Redis数据库主从复制

**一、概述：**

在分布式系统中为了解决单点故障问题，通常会把数据复制多个副本部署到不同的机器中，满足故障恢复和负载均衡等需求。Redis也是如此，它为我们提供了复制功能，实现了相同数据的多个Redis副本。复制功能是高可用Redis的基础，Redis的哨兵集群模式都是在主从复制模式的基础上实现的。复制也是Redis日常运维的常见维护点。因此深刻理解复制的工作原理与使用技巧对日常的运维非常有帮助。

1. **常用拓扑结构；**
2. 一主一从结构

一主一从结构是最简单的复制拓扑结构，用于主节点出现宕机时从节点提供故障转移支持。当应用写命令并发量较高且需要持久化时，可以只在从节点上开启AOF（即Append Only File持久化），将Redis执行的每次写命令记录到单独的日志文件中（有点像MySQL的binlog）；这样既保证数据安全性同时，也避免了持久化对主节点的性能干扰。

Redis-A

Redis-B

1. 一主多从结构

一主多从结构（又称为星型拓扑结构）使得应用端可以利用多个从节点实现读写分离。对于读占比较大的场景，可以把读命令发送到从节点来分担主节点压力。同时在日常开发中如果需要执行一些比较耗时的读命令，可以在其中一台从节点上执行，防止慢查询（**超过指定时间的SQL语句查询称为慢查询**）对主节点造成阻塞从而影响线上服务的稳定性。对于写并发量较高的场景，多个从节点会导致主节点写命令的多次发送从而过度消耗网络带宽，同时也加重了主节点的负载影响服务稳定性。

Redis-C

Redis-E

Redis-D

Redis-B

Redis-A

1. 树状主从结构（级联结构）

树状结构使得从节点不但可以复制主节点数据，同时可以作为其他从节点的主节点继续向下层复制。通过引入中间复制层，可以有效降低主节点负载和需要传送给从节点的数据量。当主节点需要挂载多个从节点时为了避免对主节点的性能干扰，可以采用树状主从结构降低主节点压力。

Redis-C

Redis-E

Redis-A

Redis-B

Redis-D

1. **主从复制原理；**
2. **过程**

在从节点执行slaveof命令后，复制过程便开始运作，大致分为6个过程：

1. 保存主节点信息

执行slaveof（slaveof命令可以将当前服务器转变为指定服务器的从属服务器(slave server)）命令后从节点只保存主节点的地址信息便直接返回，这时建立复制流程还没有开始，在从节点执行info replication命令可以看到主节点的相关信息，并在日志中记录复制启动信息

1. 主从建立socket连接

从节点内部通过每秒运行的定时任务维护复制相关逻辑，当定时任务发现存在新的主节点后，会尝试与该节点建立网络连接。

从节点会建立一个socket套接字，专门用于接收主节点发送的复制命令。如果从节点无法建立连接，定时任务会无限重试，直到连接成功或者执行slaveof no one命令取消复制

1. 发送ping命令

连接建立成功后，从节点发送ping请求进行首次通信，目的在于：

* 检测主从之间网络套接字是否可用。
* 检测主节点当前是否可接受处理命令。

如果发送ping命令后，从节点没有收到主节点回复pong或者超时未回复，从节点会断开复制连接，等待下次定时任务发起重连。

1. 权限验证

如果主节点设置了requirepass（需要密码）参数，则需要密码验证，从节点必须配置masterauth（主节点验证）参数保证与主节点相同的密码才能通过验证。如果验证失败复制将终止，从节点重新发起复制流程。

1. 同步数据集

主从复制连接正常通信后，对于首次建立复制的场景，主节点会把所有的数据全部发送给从节点，这部分操作是耗时最长的步骤。在同步过程中会分为两种情况：全量同步和部分同步

1. 命令持续复制

当主节点把当前的数据同步给从节点后，便完成了复制建立的流程。接下来主节点会持续地把写命令发送给从节点，保证数据的一致性。

1. **数据同步**

Redis在2.8及以上版本使用psync（同步）命令完成主从数据，同步过程分为：全量复制和部分复制。

* **全量复制：**一般用于初次复制的场景，Redis早期支持的复制功能只有全量复制，他会把主节点全部数据一次性发送给从节点，当数据量较大时，会对主从节点和网络造成很大的开销。
* **部分复制：**用于处理在主从复制中因网络闪断等原因造成的数据丢失场景，当从节点再次连上主节点后，如果条件允许，主节点会补发丢失数据给从节点。而补发的数据量远远小于全量数据，可以有效避免全量复制的过高开销。

