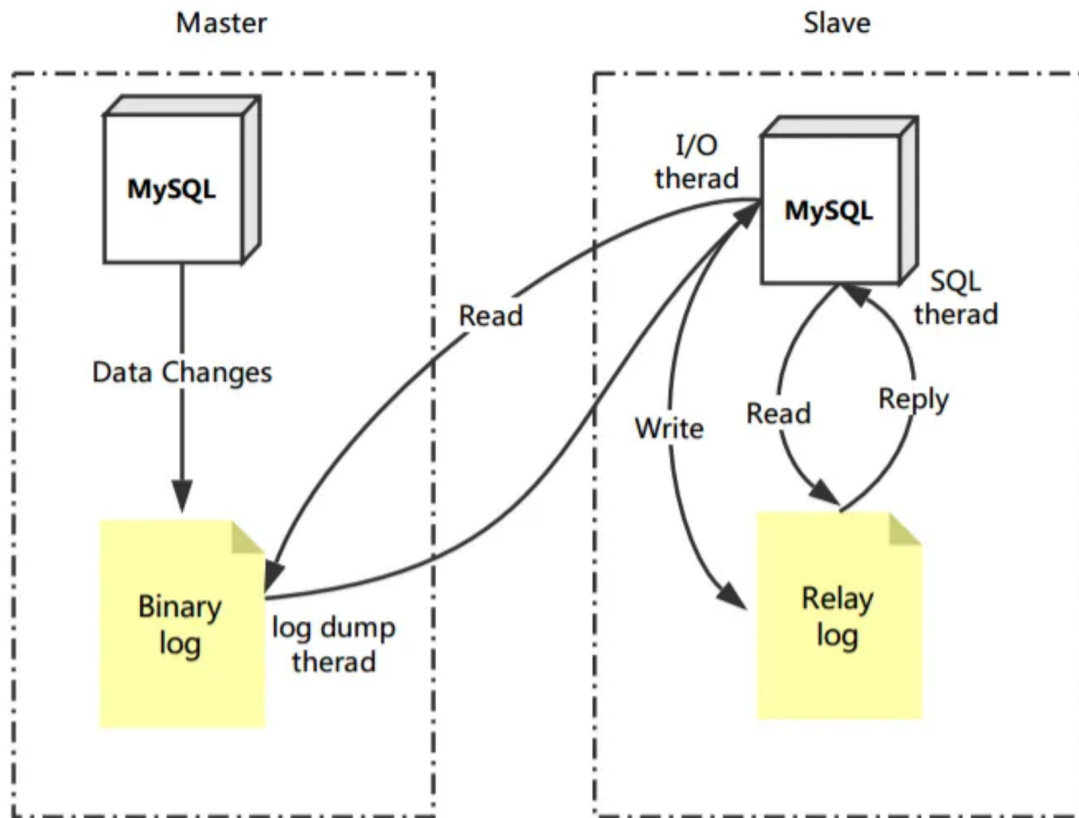


# MySQL集群篇

## 一、集群搭建之主从复制

### 1.1 主从复制原理



- (1) Master 将数据改变记录到二进制日志(binary log)中, 也就是配置文件 log-bin 指定的文件, 这些记录叫做二进制日志事件(binary log events);
- (2) Slave 通过 I/O 线程读取 Master 中的 binary log events 并写入到它的中继日志(relay log);
- (3) Slave 重做中继日志中的事件, 把中继日志中的事件信息一条一条的在本地执行一次, 完成数据在本地的存储, 从而实现将改变反映到它自己的数据(数据重放)。

#### 注意事项:

- 主从服务器操作系统版本和位数一致;
- Master 和 Slave 数据库的版本要一致;
- Master 和 Slave 数据库中的数据要一致;
- Master 开启二进制日志, Master 和 Slave 的 server\_id 在局域网内必须唯一;

### 1.2 binlog和relay日志

- **bin log:** 将数据改变记录到二进制日志(binary log)中, 可用于本机数据恢复和主从同步。
- **relay (中继) log:** 重做中继日志。Slave节点会把中继日志中的事件信息一条一条的在本地执行一次, 实现主从同步, 这个过程也叫数据重放。

## 1.2.1 binlog的三种模式

### 1. ROW模式

日志中会记录成每一行数据被修改的形式，然后在slave端再对相同的数据进行修改

- **优点：**bin-log中可以**不记录执行的sql语句的上下文相关的信息**，仅仅只需要记录那一条记录被修改了，修改成什么样了。所以ROW模式的日志的内容会非常清楚的记录下每一行数据修改的细节。而且不会出现某些特定情况下的**存储过程或函数**，以及**触发器**的调用和触发无法被正确复制的问题。
- **缺点：**ROW模式下，所有的执行的语句当记录到日志中的时候，都将以每行记录的修改记录，这样会产生大量的日志内容。
  - 比如：有这样一条update语句：update product set owner\_member\_id='d' where owner\_member\_id='a'，执行之后，日志中记录的**不是**这条update语句所对应的事件(mysql是以事件的形式来记录bin-log日志)，而是这条语句所更新的每一条记录的变化情况，这样就记录成很多条记录被更新的很多事件，自然bin-log日志的量会很大。

