redis数据恢复

redis提供两种数据恢复的方法：

1.使用redis-dump

流程如下：

redis-dump > data.json ； data.json > redis-load

利用 JSON 作为中间转换完成

2.使用redis持久化中的RDB 和 AOF文件恢复

假设场景 master+slave的环境， master突然宕机（模拟灾难）

流程如下：

slave save（bgsave）保存更新RDB和AOF文件，并打包 data.tar

master删除原有的RDB和AOF文件，并将data.tar打包到master的bin(指的是redis的 bin)目录下

利用slave备份的RDB和AOF文件重启redis-server服务，达到数据恢复的目的

---

redis的备份和还原，借助了第三方的工具，redis-dump

1、安装redis-dump

复制代码代码如下：

```
[root@localhost tank]# yum install ruby rubygems ruby-devel  //安装rubygems 以及相关包
[root@localhost tank]# gem sources -a http://ruby.taobao.org/  //源，加入淘宝，外面的源不能访问
http://ruby.taobao.org/ added to sources
[root@localhost tank]# gem install redis-dump -V  //安装redis-dump
```

2、redis-dump导出数据

复制代码代码如下：

```
[root@localhost tank]# telnet 127.0.0.1 6379 //telnet到redis
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
set test 11 //设置一个值
+OK
get test //取值
$2
11

[root@localhost tank]# redis-dump -u 127.0.0.1:6379 >test.json //导出数据
```

值得注意的是，如果Redis服务器是需要密码认证的，那么要使用如下的方式进行操作：

```
#导出，密码前面要加一个冒号
redis-dump -u :password@xxx.xxx.xxx.xxx:6379 > redis.json
#导入
cat redis.json | redis-load -u :password@localhost
```

3、redis-load还原数据

复制代码代码如下：

```
[root@localhost tank]# telnet 127.0.0.1 6379 //telnet到redis
```

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
flushall //请空所有数据
+OK
keys * //查看已清空
*0

[root@localhost tank]# < test.json redis-load //导入数据

[root@localhost tank]# telnet 127.0.0.1 6379
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
keys * //已导入成功
*1
$4
test
```

利用 快照 和 同步 实现master和 slave的数据恢复

首先，修改Master上的如下配置：

```
$ sudo vim /opt/redis/etc/redis_6379.conf

#save 900 1 #禁用Snapshot
#save 300 10
#save 60 10000

appendonly no #禁用AOF
```

接着，修改Slave上的如下配置：

```
$ sudo vim /opt/redis/etc/redis_6379.conf

save 900 1 #启用Snapshot
save 300 10
save 60 10000

appendonly yes #启用AOF
appendfilename appendonly.aof #AOF文件的名称
# appendfsync always
appendfsync everysec #每秒钟强制写入磁盘一次
# appendfsync no

no-appendfsync-on-rewrite yes    #在日志重写时，不进行命令追加操作
auto-aof-rewrite-percentage 100 #自动启动新的日志重写过程
auto-aof-rewrite-min-size 64mb  #启动新的日志重写过程的最小值
```

分别启动Master与Slave

```
$ /etc/init.d/redis start
```

启动完成后在Master中确认未启动Snapshot参数

```
redis 127.0.0.1:6379> CONFIG GET save
1) "save"
2) ""
```

然后通过以下脚本在Master中生成25万条数据：

```
dongguo@redis:/opt/redis/data/6379$ cat redis-cli-generate.temp.sh

#!/bin/bash

REDISCLI="redis-cli -a slavepass -n 1 SET"
ID=1

while(($ID<50001))
do
  INSTANCE_NAME="i-2-$ID-VM"
  UUID=`cat /proc/sys/kernel/random/uuid`
  PRIVATE_IP_ADDRESS=10.`echo "$RANDOM % 255 + 1" | bc`.`echo "$RANDOM % 255 + 1" |
bc`.`echo "$RANDOM % 255 + 1" | bc`\
  CREATED=`date "+%Y-%m-%d %H:%M:%S"`
```

```
  $REDISCLI vm_instance:$ID:instance_name "$INSTANCE_NAME"
  $REDISCLI vm_instance:$ID:uuid "$UUID"
  $REDISCLI vm_instance:$ID:private_ip_address "$PRIVATE_IP_ADDRESS"
  $REDISCLI vm_instance:$ID:created "$CREATED"

  $REDISCLI vm_instance:$INSTANCE_NAME:id "$ID"

  ID=$(($ID+1))
done
```

```
dongguo@redis:/opt/redis/data/6379$ ./redis-cli-generate.temp.sh
```

在数据的生成过程中，可以很清楚的看到Master上仅在第一次做Slave同步时创建了dump.rdb文件，之后就通过增量传输命令的方式给Slave了。
dump.rdb文件没有再增大。

