MySQL数据库高可用HA实现

1. 数据库高可用分析
    - 高可用的衡量标准
    - 数据库实现高可用的几种方式
    - MySQL数据库实现高可用
2. MySQL主从复制的容灾处理
    - MySQL支持的复制方式分析
    - 主从场景切换方式
    - 主从结构如何实现容灾

---

1. 什么是数据库高可用

1.1. 什么是高可用集群

N+1：N就是集群，1就是高可用，高可用的核心就是冗余，集群是保证服务最低使用标准的

1.2. 高可用集群的衡量标准

一般是通过系统的可靠性和可维护性来衡量的

MTTF：平均无故障时间，这是衡量可靠性的

MTTR：衡量系统的可维护性的

HA=MTTF/(MTTF+MTTR)*100%

SLA：99.999%：表示一年故障时间/宕机时间不超过6分钟

1.3. 实现高可用的三种方式

- 主从方式（非对称）

  这种方式的组织形式通常都是通过两个节点和一个或多个服务器，其中一台作为主节点（active），另一台作为备份节点（standy），备份节点应该随时都在检测主节点的健康状况，当主节点发生故障，服务会自动切换到备份节点保障服务正常运行

- 对称方式

  两个节点，都运行着不同的服务且相互备份，相互检测对方的健康，当任意一个节点发生故障，这个节点上的服务就会自动切换到另一节点

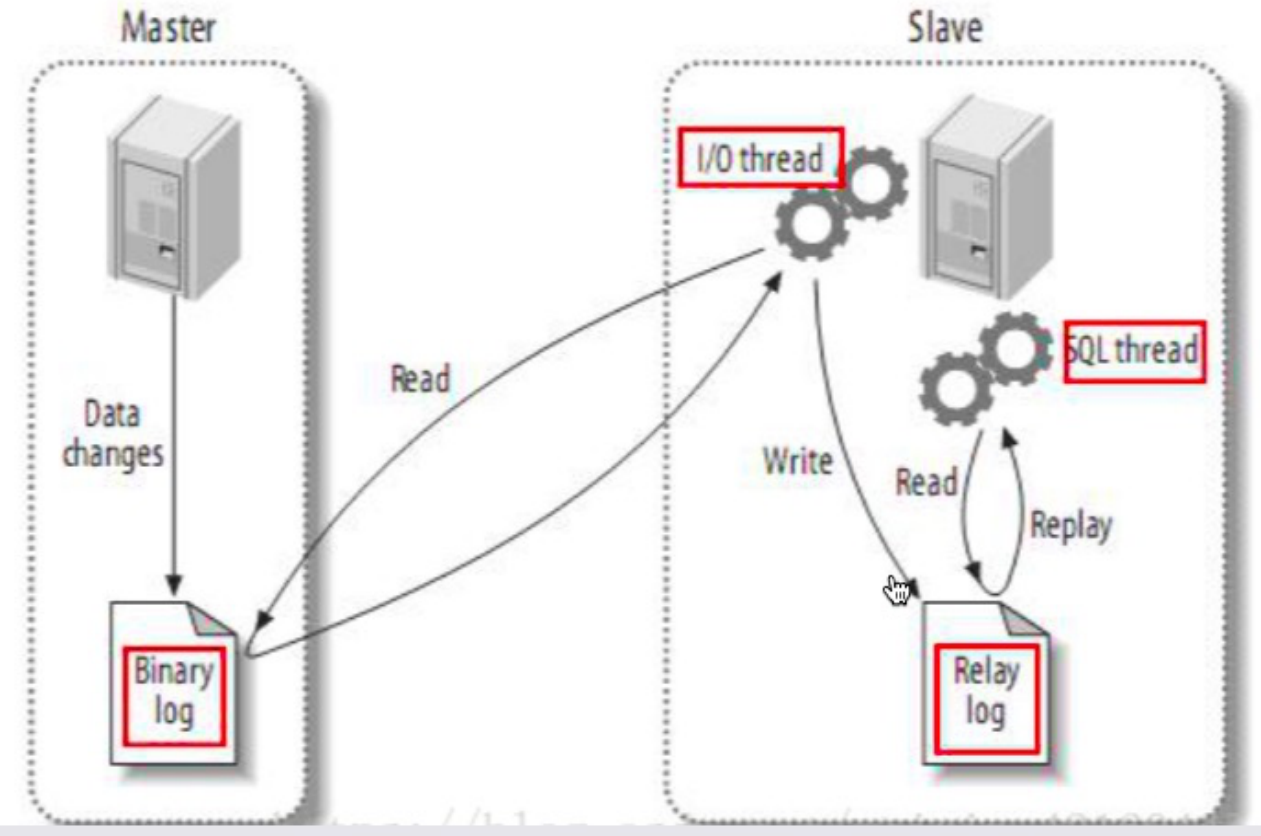
- 多机方式

  包含多个节点多个服务，每个节点都要备份运行不同的服务，出现问题自动迁移

1.4. MySQL数据的高可用实现

1.4.1. 主从方式（非对称）

- 资源：两台同版本的MySQL数据库
- 主从实现的内部运行原理和机制
  - First Step：主数据库服务器会把数据的修改记录记录进binlog日志，binlog一定要打开
  - Second Step：从库的I/O进行读取主库的binlog内容后存入自己的Relay Log中继日志中，这个I/O线程会和主库建立一个普通的客户端连接，然后主库启动一个二进制转储线程，I/O线程通过转储线程读取binlog更新事件，同步完毕后I/O进入sleep，有新的更新会再唤醒
    - Relay Log和Binlog的格式是一样的，可以用mysqlbinlog读取，也可show
    - mysql> show relaylog events in 'relay-log.000001';
    - 目前数据库有两种复制方式
      - binlog日志点position
      - GTID方式也要依赖binlog
  - 第三步：从服务器的SQL进程会从Relay Log中读取事件并在从库中重放
    - 从服务器执行重放操作时是可以在配置里声明是否写入服务器的binlog日志中



1.4.2. 配置主从服务步骤

1.4.2.1. Binlog的日志点方式配置主从同步

- 配置主从服务器参数
