基于 Docker 安装 Mysql 主从复制

一、准备工作:

1.1 服务器:

腾讯云服务器 CentOS 7.6

1.2 创建 mysql 数据映射目录

master:在/qiaoyn/mysql-master 目录下创建 data,conf,log 目录

slave: 在/qiaoyn/mysql-slave 目录下创建 data,conf,log 目录

1.3 分别配置主库和从库的 my.cnf 文件

1.4 在腾讯云防火墙安全组开放对应端口号

主库:

进入到/qiaoyn/mysql-master/conf 目录下编辑 vim my.cnf

```
[mysqld]
## 设置 server_id，同一局域网中需要唯一
server_id=101
## 指定不需要同步的数据库名称
binlog-ignore-db=mysql
## 开启二进制日志功能
log-bin=mysql-master-bin
## 设置二进制日志使用内存大小（事务）
binlog_cache_size=1M
## 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed
## 二进制日志过期清理时间。默认值为 0，表示不自动清理。
```

```
expire_logs_days=7
## 跳过主从复制中遇到的所有错误或指定类型的错误，避免 slave 端复制中断。
## 如：1062 错误是指一些主键重复，1032 错误是因为主从数据库数据不一致
slave_skip_errors=1062
#设置字符编码
collation_server=utf8mb4_unicode_ci
character_set_server=utf8mb4
```

从库:

进入到/qiaoyn/mysql-slave/conf 目录下编辑 vim my.cnf

```
[mysqld]
## 设置 server_id，同一局域网中需要唯一
server_id=102
## 指定不需要同步的数据库名称
binlog-ignore-db=mysql
## 开启二进制日志功能，以备 Slave 作为其它数据库实例的 Master 时使用
log-bin=mysql-slave1-bin
## 设置二进制日志使用内存大小（事务）
binlog_cache_size=1M
## 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed
## 二进制日志过期清理时间。默认值为 0，表示不自动清理。
expire_logs_days=7
## 跳过主从复制中遇到的所有错误或指定类型的错误，避免 slave 端复制中断。
## 如：1062 错误是指一些主键重复，1032 错误是因为主从数据库数据不一致
slave_skip_errors=1062
## relay_log 配置中继日志
relay_log=mysql-relay-bin
## log_slave_updates 表示 slave 将复制事件写进自己的二进制日志
log_slave_updates=1
```

```
## slave 设置为只读（具有 super 权限的用户除外）
read_only=1
#设置字符编码
collation_server=utf8mb4_unicode_ci
character_set_server=utf8mb4
```

二、Docker 安装 Mysql5.7（一主一从）

2.1 拉取 mysql5.7 镜像

```
#拉取镜像
docker pull mysql:5.7.13
#查看镜像
docker images
```

2.2 启动 master 主数据库容器

```
docker run -d -p 3339:3306 --name mysql-master -v /qiaoyn/mysql-master/log:/var/log/mysql -v /qiaoyn/mysql-master/data:/var/lib/mysql -v /qiaoyn/mysql-master/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=root  --privileged=true mysql:5.7.13
```

1)编辑 master 的 my.cnf 配置文件

2)docker 重启 mysql 服务

```
先重启 docker
sudo systemctl start docker
查看 docker 中运行的容器
docker ps -a
重新启动 mysql
docker restart  镜像 id
再次查看是否启动成功：
docker ps
```

3)进入 mysql-master 容器

```
#进入容器命令

docker exec -it mysql-master /bin/bash

#登录 mysql

mysql -uroot -p

（输入 root 用户密码：root）

#创建数据同步用户

CREATE USER 'slave'@'%' IDENTIFIED BY '密码';

#赋予 slave 用户权限

GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';

#查看主数据库中主从同步状态

show master status\G;
```

```
mysql> show master status;
+------------------------+----------+--------------+------------------+-------------------+
| File                   | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------------+----------+--------------+------------------+-------------------+
| mysql-master-bin.000003 |      617 |              | mysql            |                   |
+------------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

2.3 启动 slave 从数据库容器

```
docker run -d -p 3340:3306 --name mysql-slave -v /qiaoyn/mysql-slave/log:/var/log/mysql -v /qiaoyn/mysql-slave/data:/var/lib/mysql -v /qiaoyn/mysql-slave/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=root --privileged=true mysql:5.7.13
```

1)编辑 master 的 my.cnf 配置文件

2)docker 重启 mysql 服务

3)进入 mysql-slave 容器