### 2. Statement模式

**Statement模式：**每一条会修改数据的sql都会记录到master的bin-log中。slave在复制的时候sql进程会解析成和原来master端执行过的相同的sql来再次执行。

- **优点：**Statement模式下的优点，首先就是**解决了ROW模式下的缺点**，不需要记录每一行数据的变化，减少bin-log日志量，节约io，提高性能。因为他**只需要记录在master上所执行的语句的细节，以及执行语句时候的上下文的信息**。
- **缺点：**由于它是记录的执行语句，所以为了让这些语句在slave端也能正确执行，那么他还必须记录每条语句在执行的时候的一些相关信息，也就是上下文信息，以保证所有语句在slave端被执行的时候能够得到和在master端执行时候相同的结果。
  - 另外，由于mysql现在发展比较快，很多新功能加入，使mysql的复制遇到了不小的挑战，自然复制的时候涉及到越复杂的内容，bug也就越容易出现。在Statement模式下，目前已经发现的就有不少情况会造成mysql的复制问题，主要是修改数据的时候使用了某些特定的函数或者功能的时候会出现。比如：sleep()在有些版本就不能正确复制。

### 3. Mixed模式

- 实际上就是**前两种模式的结合**，在Mixed模式下，MySQL会根据执行的每一条具体的sql语句来区分对待记录的日志形式，也就是在**Statement和Row**之间选一种。
- 新版本中的Statement模式还是和以前一样，仅仅记录执行的语句。而新版本的MySQL中对ROW模式被做了优化，并不是所有的修改都会以ROW模式来记录，像遇到表结构变更的时候就会以Statement模式来记录，如果sql语句确实就是Update或者Delete等修改数据的语句，那么还是会记录所有行的变更。

## 1.2.2 开启binlog

修改my.cnf文件

在[mysqld]段下添加：

```
1  #binlog刷盘策略
2  sync_binlog=1
3  #需要备份的数据库
4  binlog-do-db=hello
5  #不需要备份的数据库
6  binlog-ignore-db=mysql
7  #启动二进制文件
8  log-bin=mysql-bin
9  #服务器ID
10 server-id=132
```

**sync\_binlog参数：**

- **0**：存储引擎不进行binlog的刷新到磁盘，而由操作系统的文件系统控制缓存刷新。
- **1**：每提交一次事务，存储引擎调用文件系统的sync操作进行一次缓存的刷新，这种方式最安全，但性能较低。
- **n**：当提交的日志组=n时，存储引擎调用文件系统的sync操作进行一次缓存的刷新。

sync\_binlog=0或sync\_binlog大于1，事务被提交，而尚未同步到磁盘。因此，在电源故障或操作系统崩溃时有可能服务器已承诺尚未同步一些事务到二进制日志。因此它是不可能执行例行程序恢复这些事务，他们将会丢失二进制日志。

## 1.2.3 调整binlog日志模式

查看binlog的日志模式：

```
1  mysql> show variables like 'binlog_format';
2  +-----+-----+
3  | variable_name | value |
4  +-----+-----+
5  | binlog_format | ROW   |
6  +-----+-----+
7  1 row in set (0.00 sec)
```

调整binlog的日志模式：

binlog的三种格式：`STATEMENT`、`ROW`、`MIXED`。

```

1  mysql> set binlog_format=STATEMENT;
2  Query OK, 0 rows affected (0.00 sec)
3
4  mysql> show variables like 'binlog_format';
5  +-----+-----+
6  | variable_name | value      |
7  +-----+-----+
8  | binlog_format | STATEMENT  |
9  +-----+-----+
10 1 row in set (0.00 sec)

```

### 1.2.4 查看bin log和relay log日志

因为binlog日志文件：mysql-bin.000005是二进制文件，没法用vi等打开，这时就需要mysql的自带的mysqlbinlog工具进行解码，执行：`mysqlbinlog mysql-bin.000005` 可以将二进制文件转为可读的sql语句。

```

1  mysqlbinlog --base64-output=decode-rows -v -v mysql-bin.000058 > binlog

```

### 1.2.5 使用命令查看binlog

`show master logs`，查看所有二进制日志列表，和 `show binary logs` 同义。

```

1  mysql> show master logs;
2  +-----+-----+
3  | Log_name      | File_size |
4  +-----+-----+
5  | mysql-bin.000001 | 385      |
6  +-----+-----+
7  1 row in set (0.00 sec)

```

使用 `show binlog events` 命令可以以列表的形式显示日志中的事件信息。

`show binlog events`命令的格式：

```

1  show binlog events [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count];

```

说明：

- (1) IN 'log\_name': 指定要查询的binlog文件名（如果省略此参数，则默认指定第一个binlog文件）；
- (2) FROM pos: 指定从哪个pos起始点开始查起（如果省略此参数，则从整个文件的第一个pos点开始算）；
- (3) LIMIT [offset]：偏移量（默认为0）；
- (4) row\_count: 查询总条数（如果省略，则显示所有行）。

```

1 mysql> show binlog events in 'mysql-bin.000001';
2 +-----+-----+-----+-----+-----+-----+
3 | Log_name      | Pos | Event_type      | Server_id | End_log_pos | Info
4 +-----+-----+-----+-----+-----+-----+
5 | mybin.000001 | 4   | Format_desc     | 132       | 123         | Server ver:
6 | mybin.000001 | 123 | Previous_gtid   | 132       | 154         |
7 +-----+-----+-----+-----+-----+-----+
8 2 rows in set (0.00 sec)