部分复制是对老版复制的重大优化，有效避免了不必要的全量复制操作。因此当使用复制功能时，尽量采用2.8以上版本的Redis。

psync命令运行时需要以下组件的支持

* 主从节点各自的复制偏移量。
* 主节点复制积压缓冲区。
* 主节点运行id。

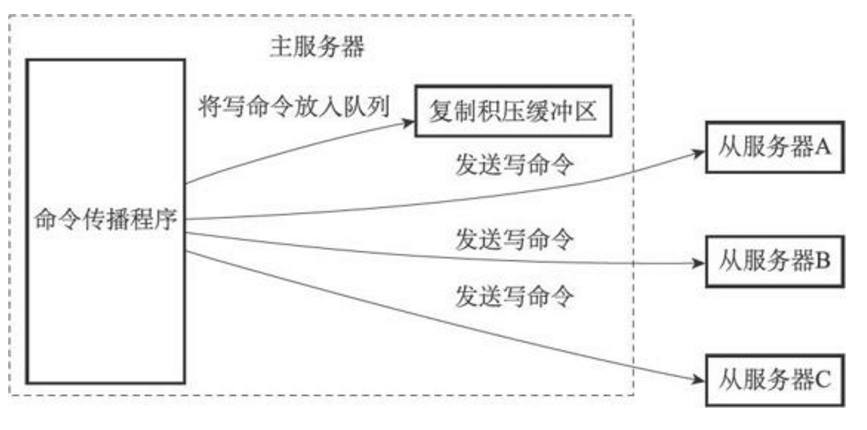
1. 复制偏移量

参与复制的主从节点都会维护自身复制偏移量。主节点在处理完写入命令后，会把命令的字节长度做累加记录，统计信息在info replication中的master\_repl\_offset指标中

从节点每秒钟上报自身的复制偏移量给主节点，因此主节点也会保存从节点复制偏移量。并且从节点在收到主节点发送的命令后，也会累加记录自身的偏移量，记录在自己的统计信息中。

1. 复制积压缓冲区

复制积压缓冲区是保存在主节点上的一个固定长度的队列，默认大小为1MB，当主节点进行主从复制时，写命令不但会发送给从节点，还会写入复制积压缓冲区中。



缓冲区本质上是一个先进先出的定长队列，可以实现保存最近已复制数据的功能，用于部分复制和复制命令丢失的数据补救。统计信息记录于info replication中

repl\_backlog\_active:1 //开启复制缓冲区

repl\_backlog\_size:1048576 //缓冲区最大大小

repl\_backlog\_first\_byte\_offset:7479 //起始偏移量

repl\_backlog\_histlen:1048576 //已保存数据的有效长度

1. 主节点运行ID

每个Redis节点启动后，都会动态分配一个运行ID，用来唯一识别一个Redis节点。从节点通过保存主节点的运行ID，来判断自己正在向谁进行复制，当主的运行ID改变后，从节点将进行全量复制。

注意：Redis关闭在启动时，运行ID会随之改变，将导致从节点全量复制

若不希望从节点对主节点进行全量复制，则需要保持主节点ID不变，而当修改配置文件需要重启操作时，可以对主节点使用redis-cli debug reload命令来进行对配置文件的重新加载，而不用关闭Redis。但是，debug reload命令会阻塞Redis主进程（阻塞主的写不阻塞从的读取），阻塞期间会生成本地RDB（是把当前内存中的数据集快照写入磁盘，也就是 Snapshot 快照）快照并清空数据之后再加载RDB文件。因此对于无法容忍阻塞的场景，谨慎使用，对于主要是读取的环境，影响较小

**四、实验配置**

**实验环境：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 系统 | IP地址 | 主机名 | 所需软件 | Redis角色及端口 |
| Centos 7.5 | 192.168.200.111 | master.redis.com | redis-4.0.9.tar.gz | master：6379 |
| Centos 7.5 | 192.168.200.112 | slave1.redis.com | redis-4.0.9.tar.gz | slave：6379  slave：6380 |