```
#进入容器命令

docker exec -it mysql-slave /bin/bash

#登录 mysql

mysql -uroot -p

（输入 root 用户密码：root）

#在从数据库中配置主从复制

#配置参数说明：
```

#master_host：主数据库的 IP 地址；

#master_port：主数据库的运行端口；

#master_user：在主数据库创建的用于同步数据的用户账号；

#master_password：在主数据库创建的用于同步数据的用户密码；

#master_log_file：指定从数据库要复制数据的日志文件，通过查看主数据的状态，获取 File 参数；

#master_log_pos：指定从数据库从哪个位置开始复制数据，通过查看主数据的状态，获取 Position 参数；

#master_connect_retry：连接失败重试的时间间隔，单位为秒。

change master to master_host='主机 IP', master_user='slave', master_password='root', master_port=3339, master_log_file='mysql-master-bin.000003', master_log_pos=617, master_connect_retry=30;

#查看从数据库中主从同步状态

show slave status \G;

```
mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State:
                  Master_Host:
                  Master_User: slave
                  Master_Port: 3339
                Connect_Retry: 30
              Master_Log_File: mysql-master-bin.000001
          Read_Master_Log_Pos: 154
               Relay_Log_File: mysql-relay-bin.000001
                Relay_Log_Pos: 4
        Relay_Master_Log_File: mysql-master-bin.000001
             Slave_IO_Running: No
            Slave_SQL_Running: No
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
```

2.4 slave 从数据库开启主从复制

#开启主从复制

start slave;

#查看主从同步状态

show slave status \G;

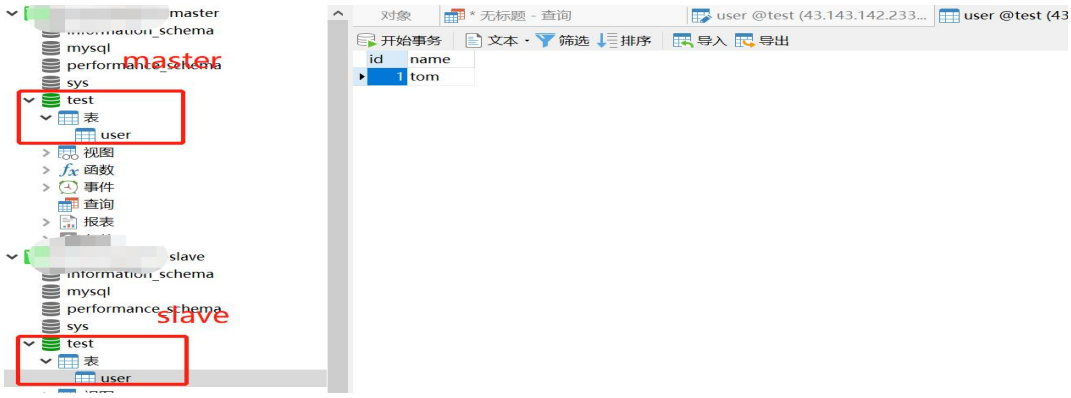
```
mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: [redacted]r to send event
                  Master_Host:
                  Master_User: slave
                  Master_Port: 3339
                Connect_Retry: 30
              Master_Log_File: mysql-master-bin.000001
          Read_Master_Log_Pos: 154
               Relay_Log_File: mysql-relay-bin.000002
                Relay_Log_Pos: 327
        Relay_Master_Log_File: mysql-master-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 154
              Relay_Log_Space: 534
              Until_Condition: None
```

此时主从复制已经配置完成;

三、 验证

用 Navicat 连接数据库,创建 test 测试库,创建 user 表,往主库插入数据,同时观察从库中数据变化

(从库与主库保持一致则证明主从复制生效)



四、 mysql 主从复制原理

4.1 为什么需要主从复制

1、在业务复杂的系统中，有这么一个情景，有一句 sql 语句需要锁表，导致暂时不能使用读的服务，那么就很影响运行中的业务，使用主从复制，让主库负责写，从库负责读，这样，即使主库出现了锁表的情景，通过读从库也可以保证业务的正常运作。

2、做数据的热备

3、架构的扩展。业务量越来越大，I/O 访问频率过高，单机无法满足，此时做多库的存储，降低磁盘 I/O 访问的频率，提高单个机器的 I/O 性能。