```

切换binlog文件:

```

1 mysql> flush logs;
2 query OK, 0 rows affected (0.00 sec)

```

会生成一个新的日志文件。

## 1.3 基于binlog主从复制

### 1.3.1 关闭主从机器的防火墙

```

1 systemctl stop iptables (需要安装iptables服务)
2 systemctl stop firewalld (默认)
3 systemctl disable firewalld.service (设置开启不启动)

```

### 1.3.2 主服务器配置

查看binlog是否开启可以使用命令:

```

1 mysql> show variables like 'log_bin%';
2 +-----+-----+
3 | variable_name      | value |
4 +-----+-----+
5 | log_bin            | OFF   |
6 | log_bin_basename   |       |
7 | log_bin_index       |       |
8 | log_bin_trust_function_creators | OFF   |
9 | log_bin_use_v1_row_events | OFF   |
10 +-----+-----+
11 5 rows in set (0.12 sec)

```

log\_bin如果是OFF代表是未开启状态。

## 第一步：修改my.cnf文件

在[mysqld]段下添加：

```
1  #binlog刷盘策略
2  sync_binlog=1
3  #需要备份的数据库
4  binlog-do-db=hello
5  #不需要备份的数据库
6  binlog-ignore-db=mysql
7  #启动二进制文件
8  log-bin=mysql-bin
9  #服务器ID
10 server-id=132
```

## 第二步：重启mysql服务

```
1  systemctl restart mysqld
```

## 第三步：主机给从机授备份权限

注意：先要登录到MySQL命令客户端

```
1  mysql> GRANT REPLICATION SLAVE ON *.* TO '从机MySQL用户名'@'从机IP' identified
    by '从机MySQL密码';
```

示例：

```
1  GRANT REPLICATION SLAVE ON *.* TO 'root'@'%' identified by 'root';
```

注意事项：

```
1  一般不用root帐号，“%”表示所有客户端都可能连，只要帐号，密码正确，此处可用具体客户端IP代替，
    如192.168.145.226，加强安全。
```

mysql5.7对密码的强度是有要求的，必须是字母+数字+符号组成的，可以使用如下方法调整密码强度

设置密码长度最低位数

```
mysql> set global validate_password_length=4;
```

设置密码强度级别

```
mysql> set global validate_password_policy=0;
```

validate\_password\_policy有以下取值：

Policy	Tests Performe
0 or LOW	Length
1 or MEDIUM	numeric, lowercase/uppercase, and special characters
2 or STRONG	Length; numeric, lowercase/uppercase, and special characters

默认是1，即MEDIUM，所以刚开始设置的密码必须符合长度，且必须含有数字，小写或大写字母，特殊字符。

#### 第四步：刷新权限

```
1 | mysql> FLUSH PRIVILEGES;
```

#### 第五步：查询master的状态

```
1 | mysql> show master status;
2 |
3 | +-----+-----+-----+-----+-----+
4 | | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
   | Executed_Gtid_Set |
5 | +-----+-----+-----+-----+-----+
6 | | mysql-bin.000002 |      593 | hello        | mysql              |
   | |
7 | +-----+-----+-----+-----+-----+
8 | 1 row in set
```

### 1.3.3 从服务器配置

#### 第一步：修改my.conf文件

```
1 | [mysqld]
2 | server-id=133
```

#### 第二步：删除UUID文件

如果出现此错误：

```
1 | Fatal error: The slave I/O thread stops because master and slave have equal
   | MySQL server UUIDs; these UUIDs must be different for replication to work.
```

原因：

```
1 | 因为mysql是克隆的系统所以mysql的uuid是一样的，所以需要修改。
```

解决方法：

```
1 | 删除/var/lib/mysql/auto.cnf文件，重新启动MySQL服务。
```

### 第三步：重启并登录到MySQL进行配置从服务器

```
1 | mysql>change master to
2 | master_host='39.103.222.177',
3 | master_port=3306,
4 | master_user='root',
5 | master_password='root',
6 | master_log_file='mysql-bin.000002',
7 | master_log_pos=593,
8 | MASTER_AUTO_POSITION=0;
```

注意：

语句中间不要断开，`master_port`为mysql服务器端口号(无引号)，`master_user`为执行同步操作的数据库账户，“593”无单引号(此处的593就是`show master status`中看到的`position`的值，这里的`mysql-bin.000002`就是`file`对应的值)。

### 第四步：启动从服务器复制功能

```
1 | mysql>start slave;
```

### 第五步：检查从服务器复制功能状态

```
1 | mysql> show slave status;
2 | .....(省略部分)
3 | Slave_IO_Running: Yes //此状态必须YES
4 | Slave_SQL_Running: Yes //此状态必须YES
5 | .....(省略部分)
```

注：Slave\_IO及Slave\_SQL进程必须正常运行，即YES状态，否则都是错误的状态(如：其中一个NO均属错误)。

运行

停止

解释

新建

加载

保存

另存为

美化 SQL

备注

导出

查询创建工具 查询编辑器

```
1 show slave status;
```

信息	结果1	概况	状态			
Relay_Master_Log_File	Slave_IO_Running	Slave_SQL_Running	Replicate_Do_DB	Replicate_Ignore_DB	Replicate_Do_Table	F
mysql-bin.000002	Yes	Yes				





