Redis的高级特性

李焕贞

河北师范大学软件学院

http://redisdoc.com/

高级应用

1. 安全性
2. 主从复制
3. 事务处理
4. 持久化机制
5. 发布订阅信息
6. 虚拟内存的使用

安全性

设置客户端连接后进行任何命令前需要使用的密码,

注意：重启服务器

```
#
# requirepass foobared
requirepass lhz123
# Command renaming.
#
```

```
127.0.0.1:6379> auth lhz123
```

```
[root@localhost bin]# ./redis-cli -a lhz123
```

主从复制

通过主从复制可以允许多个slave server拥有和master server相同的数据库副本。

主从复制特点：

1、master可以拥有多个slave

2、多个slave可以连接同一个master外，还可以互相连接

3、主从复制不会阻塞master，在同步数据时，master可以继续处理client请求

4、提高系统的伸缩性

https://redis.io/topics/replication

http://www.redis.cn/topics/replication.html

主从复制

Redis主从复制过程：

1、slave与master建立连接，发送sync同步命令

2、master会启动一个后台进程，将数据库快照保存到文件中，同时master主进程会开始收集新的写命令并缓存。

3、后台完成保存后，就将此文件发送给slave

4、slave将此文件保存到硬盘上

主从复制

配置主从服务器

在salve配置文件中加入：

replicaof 192.168.154.122 6379

#指定master的ip和端口

masterauth lhz123

#主机密码

主服务器修改：bind 127.0.0.1 改为 bind 0.0.0.0

Redis事务控制

MULTI 、 EXEC 、 DISCARD 和 WATCH 是 Redis 事务相关的命令。事务可以一次执行多个命令， 并且带有以下两个重要的保证：

- 事务是一个单独的隔离操作：事务中的所有命令都会序列化、按顺序地执行。事务在执行的过程中，不会被其他客户端发送来的命令请求所打断。
- 事务是一个原子操作：事务中的命令要么全部被执行，要么全部都不执行。

EXEC 命令负责触发并执行事务中的所有命令。

https://redis.io/topics/transactions

http://www.redis.cn/topics/transactions.html

Redis事务控制-执行事务

```
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379> set a1 12
QUEUED
127.0.0.1:6379> set a2 13
QUEUED
127.0.0.1:6379> set a3 14
QUEUED
127.0.0.1:6379> exec
1) OK
2) OK
3) OK
127.0.0.1:6379> get a1
"12"
```

Redis事务控制-放弃事务discard

当执行 DISCARD 命令时， 事务会被放弃， 事务队列会被清空， 并且客户端会从事务状态中退出。

```
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379> set name tom
QUEUED
127.0.0.1:6379> set age 20
QUEUED
127.0.0.1:6379> DISCARD
OK
```

Redis事务控制-不支持回滚

```
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379> INCR age
QUEUED
127.0.0.1:6379> INCR name
QUEUED
127.0.0.1:6379> EXEC
1) (integer) 22
2) (error) ERR value is not an integer o
r out of range
```

Redis事务控制

乐观锁：每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在更新的时候会判断在此期间别人有没有去更新这个数据，可以使用版本号等机制。乐观锁适用于多读的应用类型，这样可以提高吞吐量。

乐观锁策略：提交版本必须大于当前版本才能执行更新

Redis事务控制

悲观锁：每次去拿数据的时候都认为别人会修改，所以每次拿数据会上锁，这样别人想拿这个数据就会阻塞，直到它拿到锁。传统的关系型数据库就用到了这种锁机制，比如：行锁、表锁，读锁，写锁等，都是在操作之前先上锁。

Redis事务控制-WATCH

WATCH 命令可以为 Redis 事务提供 check-and-set（CAS）行为。

被 WATCH 的键会被监视，并会发觉这些键是否被改动过了。如果有至少一个被监视的键在 EXEC 执行之前被修改了，那么整个事务都会被取消，EXEC 返回nil来表示事务已经失败。

Redis事务控制-乐观锁实例

实例1：

```
127.0.0.1:6379> set age 20
OK
127.0.0.1:6379> WATCH age
OK
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379> set age 40
QUEUED
127.0.0.1:6379> EXEC
(nil)
```

实例2：

```
127.0.0.1:6379> get age
"20"
127.0.0.1:6379> set age 30
OK
```

高级应用

1. 安全性
2. 主从复制
3. 事务处理
4. 持久化机制
5. 发布订阅信息
6. 虚拟内存的使用

持久化

Redis 是一个持久化的内存数据库

Redis 提供了不同级别的持久化方式（内存的数据保存到磁盘上）：

RDB（Redis DataBase）持久化方式能够在指定的时间间隔能对你的数据进行快照（SnapShot ）存储。

AOF(Append Only File)持久化方式记录每次对服务器写的操作，当服务器重启的时候会重新执行这些命令来恢复原始的数据，AOF命令以redis协议追加保存每次写的操作到文件末尾，Redis还能对AOF文件进行后台重写,使得AOF文件的体积不至于过大。

https://redis.io/topics/persistence
http://www.redis.cn/topics/persistence.html

持久化-快照过程

快照是默认的持久化方式。这种方式是将内存中的数据以快照的方式写入到二进制文件中，默认的文件名为dump.rdb。可以通过配置自动做快照持久化的方式。可以对Redis 进行设置， 让它在“ N 秒内数据集至少有 M 个改动”这一条件被满足时， 自动保存一次数据集。你也可以通过调用 SAVE或者BGSAVE ， 手动让 Redis 进行数据集保存操作。

```
save 900 1
#900秒内如果超过1个key被修改，则发起快照保存
```