4.2、什么是 mysql 的主从复制

MySQL 主从复制是指数据可以从一个 MySQL 数据库服务器主节点复制到一个或多个从节点。MySQL 默认采用异步复制方式，这样从节点不用一直访问主服务器来更新自己的数据，数据的更新可以在远程连接上进行，从节点可以复制主数据库中的所有数据库或者特定的数据库，或者特定的表。

4.3、mysql 复制原理

原理：

（1）master 服务器将数据的改变记录二进制 binlog 日志，当 master 上的数据发生改变时，则将其改变写入二进制日志中；

（2）slave 服务器会在一定时间间隔内对 master 二进制日志进行探测其是否发生改变，如果发生改变，则开始一个 I/OThread 请求 master 二进制事件

（3）同时主节点为每个 I/O 线程启动一个 dump 线程，用于向其发送二进制事件，并保存至从节点本地的中继日志中，从节点将启动 SQL 线程从中继日志中读取二进制日志，在本地重放，使得其数据和主节点的保持一致，最后 I/OThread 和 SQLThread 将进入睡眠状态，等待下一次被唤醒。

也就是说：

1)从库会生成两个线程,一个 I/O 线程,一个 SQL 线程;

2)I/O 线程会去请求主库的 binlog,并将得到的 binlog 写到本地的 relay-log(中继日志)文件中;

3)主库会生成一个 log dump 线程,用来给从库 I/O 线程传 binlog;

4)SQL 线程,会读取 relay log 文件中的日志,并解析成 sql 语句逐一执行;

注意：

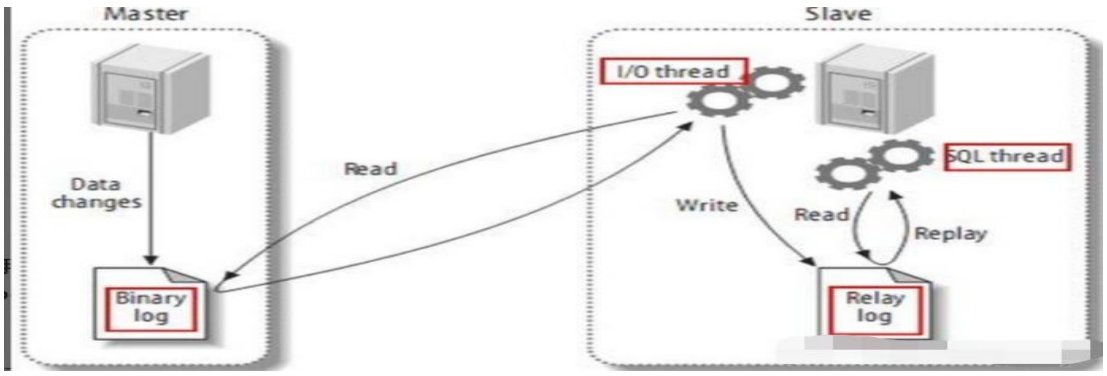
1)master 将操作语句记录到 binlog 日志中，然后授予 slave 远程连接的权限（master 一定要开启 binlog 二进制日志功能；通常为了数据安全考虑，slave 也开启 binlog 功能）。

2)slave 开启两个线程：IO 线程和 SQL 线程。其中：IO 线程负责读取 master 的 binlog 内容到中继日志 relay log 里；SQL 线程负责从 relay log 日志里读出 binlog 内容，并更新到 slave 的数据库里，这样就能保证 slave 数据和 master 数据保持一致了。

3)Mysql 复制至少需要两个 Mysql 的服务，当然 Mysql 服务可以分布在不同的服务器上，也可以在一台服务器上启动多个服务。

4)Mysql 复制最好确保 master 和 slave 服务器上的 Mysql 版本相同(如果不能满足版本一致，那么要保证 master 主节点的版本低于 slave 从节点的版本)

5)master 和 slave 两节点间时间需同步.



具体步骤：

1、从库通过手工执行 change master to 语句连接主库，提供了连接的用户一切条件（user 、password、port、ip），并且让从库知道，二进制日志的起点位置（file 名 position 号）； start slave

2、从库的 IO 线程和主库的 dump 线程建立连接。

3、从库根据 change master to 语句提供的 file 名和 position 号，IO 线程向主库发起 binlog 的请求。

4、主库 dump 线程根据从库的请求，将本地 binlog 以 events 的方式发给从库 IO 线程。

5、从库 IO 线程接收 binlog events，并存放到本地 relay-log 中，传送过来的信息，会记录到 master.info 中

6、从库 SQL 线程应用 relay-log，并且把应用过的记录到 relay-log.info 中，默认情况下，已经应用过的 relay 会自动被清理 purge

4.4、mysql 主从形式

一主一从,一主多从,主主复制,多主一层,联级复制

4.5、mysql 主从同步延时分析

mysql 的主从复制都是单线程的操作，主库对所有 DDL 和 DML 产生的日志写进 binlog，由于 binlog 是顺序写，所以效率很高，slave 的 sql thread 线程将主库的 DDL 和 DML 操作事件在 slave 中重放。DML 和 DDL 的 IO 操作是随机的，不是顺序，所以成本要高很多，另一方面，由于 sql thread 也是单线程的，当主库的并发较高时，产生的 DML 数量超过 slave 的 SQL thread 所能处理的速度，或者当 slave 中有大型 query 语句产生了锁等待，那么延时就产生了。

解决方案：

1.业务的持久化层的实现采用分库架构，mysql 服务可平行扩展，分散压力。

2.单个库读写分离，一主多从，主写从读，分散压力。这样从库压力比主库高，保护主库。

3.服务的基础架构在业务和 mysql 之间加入 memcache 或者 redis 的 cache 层。降低 mysql 的读压力。

4.不同业务的 mysql 物理上放在不同机器，分散压力。

5.使用比主库更好的硬件设备作为 slave，mysql 压力小，延迟自然会变小。