以上操作过程，从服务器配置完成。

### 1.3.4 测试

搭建成功之后，往主机中插入数据，看看从机中是否有数据

## 1.4 基于GTID的主从复制

### 1.4.1 什么是GTID

GTID就是全局事务标识符（Global Transaction Identifiers），基于事务的复制。

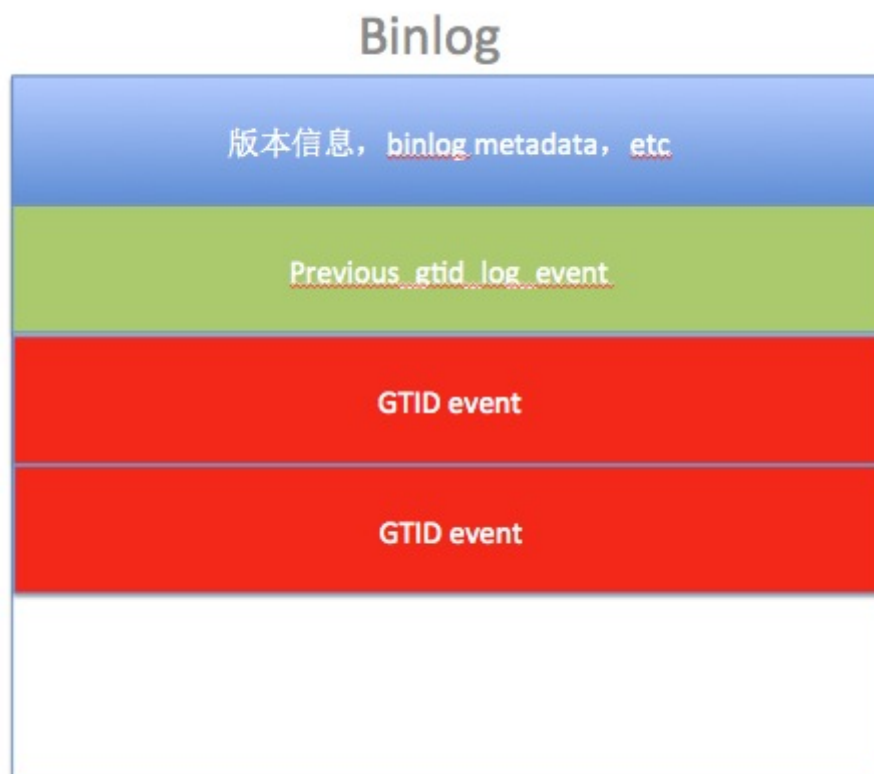
使用 GTID 时，可以识别和跟踪每个事务，因为它在原始服务器上提交并由任何从属服务器应用；这意味着在启动新从站或故障到新主服务器时，使用 GTID 引用日志文件或这些文件中的位置时，不需要引用日志文件或位置，这大大简化了这些任务。由于基于 GTID 的复制完全基于事务，因此只需确定主复制和从级复制是否一致；只要在主主机上提交的所有事务也都提交到从站上，就保证两者之间的一致性。

```
1 | GTID = server_uuid:transaction_id
```

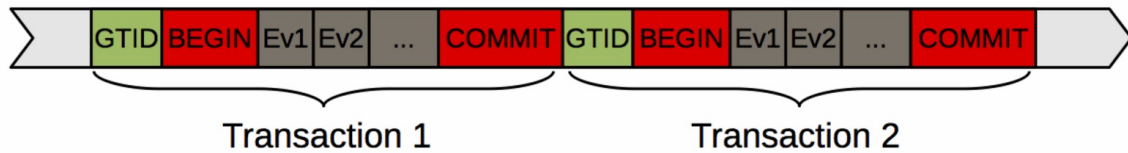
server\_uuid 来源于 auto.cnf

### 1.4.2 GTID和Binlog的关系

GTID在binlog中的结构：



GTID event 结构：



Previous\_gtid\_log\_event:

1. Previous\_gtid\_log\_event 在每个binlog 头部都会有
2. 每次binlog rotate的时候存储在binlog头部
3. Previous-GTIDs在binlog中只会存储在这台机器上执行过的所有binlog，不包括手动设置gtid\_purged值。
4. 换句话说，如果你手动set global gtid\_purged=xx；那么xx是不会记录在Previous\_gtid\_log\_event中的。

**GTID和Binlog之间的关系是怎么对应的呢：**

```

1  * 假设有4个binlog: bin.001,bin.002,bin.003,bin.004
2  * bin.001 : Previous-GTIDs=empty; binlog_event有: 1-40
3  * bin.002 : Previous-GTIDs=1-40; binlog_event有: 41-80
4  * bin.003 : Previous-GTIDs=1-80; binlog_event有: 81-120
5  * bin.004 : Previous-GTIDs=1-120; binlog_event有: 121-160
6
7  1. 假设现在我们要找GTID=$A, 那么MySQL的扫描顺序为: 从最后一个binlog开始扫描(即:
   bin.004)
8  2. bin.004的Previous-GTIDs=1-120, 如果$A=140 > Previous-GTIDs,那么肯定在bin.004
   中
9  3. bin.004的Previous-GTIDs=1-120, 如果$A=88 包含在Previous-GTIDs中,那么继续对比上一
   个binlog文件 bin.003,然后再循环前面2个步骤,直到找到为止

