程序猿大哥、

redis的RDB持久化

Redis是一个键值对数据库服务器，服务器中通常包含着任意个非空数据库，而每个非空数据库中又可以包含任意个键值对，为了方便起见，我们将服务器中的非空数据库以及它们的键值对统称为数据库状态

举个例子，图10-1 展示了一个包含三个非空数据库的Redis 服务器，这三个数据库以及数据库中的键值对就是该服务器的数据库状态

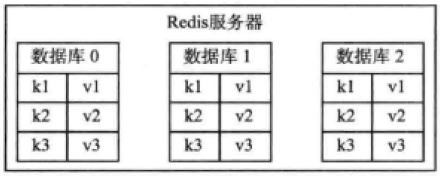

图 10-1 数据库状态示例

因为Redis 是内存数据库，它将自己的数据库状态储存在内存里面，所以如果不想办法将储存在内存中的数据库状态保存到磁盘里面，那么一旦服务器进程退出，服务器中的数据库状态也会消失不见

为了解决这个问题，Redis提供了RDB持久化功能，这个功能可以将Redis 在内存中的数据库状态保存到磁盘里面，避免数据意外丢失
RDB持久化既可以手动执行，也可以根据服务器配置选项定期执行，该功能可以将某个时间点上的数据库状态保存到一个RDB文件中，如图10-2所示
RDB 持久化功能所生成的RDB 文件是一个经过压缩的二进制文件，通过该文件可以还原生成RDB 文件时的数据库状态，如图1 0-3所示

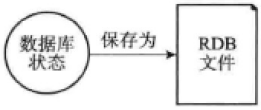

图 10-2 将数据库状态保存为 RDB 文件

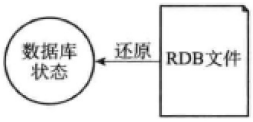

图 10-3 用 RDB 文件来还原数据库状态

因为RDB文件是保存在硬盘里面的，所以即使Redis服务器进程退出，甚至运行Redis服务器的计算机停机，但只要RDB文件仍然存在，Redis服务器就可以用它来还原数据库状态

RDB文件的创建与载入

有两个Redis命令可以用于生成RDB文件，一个是SAVE，另一个是BGSAVE
SAVE命令会阻塞Redis服务器进程，直到RDB文件创建完毕为止，在服务器进程阻塞期间，服务器不能处理任何命令请求

```
redis> SAVE  //等待直到RDB 文件创建完毕
OK
```

和SAVE命令直接阻塞服务器进程的做法不同，BGSAVE命令会派生出一个子进程，然后由子进程负责创建RDB 文件，服务器进程(父进程)继续处理命令请求

```
redis> BGSAVE //派生子进程，并由子进程创建RDB 文件
Background saving started
```

创建RDB文件的实际工作由rdb.c/rdbSave函数完成，SAVE命令和BGSAVE命令会以不同的方式调用这个函数，通过以下伪代码可以明显地看出这两个命令之间的区别

```
def SAVE()
#创建RDB 文件
rdbSave()
def BGSAVE()
#创建子进程
pid = fork()
```

昵称：程序猿大哥、
园龄：2年5个月
粉丝：3
关注：0
+加关注

< 2019年4月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签
redis(6)

文章分类
drools
dubbo
git
linux+shell
maven
mongo
python
redis(6)
zookeeper
多线程
工作历程
架构
框架
设计模式
数据结构
算法
随笔记录(1)
无聊随笔

最新评论

1. Re:redis的RDB持久化
简洁、明了、易理解，赞一个
--一肥惊人
2. Re:redis设置键的生存时间或过期时间

```
if pid==0
#子进程负责创建RDB 文件
rdbSave()
#完成之后向父进程发送信号
signal_parent ()
elif pid>0
#父进程继续处理命令请求，并通过轮询等待子进程的信号
handle_request_and_wait_signal()
else
#处理出错情况
handle_fork_error()
```