持久化-快照过程

Redis会单独创建fork子进程来进行持久化，会先将数据写入到一个临时文件中，待持久化过程成结束了，再用这个临时文件替换上次持久化好的文件。

整个过程，主进程不进行任何IO操作，这就确保了极高的性能。

如果需要大规模的数据恢复，且对于数据恢复的完整性不是非常敏感，那RDB方式要比AOF更加高效。

RDB缺点：最后一次持久化后的数据可能丢失。

持久化-如何触发RDB快照

1、配置文件中默认的快照配置，冷拷贝后重新使用

2、命令save或者bgsave

save：只管保存，其他阻塞

bgsave：后台异步进行快照操作，还可以处理客户端请求

3、执行flushall，也会产生dump.rdb文件，但里面是空的，无意义

持久化-如何恢复RDB快照

1、将备份文件(dump.rdb)移到安装目录并启动服务即可恢复

2、CONFIG GET dir 获取目录

持久化-AOF

由于快照方式是在一定时间间隔做一次的，所以如果redis意外down掉的话，就会丢失最后一次快照后的所有修改。

AOF比快照有更好的持久化性，是由于在使用aof的时候，redis会将每一个收到的命令都通过write函数追加到文件中，当redis重启时会通过重新执行文件中保存的写命令来在内存中重建整个数据库的内容。

保存在appendonly.aof文件中

持久化-AOF

appendonly yes #启用持久化方式

appendsync always #收到写命令就立即写入磁盘，最慢，但保证安全的持久化

appendsync eveysec #每秒钟写入磁盘一次，在性能和持久化方面做了很好的折中

appendsync no #完全依赖OS，性能最好，持久性没有保证

持久化-AOF

正常恢复

- 启动：设置yes，修改默认的appendonly no修改为yes
- 将aof文件复制到对应目录下
- 恢复：重启redis然后重新加载

异常恢复

- 启动：设置yes，修改默认的appendonly no修改为yes
- 将写坏的aof文件复制到对应目录下
- 修复：redis-check-aof –fix进行修复
- 恢复：重启redis然后重新加载

持久化-AOF Rewrite是什么

因为 AOF 的运作方式是不断地将命令追加到文件的末尾，所以随着写入命令的不断增加， AOF 文件的体积也会变得越来越大。当AOF文件的大小超过所设定的阈值时，redis就会启动AOF文件的内容压缩。执行bgrewriteof命令， Redis 将生成一个新的AOF文件， 这个文件包含重建当前数据集所需的最少命令。

持久化-AOF Rewrite原理

AOF 文件持续增长时，会fork出一条新进程来将文件重写，遍历新进程的内存中数据，每条记录有一条set语句。重写aof文件的操作，并没有读取旧的aof文件，而是将整个内存中的数据库内容用命令的方式重写了一个新的aof文件，这点和快照类似。

持久化-AOF Rewrite触发机制

Redis会记录上次重写时的AOF大小，默认配置适当AOF文件大小是上次rewrite后大小的一倍且文件大于64M时触发

持久化总结

RDB的优点

- RDB是一个非常紧凑的文件
- RDB在保存RDB文件时父进程唯一需要做的就是fork出一个子进程,接下来的工作全部由子进程来做，父进程不需要再做其他IO操作，所以RDB持久化方式可以最大化redis的性能.
- 与AOF相比,在恢复大的数据集的时候，RDB方式会更快一些

RDB的缺点

- 数据丢失风险大
- RDB 需要经常fork子进程来保存数据集到硬盘上,当数据集比较大的时候,fork的过程是非常耗时的,可能会导致Redis在一些毫秒级内不能响应客户端的请求

持久化总结

AOF的优点

- AOF文件是一个只进行追加的日志文件
- 使用AOF 会让你的Redis更加灵活: 你可以使用不同的fsync策略
- Redis 可以在 AOF 文件体积变得过大时，自动地在后台对 AOF 进行重写
- AOF 文件有序地保存了对数据库执行的所有写入操作， 这些写入操作以 Redis 协议的格式保存， 因此 AOF 文件的内容非常容易被人读懂， 对文件进行分析也很轻松

持久化总结

AOF的缺点

- 对于相同的数据集来说，AOF 文件的体积通常要大于 RDB 文件的体积
- 根据所使用的 fsync 策略，AOF 的速度可能会慢于 RDB

Redis 发布/订阅（Pub/Sub）

发布/订阅（Pub/Sub）是一种消息通信模式，主要是目的是解除消息发布者和消息订阅之间的耦合，Redis作为一个Pub/Sub的Server，在发布者和订阅者之间起到了消息路由的功能。订阅者可以通过SUBSCRIBE和PSUBSCRIBE 命令向server订阅自己感兴趣的消息类型，redis将信息类型称为频道(channel)。当发布者通过PUBLISH命令向server发送特定类型的信息时，订阅该信息类型的全部client都会收到此消息。

Redis 发布/订阅（Pub/Sub）

subscribe 订阅给定的一个或多个频道的信息

psubscribe 订阅一个或多个符合给定模式的频道

publish 信息 message 发送到指定的频道 channel

unsubscribe 指示客户端退订给定的频道

punsubscribe 指示客户端退订所有给定模式

Redis 发布/订阅（Pub/Sub）

```
127.0.0.1:6379> SUBSC
Reading messages...  (
1) "subscribe"
2) "cctv1"
3) (integer) 1
1) "message"
2) "cctv1"
3) "hello cctv1"
```

```
127.0.0.1:6379> SUBSCRIBE cctv1 cctv2
Reading messages...  (press Ctrl-C to quit)
1) "subscribe"
2) "cctv1"
3) (integer) 1
1) "subscribe"
2) "cctv2"
3) (integer) 2
1) "message"
2) "cctv1"
3) "hello cctv1"
1) "message"
```

```
127.0.0.1:6379> PUBLI
(integer) 2
```