```

### 1.4.3 配置GTID主从复制

#### 1. 修改master、slave服务器的my.cnf文件

```

1  #开启GTID模式(必选)
2  gtid_mode=ON
3  #强制gtid一致性(必选)
4  enforce-gtid-consistency=true

```

#### 2. 重启mysql

```

1  [root@mysql132 ~]# systemctl restart mysqld

```

#### 3. 从服务器中执行change master

```

1  mysql> change master to
2  master_host='39.103.222.177',
3  master_port=3306,
4  master_user='root',
5  master_password='root',
6  master_auto_position = 1;

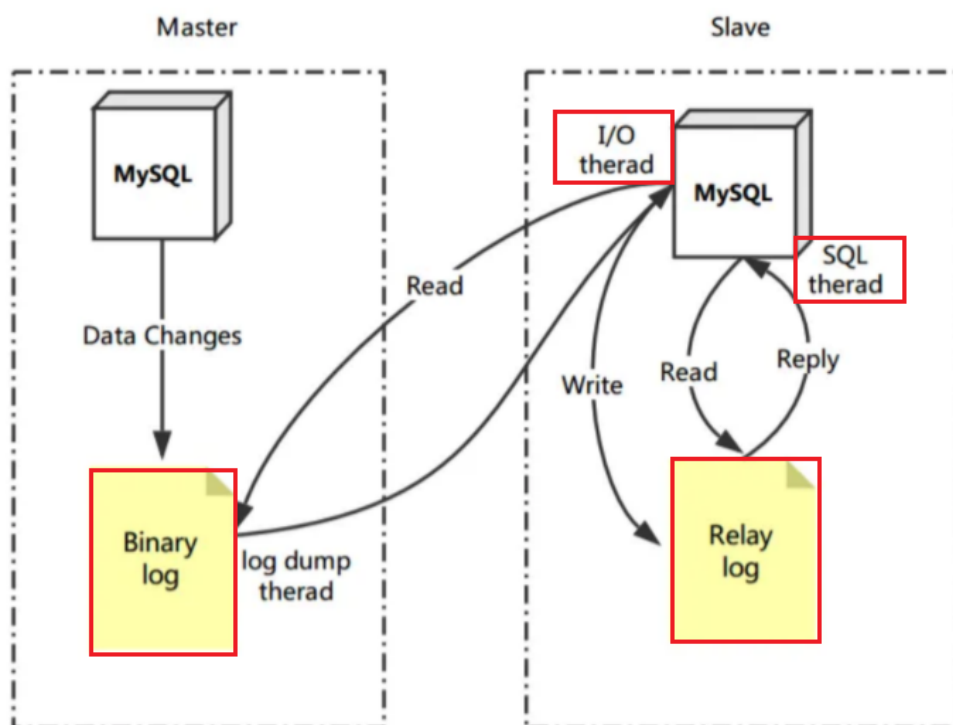
```

## 4. 开启同步

```
1 | START SLAVE;
```

### 1.5 主从同步延迟的原因及解决办法

mysql 用主从同步的方法进行读写分离，减轻主服务器的压力的做法现在在业内做的非常普遍。主从同步基本上能做到实时同步。



在配置好了，主从同步以后，主服务器会把更新语句写入binlog，从服务器的IO 线程回去读取主服务器的binlog，并且写到从服务器的Relay log 里面，然后从服务器的 SQL thread 会一个一个执行 relay log 里面的sql，进行数据恢复。

注意：5.6.3 之前的IO线程仅有一个。5.6.3之后有多线程去读，速度会大幅提升，所以主从延迟也会相对少一些

#### 1、原因

我们知道，一个服务器开放N个链接给客户端来连接的，这样会有大并发的更新操作，但是从服务器的里面读取binlog 的线程仅有一个，当某个SQL在从服务器上执行的时间稍长 或者由于某个SQL要进行锁表就会导致，主服务器的SQL大量积压，未被同步到从服务器里。这就导致了主从不一致，也就是主从延迟。

#### 2、解决办法

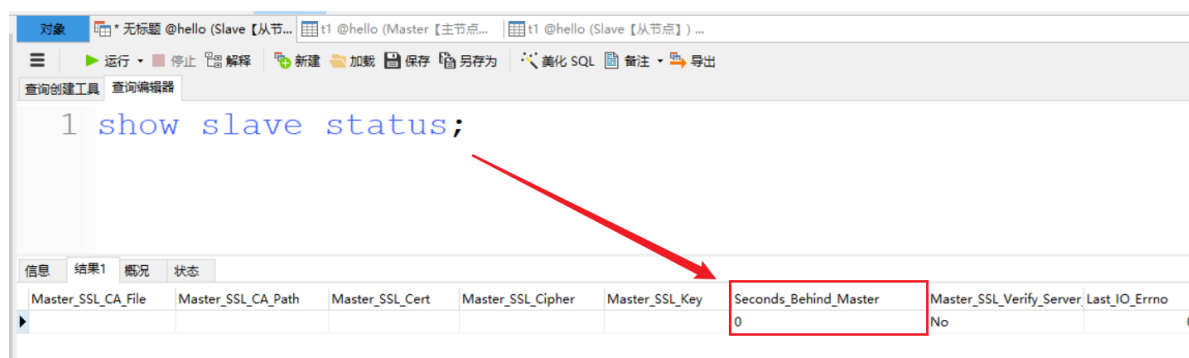
实际上主从同步延迟根本没有什么一招制敌的办法，因为所有的SQL必须都要在从服务器里面执行一遍，但是主服务器如果不断的有更新操作源源不断的写入，那么一旦有延迟产生，那么延迟加重的可能性就会原来越大。当然我们可以做一些缓解的措施。

