /* 上一次推迟fsync的时间 */
    time_t aof_last_fsync;            /* 上一次fsync的时间 */
    time_t aof_rewrite_time_last;   /* 上一次rewrite时间 */
    time_t aof_rewrite_time_start;  /* 这次rewrite的开始时间 */
    int aof_lastbgrewrite_status;   /* REDIS_OK or REDIS_ERR */
    unsigned long aof_delayed_fsync;  /* fsync拖延次数 */
```

AOF如何发生的

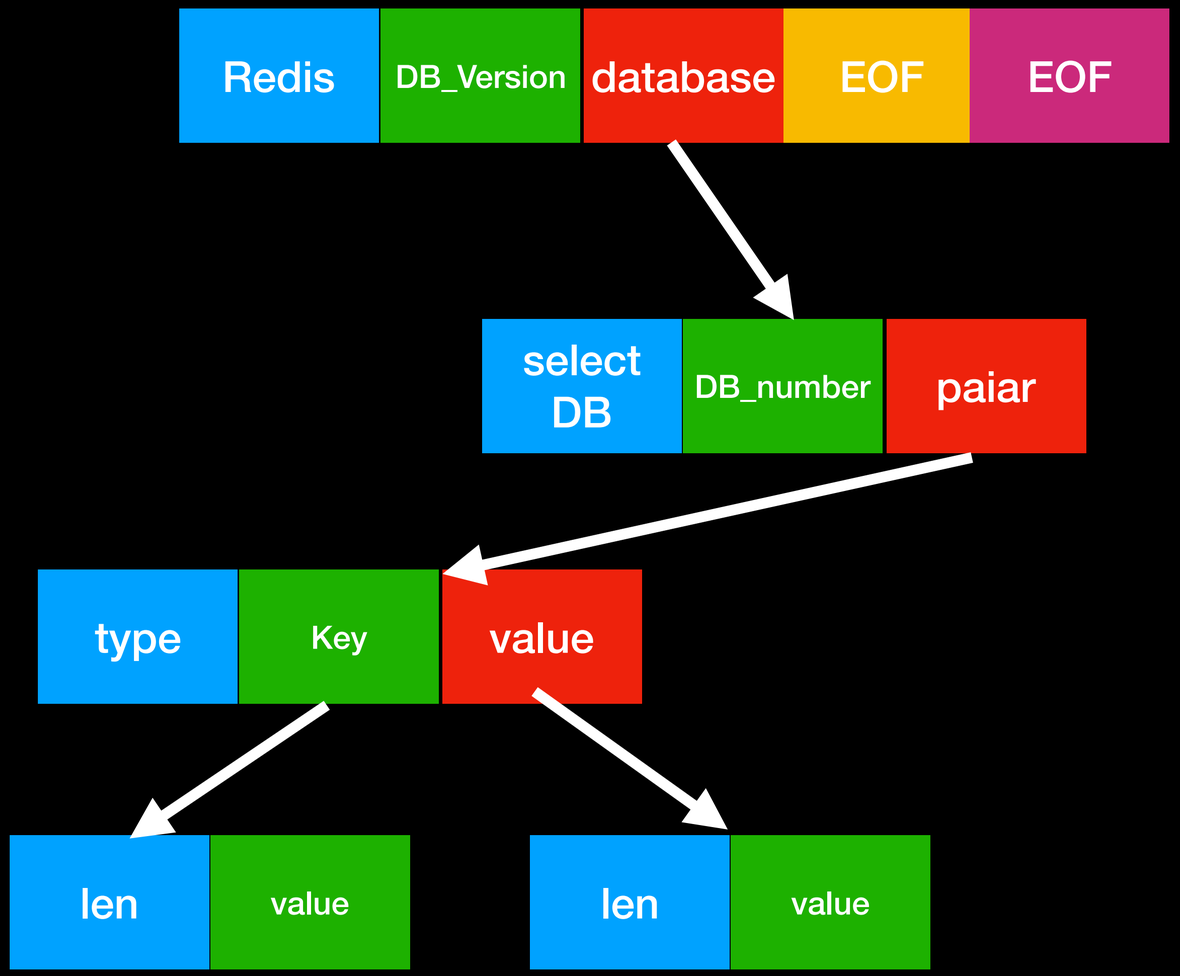

feedAppendOnlyFile
执行用户命令
aof_rewrite_buf_blocks
FileEvent
aof_buf
TimeEvent
参数
ServerCron
参数
Aof重建
1.rewriteAppendOnlyFileBackground
Aof 进程结束
flushAppendOnlyFile
构建aof子进程
3.backgroundRewriteDoneHandler
Flush aof_buf
Append
aof_rewrite_buf_blocks
2.rewriteAppendOnlyFile
AOF文件
Tmpfile
替换
http://blog.csdn.net/ordeder

RDB详解

RDB是什么

RDB是基于数据快照的持久化

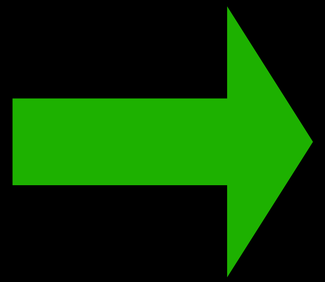
RDB是什么样的




