[**Redis主从复制**](http://blog.csdn.net/zouyujie1127/article/details/41411195)

2014-11-23 15:18 4661人阅读 [评论](http://blog.csdn.net/zouyujie1127/article/details/41411195#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/zouyujie1127/article/details/41411195#report)

本文章已收录于：

[[](http://lib.csdn.net/base/34?source=blogtop) **Redis知识库**](http://lib.csdn.net/base/34?source=blogtop)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

ASP.NET WebForm开发（77） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg ASP.NET 分布式开发（5） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。博客新家：http://www.cnblogs.com/jiekzou/

目录[(?)](http://blog.csdn.net/zouyujie1127/article/details/41411195" \o "系统根据文章中H1到H6标签自动生成文章目录)[[+]](http://blog.csdn.net/zouyujie1127/article/details/41411195)

[**Redis**](http://lib.csdn.net/base/redis)的主从复制功能非常强大，一个master可以拥有多个slave，而一个slave又可以拥有多个slave，如此下去，形成了强大的多级服务器集群**[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "_blank)**.

实现步骤如下:

**1.在Windows某个磁盘上创建两个目录,例如; MasterRedis(存储的是Master服务) SlaveRedis(存储的是Slave服务).**

**2.在Master服务中的配置文件redis.conf修改 :bind 127.0.0.1.**

**3.在Slave服务中的配置文件redis.conf修改：**

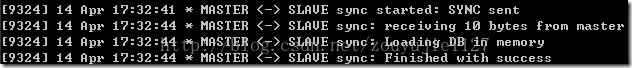
      port 6381（服务端口号要分开）

      bind 127.0.0.1

      slaveof 127.0.0.1 6379 （设置master的Host以及Port）

**4.分别启动Master服务与Slave服务。**

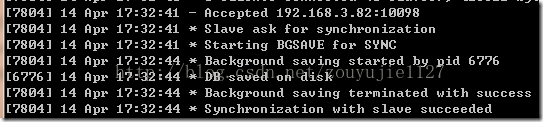
注意到，当我启动master，然后启动一个slave的时候，可以发现slave上：



会发送一个SYNC请求，从Master上面进行相应，而且它支持自动重连，

即当master掉线的情况下，它会处于等待请求的状态。

而Master上：



第一次Slave向Master同步的实现是：Slave向Master发出同步请求，Master先dump出rdb文件，然后将rdb文件全量传输给slave，然后Master把缓存的命令转发给Slave，初次同步完成。第二次以及以后的同步实现是：Master将变量的快照直接实时依次发送给各个Slave。不管什么原因导致Slave和Master断开重连都会重复以上过程。Redis的主从复制是建立在内存快照的持久化基础上，只要有Slave就一定会有内存快照发生。虽然Redis宣称主从复制无阻塞，但由于Redis使用单线程服务，如果Master快照文件比较大，那么第一次全量传输会耗费比较长时间，且文件传输过程中Master可能无法提供服务，也就是说服务会中断

**Redis数据快照**

数据快照的原理是将整个Redis内存中的所有的数据遍历一遍存储到一个扩展名为rdb的数据文件中，通过save命令

可以调用这个过程。数据快照配置如下:

Save 900 1

Save 300 10

Save 60 10000

以上在redis.conf中的配置指出在多长时间内，有多少次更新操作，就将数据同步到数据文件中，这个可以多个条件进行配合，上面的含义是900秒后有一个key发生改变就执行save,300秒后有10个key发生改变就执行save,60秒有10000个key发生改变就执行save.

数据快照的缺点是持久化之后如果出现系统宕机则会丢失一段数据，因此增加了另外一种追加式的操作日志记录，叫append only file,其日志文件以aof结尾，我们称为aof文件，要开启aof日志的记录，需要在配置文件中进行如下配置: appendonly yes

Appendonly配置不开启，可能在断电时导致一段时间的数据丢失，因为redis本身同步数据文件时按save条件来同步的，所以有的数据会在一段时间内只存在于内存中。

Appendfsync no/always/everysec

no:表示等**[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "_blank)**进行数据缓存同步到磁盘。性能最好，持久化没有保障。

Always:表示每次更新操作后手动调用fsync()将数据写到磁盘.每次收到写命令就立即强制写入磁盘，最慢的，但是保障完全的持久化。

Everysec:表示每秒同步一次.每秒钟强制写入磁盘一次，在性能和持久化方面做了很好的折中。

为了定时减小AOF文件的大小，Redis2.4以后增加了自动的bgrewriteaof的功能，Redis会选择一个自认为负载低的情况下执行bgrewriteaof,这个重写AOF文件的过程是很影响性能的。解决方案：Master关闭Save功能，关闭AOF日志功能，以求达到性能最佳。Slave开启Save并开启AOF日志功能，并开启bgrewriteaof功能，不对外提供服务，这样Slave的负载总体上会高于Master负载，但是Master性能达到最好.

Bgrewriterof内部实现：

 1.Redis通过fork一个子进程，遍历数据，写入新临时文件

 2.父进程继续处理client请求，子进程继续写临时文件。

 3.父进程把新写入的AOF写在缓冲区。

 4.子进程写完退出，父进程接收退出消息，将缓冲区AOF写入临时文件。

 5.临时文件重命名成appendonly.aof,原来文件被覆盖，整个过程完成。

**Redis数据恢复**

当Redis服务器挂掉以后，重启时将按以下优先级恢复数据到内存：

1.如果只配置了AOF，重启时加载AOF文件恢复数据。

2.如果同时配置了RBD和AOF，启动时只加载AOF文件恢复数据。

3.如果只配置了RDB，启动时将加载dump文件恢复数据。