1. 因为主服务器要负责更新操作，所以对安全性的要求比从服务器高，有些设置可以修改，比如sync\_binlog=1, innodb\_flush\_log\_at\_trx\_commit = 1 之类的设置。而slave则不需要这么高的数据安全，完全可以讲sync\_binlog设置为0或者关闭binlog, innodb\_flushlog, innodb\_flush\_log\_at\_trx\_commit 也可以设置为0来提高sql的执行效率 这个能很大程度上提高效率。
  - 另外就是使用比主库更好的硬件设备作为slave。
2. 把一台从服务器当度作为备份使用，而不提供查询，那边他的负载下来了，执行relay log 里面的SQL效率自然就高了。
3. 增加从服务器，目的还是分散读的压力，降低服务器负载。

### 3、判断延迟的方法

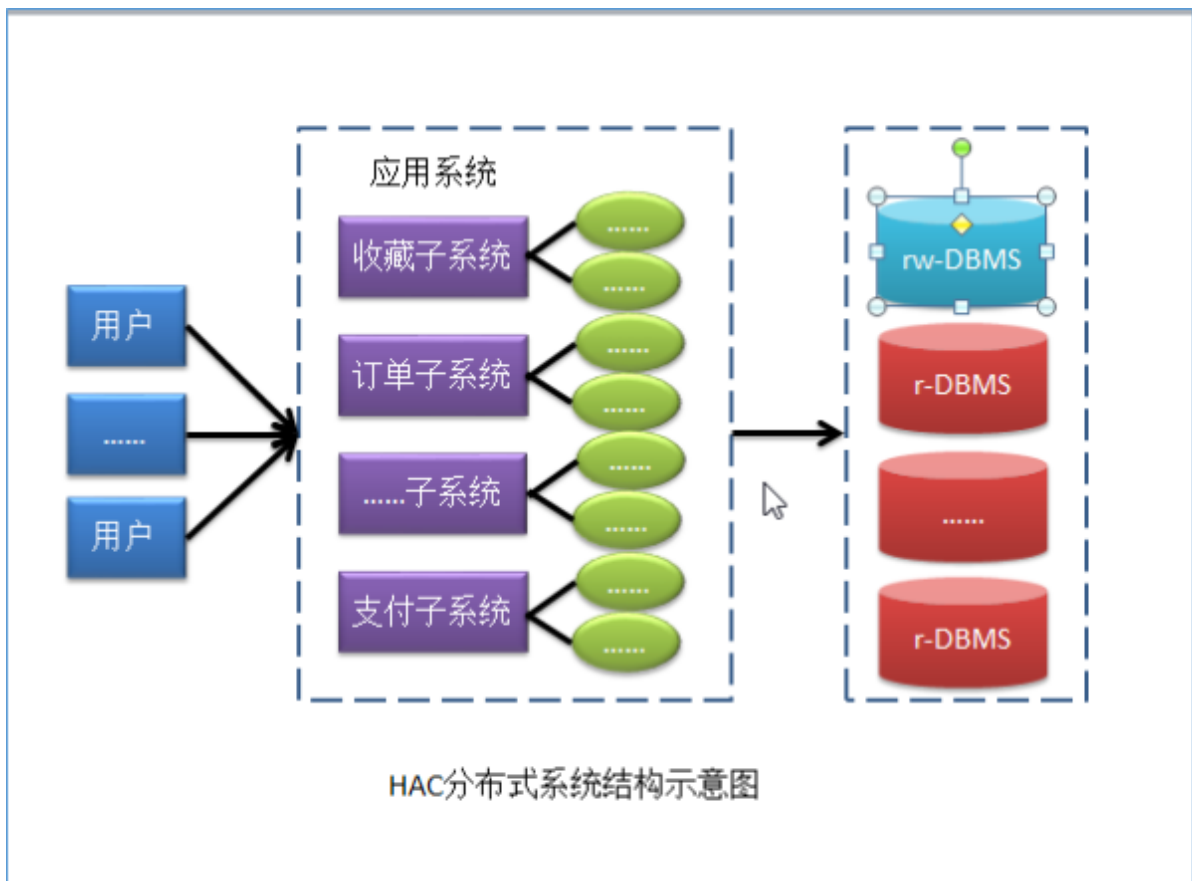
MySQL提供了从服务器状态命令，可以通过 `show slave status` 进行查看，比如可以看看 [Seconds Behind Master](#) 参数的值来判断，是否有发生主从延时。其值有这么几种：