```
0000000 4552 4944 3053 3030 fa37 7209 6465 7369
0000010 762d 7265 3305 322e 372e 0afa 6572 6964
0000020 2d73 6962 7374 40c0 05fa 7463 6d69 c265
0000030 d64a 5961 08fa 7375 6465 6d2d 6d65 c8c2
0000040 11d9 fe00 fb00 7c4d 0b00 8dc2 0cb9 1000
0000050 0008 0000 0001 0000 54e0 5358 6d67 1bc1
0000060 c20b 3fa5 0003 0810 0000 0100 0000 5e00
0000070 1c44 fe4d a303 0b02 6ec2 1052 1000 0008
0000080 0000 0001 0000 89e9 4da2 03fe 02a3 c20b
0000090 a497 0010 040c 0000 0100 0000 1c00 9707
00000a0 0b24 53c2 03f3 1000 0008 0000 0001 0000
00000b0 10b6 4bcc 03fe 02a3 c20b 04bd 000d 0810
00000c0 0000 0100 0000 f500 3839 ba82 54a6 0b03
00000d0 71c2 0fb2 1000 0008 0000 0001 0000 090d
00000e0 86ba a6ba 0354 c10b 34c4 0810 0000 0100
00000f0 0000 4200 664f ba82 54a6 0b03 47c2 09ae
0000100 1000 0008 0000 0001 0000 8416 861e a6ba
0000110 0354 c20b b83c 000a 0810 0000 0100 0000
0000120 ae00 4e2e 0211 0000 0b00 bbc2 11fc 1000
0000130 0008 0000 0001 0000 3dc6 55e2 0577 0000
0000140 c20b 7238 0010 0810 0000 0100 0000 5300
0000150 290b fe4d a303 0b02 d9c1 1054 0008 0000
0000160 0001 0000 cbbf 4dca 03fe 02a3 c20b 2f78
0000170 0007 0810 0000 0100 0000 2000 cd8f fe4d