- 在Master服务器上创建用于复制并授权的数据库账号
- 备份Master数据库并初始化Slave服务器数据
- 启动复制链路

Master服务器配置

```
chown -R mysql:mysql /usr/local/binlog/

#配置文件
server_id=163
log_bin=/usr/local/binlog/mysql-bin
```

Slave服务器配置

```
server_id=196
log_bin=/usr/local/binlog/mysql-bin
relay_log=/usr/local/relaylog/relay-bin
#当slave宕机后，如果relay log损坏了，导致一部分中继日志没有处理，则放弃所有未完成的，重新获取执行，保证完整性
relay_log_recovery=1
#让从库数据只读，super用户，super_read_only=on
read_only=on
#从库的复制链路服务不会随数据库重启而重启，需要手动启动
skip_slave_start=on
#确保数据一致性，通过innoDB的崩溃恢复机制来保护哦
master_info_repository=TABLE
relay_log_info_repository=TABLE
#select * from mysql.slave_master_info;
#select * from mysql.slave_relay_log_info;
```

主库授权

```
mysql> use msyql;
mysql> grant replication slave on *.* to 'syncuser'@'192.168.0.103' identified by '123456';
mysql> flush privileges;

set global validate_password_policy=LOW;
set global validate_password_length=6;
```

初始化数据

```
mysqldump -uroot -p123456 --master-data=2 --single-transaction --routines --triggers --events --databases mydb > mydb.sql
```

创建复制链路

```
mysql>
CHANGE MASTER TO
MASTER_HOST='192.168.0.102',
MASTER_PORT=3306,
MASTER_USER='syncuser',
MASTER_PASSWORD='123456',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=8122;

mysql> start slave;
mysql> show slave status \G;
```

从库的binlog是否写入?