```
dongguo@redis:/opt/redis/data/6379$ ls -lh
total 4.0K
-rw-r--r-- 1 root root 10 Sep 27 00:40 dump.rdb
```

而Slave上则可以看到dump.rdb文件和AOF文件在不断的增大，并且AOF文件的增长速度明显大于dump.rdb文件。

```
dongguo@redis-slave:/opt/redis/data/6379$ ls -lh
total 24M
-rw-r--r-- 1 root root 15M Sep 27 12:06 appendonly.aof
-rw-r--r-- 1 root root 9.2M Sep 27 12:06 dump.rdb
```

```
redis 127.0.0.1:6379> info
```

信息太多只显示后三条

```
role:master
slave0:10.6.1.144,6379,online
db1:keys=250000,expires=0
```

当前的数据量为25万条key，占用内存31.52M。

然后我们直接Kill掉Master的Redis进程，模拟灾难。

```
dongguo@redis:/opt/redis/data/6379$ sudo killall -9 redis-server
```

我们到Slave中查看状态：

```
redis 127.0.0.1:6379> info
```

信息太多只显示后三条

```
master_link_down_since_seconds:25
slave_priority:100
db1:keys=250000,expires=0
```

可以看到master_link_status的状态已经是down了，Master已经不可访问了。
而此时，Slave依然运行良好，并且保留有AOF与RDB文件。

下面我们将通过Slave上保存好的AOF与RDB文件来恢复Master上的数据。

首先，将Slave上的同步状态取消，避免主库在未完成数据恢复前就重启，进而直接覆盖掉从库上的数据，导致所有的数据丢失。

（这时候slave要先 save）

```
redis 127.0.0.1:6379> SLAVEOF NO ONE
OK
```

确认一下已经没有了master相关的配置信息：

```
redis 127.0.0.1:6379> INFO
```

信息太多只显示后三条

```
aof_buffer_length:0
aof_pending_bio_fsync:0
db1:keys=250000,expires=0
```

在Slave上复制数据文件：

```
dongguo@redis-slave:/opt/redis/data/6379$ tar cvf /home/dongguo/data.tar *
appendonly.aof
dump.rdb
```

将data.tar上传到Master上，尝试恢复数据：
可以看到Master目录下有一个初始化Slave的数据文件，很小，将其删除。

```
dongguo@redis:/opt/redis/data/6379$ ls -l
total 4
```

```
-rw-r--r-- 1 root root 10 Sep 27 00:40 dump.rdb
dongguo@redis:/opt/redis/data/6379$ sudo rm -f dump.rdb
```

然后解压缩数据文件：

```
dongguo@redis:/opt/redis/data/6379$ sudo tar xf /home/dongguo/data.tar
dongguo@redis:/opt/redis/data/6379$ ls -lh
total 29M
-rw-r--r-- 1 root root 18M Sep 27 01:22 appendonly.aof
-rw-r--r-- 1 root root 12M Sep 27 01:22 dump.rdb
```

启动Master上的Redis；

```
dongguo@redis:/opt/redis/data/6379$ sudo /etc/init.d/redis start
Starting Redis server...
```

查看数据是否恢复：

```
redis 127.0.0.1:6379> INFO

role:master
db1:keys=250000,expires=0
```

可以看到25万条数据已经完整恢复到了Master上。

此时，可以放心的恢复Slave的同步设置了。

```
redis 127.0.0.1:6379> SLAVEOF 10.6.1.143 6379
OK
```

查看同步状态：

```
redis 127.0.0.1:6379> INFO
```

master_link_status显示为up，同步状态正常。

在此次恢复的过程中，我们同时复制了AOF与RDB文件，那么到底是哪一个文件完成了数据的恢复呢？
实际上，当Redis服务器挂掉时，重启时将按照以下优先级恢复数据到内存：

1. 如果只配置AOF,重启时加载AOF文件恢复数据；
2. 如果同时 配置了RDB和AOF,启动是只加载AOF文件恢复数据；
3. 如果只配置RDB,启动是将加载dump文件恢复数据。

也就是说，AOF的优先级要高于RDB，这也很好理解，因为AOF本身对数据的完整性保障要高于RDB。

在此次的案例中，我们通过在Slave上启用了AOF与RDB来保障了数据，并恢复了Master。

但在我们目前的线上环境中，由于数据都设置有过期时间，采用AOF的方式会不太实用，过于频繁的写操作会使AOF文件增长到异常的庞大，大大超过了我们实际的数据量，这也会导致在进行数据恢复时耗用大量的时间。
因此，可以在Slave上仅开启Snapshot来进行本地化，同时可以考虑将save中的频率调高一些或者调用一个计划任务来进行定期bgsave的快照存储，来尽可能的保障本地化数据的完整性。
在这样的架构下，如果仅仅是Master挂掉，Slave完整，数据恢复可达到100%。
如果Master与Slave同时挂掉的话，数据的恢复也可以达到一个可接受的程度。