和使用SAVE命令或者BGSAVE命令创建RDB文件不同，RDB 文件的载入工作是在服务启动时自动执行的，所以Redis并没有专门用于载入RDB文件的命令，只要Redis服务器在启动时检测到RDB文件存在，它就会自动载入RDB文件
以下是Redis服务器启动时打印的日志记录，其中第二条日志DB loaded from disk就是服务器在成功载入RDB 文件之后打印的

```
$ redis-server
[7379] 30 Aug 21:07:01.270 # Server started, Redis version 2.9.11
[7379] 30 Aug 21:07:01.289 * DB loaded from disk : 0.018 seconds
[7379] 30 Aug 21:07:01.289 * Tbe server is now ready to accept connections on port 6379
```

另外值得一提的是，因为AOF文件的更新频率通常比RDB 文件的更新频率高，所以如果服务器开启了AOF持久化功能，那么服务器会优先使用AOF文件来还原数据库状态
只有在AOF持久化功能处于关闭状态时，服务器才会使用RDB 文件来还原数据库状态服务器判断该用哪个文件来还原数据库状态的流程如图10-4 所示
载入RDB文件的实际工作由rdb.c/rdbLoad函数完成，这个函数和rdbSave函数之间的关系可以用图10-5表示

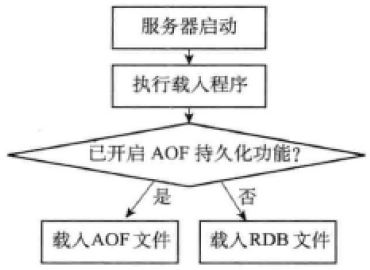

图 10-4 服务器载入文件时的判断流程

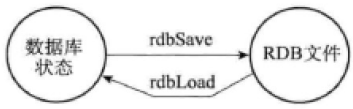

图 10-5 创建和载入 RDB 文件

SAVE命令执行时的服务器状态

前面提到过，当SAVE命令执行时，Redis服务器会被阻塞，所以当SAVE命令正在执行时，客户端发送的所有命令请求都会被拒绝
只有在服务器执行完SAVE命令、重新开始接受命令请求之后，客户端发送的命令才会被处理

BGSAVE命令执行时的服务器状态

因为BGSAVE命令的保存工作是由子进程执行的，所以在子进程创建RDB文件的过程中，Redis服务器仍然可以继续处理客户端的命令请求，但是，在BGSAVE命令执行期间，服务器处理SAVE、BGSAVE、BGREWRITEAOF三个命令的方式会和平时有所不同
首先，在BGSAVE命令执行期间，客户端发送的SAVE命令会被服务器拒绝，服务器禁止SAVE命令和BGSAVE命令同时执行是为了避免父进程(服务器进程)和子进程同时执行两个rdbSave调用，防止产生竞争条件
其次，在BGSAVE命令执行期间，客户端发送的BGSAVE命令会被服务器拒绝，因为同时执行两个BGSAVE 命令也会产生竞争条件
最后，BGREWRITEAOF和BGSAVE两个命令不能同时执行
如果BGSAVE命令正在执行，那么客户端发送的BGREWRITEAOF命令会被延迟到BGSAVE命令执行完毕之后执行
如果BGREWRITEAOF命令正在执行，那么客户端发送的BGSAVE命令会被服务器拒绝
因为BGREWRITEAOF和BGSAVE两个命令的实际工作都由子进程执行，所以这两个命令在操作方面并没有什么冲突的地方，不能同时执行它们只是一个性能方面的考虑——并
发出两个子进程，并且这两个子进程都同时执行大量的磁盘写入操作，这怎么想都不会是一个好主意

RDB 文件载入时的服务器状态

服务器在载入RDB 文件期间，会一直处于阻塞状态，直到载入工作完成为止

自动间隔性保存

因为BGSAVE命令可以在不阻塞服务器进程的情况下执行，所以Redis允许用户通过设置服务器配置的save选项，让服务器每隔一段时间自动执行一次BGSAVE命令

用户可以通过save选项设置多个保存条件，但只要其中任意一个条件被满足，服务器就会执行BGSAVE命令
举个例子，如果我们向服务器提供以下配置

图9-12在哪?
--bijian1013

```
save 900 1
save 300 10
save 60 10000
```

那么只要满足以下三个条件中的任意一个，BGSAVE命令就会被执行
服务器在900秒之内，对数据库进行了至少1次修改
服务器在300秒之内，对数据库进行了至少10次修
改