- 默认情况下是不写入的：因为写入binlog会消耗I/O，所以性能会下降，如果需要在从库上恢复数据就到Relay Log里进行导出处理
- 直接在从库上操作更行语句则会写入binlog
- 如果就是需要写入？在从库的my.cnf : log_slave_updates=on #开启同步并写入binlog
- 开启同步并写入binlog应用于从到从的情况

汤汤提问：只同步其中三个表

```
#Master配置文件
#不同步哪些数据库
binlog-ignore-db=mysql
binlog-ignore-db=test
binlog-ignore-db=information_schema
#同步哪些库
binlog-do-db=game
binlog-do-db=mydb

#Slave配置文件
#复制哪些数据库
replicate-do-db=mydb
replicate-do-db=game
#不复制哪些数据库
replicate-ignore-db=mysql
replicate-ignore-db=test

--replicate-wild-ignore-table=foo%.bar% 不复制使用表名称以开头foo且表名称以开头的表的更新bar
作业：
? replicate-ignore-table=mydb.ad_user?
```

mysqldump只导出3张表，复制的配置只复制三张表

1.4.2.1. GTID的方式来进行主从复制

不同点

- 主从服务器的参数有不同的地方

```
#在上面的基础上，需要给主从服务器都加上
gtid_mode=on
enforce_gtid_consistency=on #开启强制GTID的一致性确保事务
```

- GTID下复制链路的启动

```
mysql>
CHANGE MASTER TO
MASTER_HOST='192.168.0.102',
MASTER_PORT=3306,
MASTER_USER='syncuser',
MASTER_PASSWORD='123456',
MASTER_AUTO_POSITION=1;
```

- 启动GTID后以下数据库操作不可用
  - create table tableName.... select
  - 在一个事务中创建临时表
  - 在一个transaction中更新innoDB表和myisam表

2. 数据主从复制方式的容灾处理

2.1. MySQL支持的复制格式

2.1.1. 基于语句的复制（statement）

- 优点：记录少，只记录执行语句，易懂
- 缺点：insert into table1(create_time) values(now())，这个now就不是当时的时间了

2.1.2. 基于行复制（row）

- 优点：几乎没有基于行复制无法处理的场景
- 缺点：数据量太大了

2.1.3. 混合类型的复制（MIXED）

mixed格式默认采用statement，比如用到UUID()，ROW_COUNT()

2.2. 主从切换

作业：自己把主从更换一下

- 从库的binlog目前是没有写入的
- 需要给old主库授权new主库的权限
- 在old主库上创建复制链路
- 从库还需要把read_only关闭

2.3. MySQL主从复制模式

- 异步复制：MySQL默认就是异步复制，性能最好，但主从复制的数据不一致性概率最大
- 同步复制：当客户端发过来一个请求后，只有当所有的从库都写到Relay Log中，才回复给前端事务完成，性能最差，但一致性很强
- 半同步复制：至少一个从库完成Relay Log写入后就返回事务完成给前端

```
主从上都要安装
mysql> install plugin rpl_semi_sync_master soname='semisync_master.so'

rpl_semi_sync_master_enabled
rpl_semi_sync_master_timeout #单位是毫秒，如果主库等待从库回复超过这个时间就自动切换为异步
```

问题?

- 做过主从复制，主从一般都是实时的同步的?
- update tableName set score=99;
- 从库是不是也会被直接更新掉?
- 一般情况下，我的从库对数据的实时性要求都不是非常高
- 如果我们有一个从库更新可以延时10分钟
- 如果运气好，在你拿到10分钟前的数据和你更新之间这个表没有操作，是不是完美解决?
- 设置一个从库，将延迟时间设置成我们能处理和反应的周期长度即可

```
mysql> stop slave;
mysql> change master to master_delay=600; #单位是秒，SQL_Delay: 600
mysql> start slave;
```