**实验步骤：**

* 安装并配置master角色的redis服务；
* 安装并配置slave角色的redis服务（双实例）；
* 配置master角色的文件；
* 配置slave角色的文件；
* 验证主从复制；
* **安装并配置master角色的redis服务；**

[root@master ~]# tar zxvf redis-4.0.9.tar.gz -C /usr/src

[root@master ~]# cd /usr/src/redis-4.0.9

包里已经有Makefile文件，无须配置直接编译安装即可

[root@master redis-4.0.9]# make

[root@master redis-4.0.9]# make install

[root@master redis-4.0.9]# cd utils/

[root@master utils]# ./install\_server.sh

#执行安装脚本，一直回车接受默认值，端口默认6379

redis配置文件为/etc/redis/6379.conf,安装向导时可以更改，不同的实例有不同的配置文件

**6379.conf 配置项说明如下**：

1. Redis默认不是以守护进程的方式运行，可以通过该配置项修改，使用yes启用守护进程

    daemonize no

2. 当Redis以守护进程方式运行时，Redis默认会把pid写入/var/run/redis\_6379.pid文件，可以通过pidfile指定(6379是端口号，不同的端口号有不同的pid文件）

    pidfile /var/run/redis\_6379.pid

3. 指定Redis监听端口，默认端口为6379，作者在自己的一篇博文中解释了为什么选用6379作为默认端口，因为6379在手机按键上MERZ对应的号码，而MERZ取自意大利歌女Alessia Merz的名字

    port 6379

4. 绑定的主机地址，默认为127.0.0.1，需要更改为真实网卡地址

    bind 127.0.0.1

5.当 客户端闲置多长时间后关闭连接，如果指定为0，表示关闭该功能

    timeout 300

6. 指定日志记录级别，Redis总共支持四个级别：debug、verbose、notice、warning，默认为verbose

    loglevel verbose

7. 日志记录方式，默认为标准输出，如果配置Redis为守护进程方式运行，而这里又配置为日志记录方式为标准输出，则日志将会发送给/dev/null

    logfile stdout

8. 设置数据库的数量，默认数据库为0，可以使用SELECT <dbid>命令在连接上指定数据库id

    databases 16

9. 指定在多长时间内，有多少次更新操作，就将数据同步到数据文件，可以多个条件配合

    save <seconds> <changes>

    Redis默认配置文件中提供了三个条件：

    save 900 1

    save 300 10

    save 60 10000

    分别表示900秒（15分钟）内有1个更改，300秒（5分钟）内有10个更改以及60秒内有10000个更改。

10. 指定存储至本地数据库时是否压缩数据，默认为yes，Redis采用LZF压缩，如果为了节省CPU时间，可以关闭该选项，但会导致数据库文件变的巨大

    rdbcompression yes

11. 指定本地数据库文件名(rdb半持久化保存在本地的文件），默认值为dump.rdb

    dbfilename dump.rdb

12. 指定本地数据库存放目录（rdb文件存放目录）

    dir /var/lib/redis/6379

13. 设置当本机为slave服务时，设置master服务的IP地址及端口，在Redis启动时，它会自动从master进行数据同步

    slaveof <masterip> <masterport>

14. 当master服务设置了密码保护时，slave服务连接master的密码

    masterauth <master-password>

15. 设置Redis连接密码，如果配置了连接密码，客户端在连接Redis时需要通过AUTH <password>命令提供密码，默认关闭

    requirepass foobared

16. 设置同一时间最大客户端连接数，默认无限制，Redis可以同时打开的客户端连接数为Redis进程可以打开的最大文件描述符数，如果设置 maxclients 0，表示不作限制。当客户端连接数到达限制时，Redis会关闭新的连接并向客户端返回max number of clients reached错误信息

    maxclients 128

17. 指定Redis最大内存限制，Redis在启动时会把数据加载到内存中，达到最大内存后，Redis会先尝试清除已到期或即将到期的Key，当此方法处理 后，仍然到达最大内存设置，将无法再进行写入操作，但仍然可以进行读取操作。Redis新的vm机制，会把Key存放内存，Value会存放在swap区