服务器在60秒之内，对数据库进行了至少10000次修改。

举个例子，以下是Redis服务器在60秒之内，对数据库进行了至少10000次修改之后，服务器自动执行BGSAVE命令时打印出来的日志：

```
[5085] 03 Sep 17:09:49.463 * 10000 changes in 60 seconds . Saving ...
[5085] 03 Sep 17:09:49.463 * Background saving started by pid 5189
[5189] 03 Sep 17:09:49.522 • DB saved on disk
[5189] 03 Sep 17:09:49.522 • RDB: 0 MB of memory used by copy-on-write
[5085] 03 Sep 17:09:49.563 • Background saving terminated with succes
```

设置保存条件

当Redis服务器启动时，用户可以通过指定配置文件或者传人启动参数的方式设置save选项，如果用户没有主动设置save选项，那么服务帮会为save选项设置默认条件：

```
save 900 1
save 300 10
save 60 10000
```

接着，服务器程序会根据save选项所设置的保存条件，设置服务器状态redisServer
结构的saveparams属性：

```
struct redisServer {
//...
//记录了保存条件的数组
struct saveparam *saveparams;
//...
};
```

saveparams属性是一个数组，数组中的每个元素都是一个saveparam结构，每个
saveparam结构都保存了一个save选项设置的保存条件：

```
struct saveparam {
//秒数
time_t seconds;
//修改数
int changes;
};
```

比如说，如果save选项的值为以下条件：

```
save 900 1
save 300 10
save 60 10000
```

那么服务器状态中的saveparams数组将会是图10-6所示的样子。

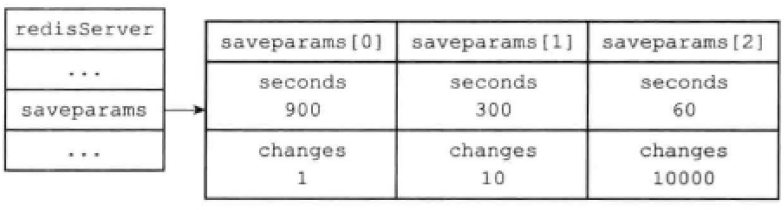

图 10-6 服务器状态中的保存条件

dirty计数器和lastsave属性

除了saveparams数组之外，服务器状态还维持着一个dirty计数器，以及一个lastsave属性：

dirty计数器记录距离上一次成功执行SAVE命令或者BGSAVE命令之后，服务器对数据库状态(服务器中的所有数据库)进行了多少次修改(包括写入、删除、更新等操作)。
lastsave属性是一个UNIX时间戳，记录了服务器上一次成功执行SAVE命令或者BGSAVE命令的时间。

当服务器成功执行一个数据库修改命令之后，程序就会对dirty计数器进行更新:命令修改了多少次数据库.dirty计数器的值就增加多少。
例如，如果我们为一个字符串键设置值:

```
redis>SET message "hello"
OK
```

那么程序会将dirty计数器的值增加1。
又例如，如果我们向一个集合键增加三个新元素:

```
redis>SADD database Redis MongoDB MariaDB
(integer) 3
```

那么程序会将dirty计数器的值增加3.

分类: redis

标签: redis

程序猿大哥、
关注 - 0
粉丝 - 3
+加关注

1 0

« 上一篇: redis数据库通知
» 下一篇: redis的AOF持久化

posted @ 2016-12-01 17:44 程序猿大哥、 阅读(5097) 评论(1) 编辑 收藏

评论

#1楼 2018-11-05 13:42 | 一肥惊人

简洁、明了、易理解，赞一个

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。