0000180 a303 0b02 0ac2 09ca 1000 0008 0000 0001
0000190 0000 5233 4bdb 03fe 02a3 c20b 9b92 0001
00001a0 0810 0000 0100 0000 1400 129b ba7f 54a6
00001b0 0b03 55c2 0a93 1000 0008 0000 0001 0000
00001c0 b486 4bce 03fe 02a3 c20b 3f5e 0003 0810
00001d0 0000 0100 0000 5e00 1c44 fe4d a303 0b02
00001e0 d8c2 0b4d 1000 0008 0000 0001 0000 6d26
00001f0 4cb5 03fe 02a3 c20b af41 0001 0810 0000
0000200 0100 0000 2c00 5c53 fe4b a303 0b02 e0c2
0000210 06a3 1000 0008 0000 0001 0000 64bb 84ed
```

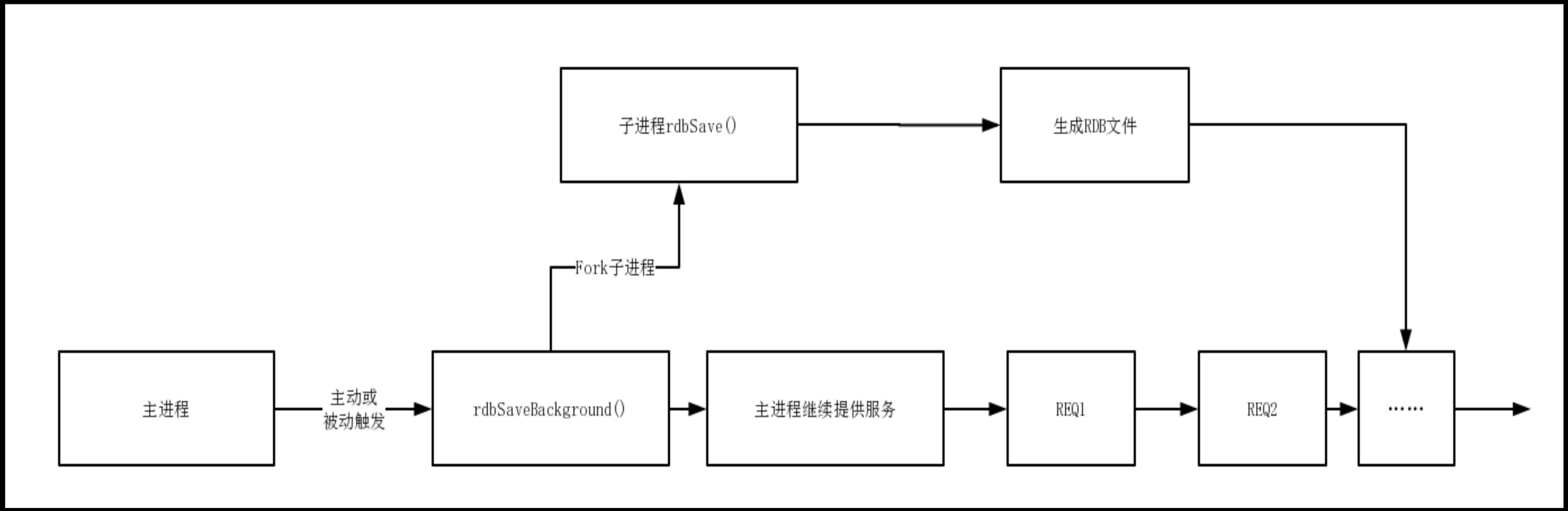
RDB持久化策略

- save 900 1
- save 300 10
- save 60 10000



- 900秒（15分钟）内至少1个key值改变（则进行数据库保存--持久化）
- 300秒（5分钟）内至少10个key值改变（则进行数据库保存--持久化）
- 60秒（1分钟）内至少10000个key值改变（则进行数据库保存--持久化）

RDB如何发生的


子进程rdbSave()
生成RDB文件
Fork子进程
主进程
主动或
被动触发
rdbSaveBackground()
主进程继续提供服务
REQ1
REQ2
……

SAVE与BGSAVE

```
def SAVE():

    rdbSave()


def BGSAVE():

    pid = fork()

    if pid == 0:

        # 子进程保存 RDB
        rdbSave()

    elif pid > 0:

        # 父进程继续处理请求，并等待子进程的完成信号
        handle_request()

    else:

        # pid == -1
        # 处理 fork 错误
        handle_fork_error()
```

AOF与RDB

- AOF保存的数据集比RDB更完整
- RDB文件比AOF文件更加紧凑
- RDB文件恢复速度优于AOF


Q/A