    maxmemory <bytes>

18. 指定是否在每次更新操作后进行日志记录（是否开启aof功能），Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。因为 redis本身同步数据文件是按上面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。默认为no，更改为yes则开启

    appendonly no

19. 指定更新日志文件名（aof文件名），默认为appendonly.aof，存放目录与rdb的一样

     appendfilename appendonly.aof

20. 指定更新日志条件，共有3个可选值：

    no：表示等操作系统进行数据缓存同步到磁盘（快）

    always：表示每次更新操作后手动调用fsync()将数据写到磁盘（慢，安全）

    everysec：表示每秒同步一次（折衷，默认值）

    appendfsync everysec

21. 指定是否启用虚拟内存机制，默认值为no，简单的介绍一下，VM机制将数据分页存放，由Redis将访问量较少的页即冷数据swap到磁盘上，访问多的页面由磁盘自动换出到内存中。后面有若干vm相关项，略过，保持默认

     vm-enabled no

22. 指定包含其它的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各个实例又拥有自己的特定配置文件

    include /path/to/local.conf

**更改配置**

[root@master ~]# sed -i '/^bind 127.0.0.1$/s/127.0.0.1/192.168.200.111/g' /etc/redis/6379.conf ##设置服务器监听地址

[root@master ~]# sed -i '/protected-mode/s/yes/no/g' /etc/redis/6379.conf

##关闭redis的保护模式,允许远程连接此服务器

[root@master ~]# sed -i '/daemonize/s/no/yes/g' /etc/redis/6379.conf

##开启redis的后台守护进程模式

[root@master ~]# sed -i '/requirepass/s/foobared/123456/g' /etc/redis/6379.conf

##设置redis的密码为123456

[root@master ~]# sed -i '/requirepass 123456/s/^# //g' /etc/redis/6379.conf

##开启redis的密码

[root@master utils]# /etc/init.d/redis\_6379 restart

Stopping ...

Redis stopped

Starting Redis server...

如果再次**重启redis-server需要先杀死进程再删除pid**

killall -9 redis-server

rm -f /var/run/redis\_6379.pid

/etc/init.d/redis\_6379 start

[root@master ~]# netstat -anpt |grep redis

tcp 0 192.168.200.111:6379 0.0.0.0:\* LISTEN 4204/redis-server 1

[root@master ~]#redis-cli -h 192.168.200.111 -a 123456 -p 6379

192.168.200.111:6379> exit

* **安装并配置slave角色的redis服务（双实例）；**

[root@slave1 ~]# tar zxvf redis-4.0.9.tar.gz -C /usr/src

[root@slave1 ~]# cd /usr/src/redis-4.0.9

包里已经有Makefile文件，无须配置直接编译安装即可

[root@slave1 redis-4.0.9]# make

[root@slave1 redis-4.0.9]# make install

[root@slave1 redis-4.0.9]# cd utils/

[root@slave1 utils]# ./install\_server.sh

#执行安装脚本，一直回车接受默认值，端口默认6379

#再次执行安装脚本，端口设置为6380，然后一直回车接受默认值，安装第二个实例

[root@slave1 ~]# sed -i '/^bind 127.0.0.1$/s/127.0.0.1/192.168.200.112/g' /etc/redis/6379.conf

[root@slave1 ~]# sed -i '/^bind 127.0.0.1$/s/127.0.0.1/192.168.200.112/g' /etc/redis/6380.conf

##更改监听地址

[root@slave1 ~]# sed -i '/protected-mode/s/yes/no/g' /etc/redis/6379.conf

[root@slave1 ~]# sed -i '/protected-mode/s/yes/no/g' /etc/redis/6380.conf

##关闭redis的保护模式

[root@slave1 ~]# sed -i '/daemonize/s/no/yes/g' /etc/redis/6379.conf

[root@slave1 ~]# sed -i '/daemonize/s/no/yes/g' /etc/redis/6380.conf

##开启redis的后台守护进程模式

[root@slave1 ~]# sed -i '/requirepass/s/foobared/123456/g' /etc/redis/6379.conf

[root@slave1 ~]# sed -i '/requirepass/s/foobared/123456/g' /etc/redis/6380.conf

##设置redis的密码为123456

[root@slave1 ~]# sed -i '/requirepass 123456/s/^#//g' /etc/redis/6379.conf

[root@slave1 ~]# sed -i '/requirepass 123456/s/^#//g' /etc/redis/6380.conf

##开启redis的密码

[root@slave1 ~]# /etc/init.d/redis\_6379 restart

Starting Redis server...

Redis is running...

如不能正常启动，先杀死进程再删除pid文件再启动

killall -9 redis-server

rm -f /var/run/redis\_63\*

[root@slave1 ~]# /etc/init.d/redis\_6380 restart

Starting Redis server...

Redis is running...

[root@ slave1 ~]#redis-cli -h 192.168.200.112 -a 123456 -p 6379

192.168.200.112:6379> exit

[root@ slave1 ~]#redis-cli -h 192.168.200.112 -a 123456 -p 6380

192.168.200.112:6380> exit

* **配置master角色的文件；**

[root@master ~]# sed -i 's/# min-slaves-to-write 3/min-slaves-to-write 2/g' /etc/redis/6379.conf

##设置slave节点的数量，如果slave节点数量少于此值，那么master节点将停止客户端的一切写请求

[root@master ~]#sed -i 's/# min-slaves-max-lag 10/min-slaves-max-lag 10/g' /etc/redis/6379.conf

##master与slave之间同步数据的超时时间，若超过此时间，master节点将停止客户端的一切写操作

[root@master ~]# killall -9 redis-server

[root@master ~]# rm -f /var/run/redis\_63\*

[root@master ~]# /etc/init.d/redis\_6379 start

Starting Redis server...

* **配置slave角色的文件；**

[root@slave1 ~]# sed -i '281s/^\.\*/slaveof 192.168.200.111 6379\n/g' /etc/redis/6379.conf

[root@slave1 ~]# sed -i '281s/^\.\*/slaveof 192.168.200.111 6379\n/g' /etc/redis/6380.conf

##指定master的ip地址以及端口

[root@slave1 ~]# sed -i '288s/^\.\*/masterauth 123456\n/g' /etc/redis/6379.conf

[root@slave1 ~]# sed -i '288s/^\.\*/masterauth 123456\n/g' /etc/redis/6380.conf

##指定master的连接密码

[root@master ~]# killall -9 redis-server

[root@master ~]# rm -f /var/run/redis\_63\*

[root@master ~]# /etc/init.d/redis\_6379 start

Starting Redis server...

先只启动一台从redis，测试从节点至少需要两台才可以

* **验证主从复制；**

[root@master ~]# redis-cli -h 192.168.200.111 -a 123456 -p 6379

##在master节点上创建键值对

192.168.200.111:6379> set name xiaoming

(error) NOREPLICAS Not enough good slaves to write.

然后再将从节点6380实例启动再测试

192.168.200.111:6379> set name xiaoming

OK

192.168.200.111:6379> keys \*

1) "name"

192.168.200.111:6379> get name

"xiaoming"

192.168.200.111:6379> info replication

##查看复制信息

# Replication

role:master

connected\_slaves:2

min\_slaves\_good\_slaves:2

slave0:ip=192.168.200.112,port=6380,state=online,offset=522,lag=1

slave1:ip=192.168.200.112,port=6379,state=online,offset=522,lag=1

master\_replid:46ec2c7e971d337d060d183d3a0c1313c5dd1683

master\_replid2:0000000000000000000000000000000000000000

master\_repl\_offset:522

second\_repl\_offset:-1

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:1

repl\_backlog\_histlen:522

192.168.200.111:6379> exit

[root@slave1 ~]# redis-cli -h 192.168.200.112 -p 6379 -a 123456

##登录slave节点验证键值同步情况，并测试无法写入

192.168.200.112:6379> keys \*

1) "name"

192.168.200.112:6379> get name

"xiaoming"

192.168.200.112:6380> set name1 xiaohong

(error) READONLY You can't write against a read only slave.

192.168.200.112:6379> exit

[root@slave1 ~]# redis-cli -h 192.168.200.112 -p 6380 -a 123456

192.168.200.112:6380> keys \*

1) "name"

192.168.200.112:6380> get name

"xiaoming"

192.168.200.112:6380> exit

说明已经同步，并且只有在主上可写，主从复制完成

主从复制成功后保存好快照，以便于哨兵集群继续实验使用