- NULL：表示io\_thread或是sql\_thread有任何一个发生故障，也就是该线程的Running状态是No，而非Yes。
- 0：该值为零，表示主从复制状态正常



## 二、集群搭建之读写分离

### 2.1 读写分离的理解



#### 名词解释:

- 1 | HAC: High Availability Cluster, 高可用集群

#### 注意事项:

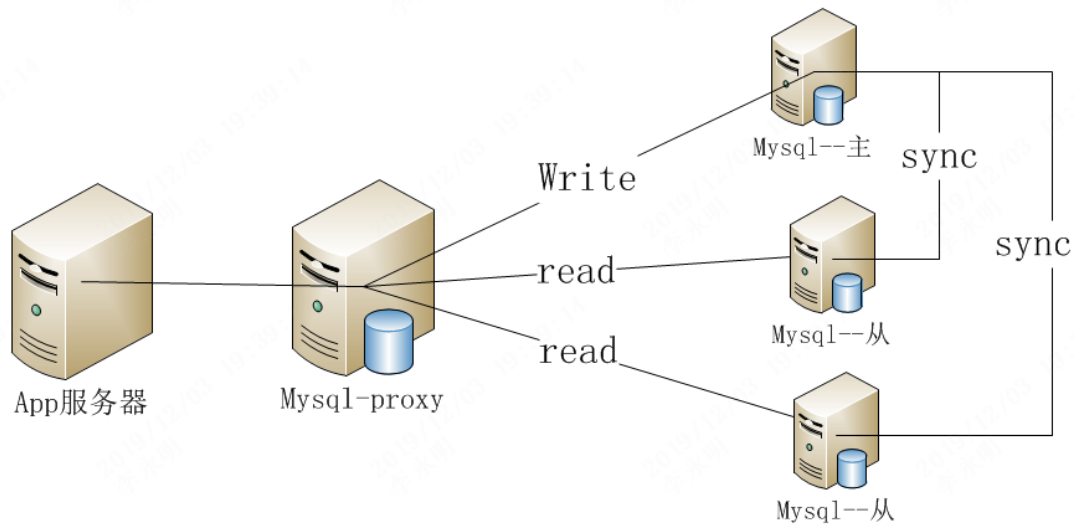
- 1 | MySQL的主从复制，只会保证主机对外提供服务，而从机是不对外提供服务的，只是在后台为主机进行备份。

## 2.2 读写分离演示需求

- 1 | MySQL master: 132
- 2 |
- 3 | MySQL slave : 133

## 2.3 MySQL-Proxy

mysql-proxy是mysql官方提供的mysql中间件服务，提供读写分离的功能。



MySQLProxy虽然可以实现读写分离的操作，但是MySQLProxy官方并没有推出稳定版，其中的坑还是挺多的，并不推荐在生产环境使用。官方推荐使用MySQLRouter，所以关于MySQLProxy的使用大家做了解内容即可。

## 2.4 Atlas

### 2.4.1 简介

Atlas是由 Qihoo 360公司Web平台部基础架构团队开发维护的一个基于MySQL协议的数据中间层项目。它在MySQL官方推出的MySQL-Proxy 0.8.2版本的基础上，修改了大量bug，添加了很多功能特性。目前该项目在360公司内部得到了广泛应用，很多MySQL业务已经接入了Atlas平台，每天承载的读写请求数达几十亿条。同时，有超过50家公司在生产环境中部署了Atlas。

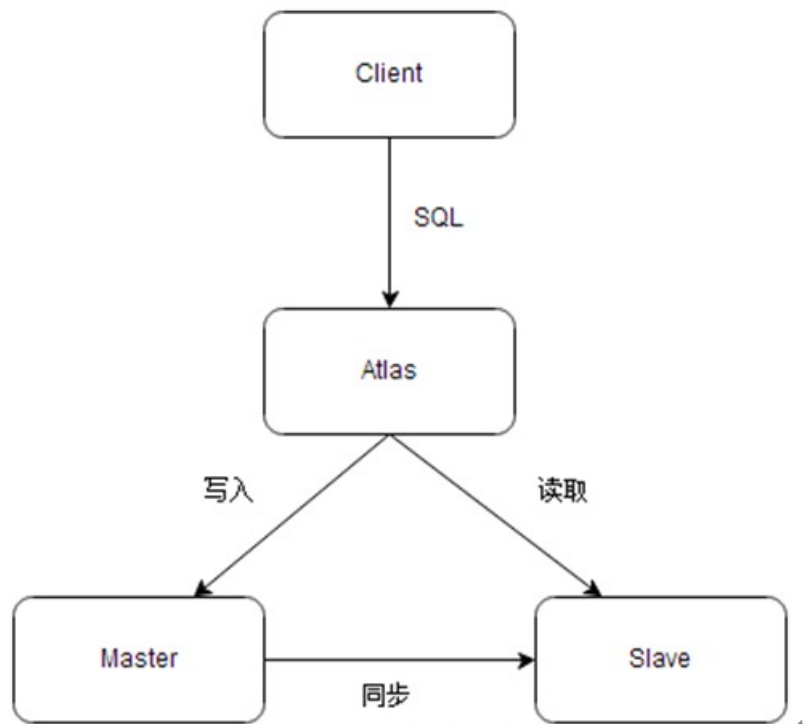


图 2 Atlas 总体架构

官方文档:

[https://github.com/Qihoo360/Atlas/blob/master/README\\_ZH.md](https://github.com/Qihoo360/Atlas/blob/master/README_ZH.md)

## 2.4.2 下载安装

```
1 | > wget https://github.com/Qihoo360/Atlas/releases/download/2.2.1/Atlas-2.2.1.el6.x86_64.rpm
```

下载好了之后, 进行安装

```
1 | > rpm -ivh Atlas-2.2.1.el6.x86_64.rpm
```

安装好了, 它会默认在"/usr/local/mysql-proxy"下给你生成4个文件夹, 以及需要配置的文件, 如下:

```
[root@k8s-master01 mysql-proxy]# pwd
/usr/local/mysql-proxy
[root@k8s-master01 mysql-proxy]# ll
总用量 16
drwxr-xr-x 2 root root 4096 3月 24 14:51 bin
drwxr-xr-x 2 root root 4096 3月 24 14:51 conf
drwxr-xr-x 3 root root 4096 3月 24 14:51 lib
drwxr-xr-x 2 root root 4096 12月 17 2014 log
```

- **bin目录:** 下放的都是可执行文件
  - "encrypt"是用来生成MySQL密码加密的, 在配置的时候会用到
  - "mysql-proxy"是MySQL自己的读写分离代理
  - "mysql-proxyd"是360弄出来的, 后面有个"d", 服务的启动、重启、停止。都是用他来执行的
- **conf目录:** 下放的是配置文件
  - "test.cnf"只有一个文件, 用来配置代理的, 可以使用vim来编辑
- **lib目录:** 下放的是一些包, 以及Atlas的依赖

- **log目录**: 下放的是日志, 如报错等错误信息的记录

### 2.4.3 配置

进入bin目录, 使用encrypt来对数据库的密码进行加密, 我的MySQL数据的用户名是root, 密码是root, 我需要密码进行加密

```
1 [root@localhost bin]# ./encrypt root
2 DAJn18cvzy8=
```

配置Atlas, 使用vim进行编辑

```
1 [root@localhost conf]# cd /usr/local/mysql-proxy/conf/
2 [root@localhost conf]# vim test.cnf
```

进入后, 可以在Atlas进行配置, 360写的中文注释都很详细, 根据注释来配置信息, 其中比较重要, 需要说明的配置如下:

这是用来登录到Atlas的管理员的账号与密码, 与之对应的是“#Atlas监听的管理接口IP和端口”, 也就是说需要设置管理员登录的端口, 才能进入管理员界面, 默认端口是2345, 也可以指定IP登录, 指定IP后, 其他的IP无法访问管理员的命令界面。方便测试, 我这里没有指定IP和端口登录。

```
1 #管理接口的用户名
2 admin-username = admin
3 #管理接口的密码
4 admin-password = admin
```

这是用来配置主数据的地址与从数据库的地址, 这里配置的主数据库是132, 从数据库是133

```
1 #Atlas后端连接的MySQL主库的IP和端口, 可设置多项, 用逗号分隔
2 proxy-backend-addresses = 39.103.222.177:3306
3
4 #Atlas后端连接的MySQL从库的IP和端口, @后面的数字代表权重, 用来作负载均衡, 若省略则默
  认为1, 可设置多项, 用逗号分隔
5 proxy-read-only-backend-addresses = 39.99.131.178:3306@1
6
7 # proxy-read-only-backend-addresses =
  39.99.131.178:3306@1,39.103.182.34:3306@2
```

这个是用来配置MySQL的账户与密码的, 我的MySQL的用户是buck,密码是hello,刚刚使用Atlas提供的工具生成了对应的加密密码

```
1 #用户名与其对应的加密过的MySQL密码, 密码使用PREFIX/bin目录下的加密程序encrypt加
  密, 下行的user1和user2为示例, 将其替换为你的MySQL的用户名和加密密码!
2 pwds = root:DAJn18cvzy8=
```



这是设置工作接口与管理接口的，如果ip设置的“0.0.0.0”就是说任意IP都可以访问这个接口，当然也可以指定IP和端口，方便测试我这边没有指定，工作接口的用户名密码与MySQL的账户对应的，管理员的用户密码与上面配置的管理员的用户密码对应。

```
1 #Atlas监听的工作接口IP和端口
2 proxy-address = 0.0.0.0:1234
3
4 #Atlas监听的管理接口IP和端口
5 admin-address = 0.0.0.0:2345
```

## 2.4.4 启动Atlas

```
1 [root@localhost bin]# ./mysql-proxyd test start
2 OK: MySQL-Proxy of test is started
```

使用如下命令，进入Atlas的管理模式 `mysql -h127.0.0.1 -P2345 -uuser -ppwd`，能进去说明Atlas正常运行，因为它会把自己当成一个MySQL数据库，所以在不需要数据库环境的情况下，也可以进入到MySQL数据库模式。

```
1 [root@localhost bin]# mysql -h127.0.0.1 -P2345 -uuser -ppwd
2 Welcome to the MySQL monitor.  Commands end with ; or \g.
3 Your MySQL connection id is 1
4 Server version: 5.0.99-agent-admin
5
6 Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
7
8 Oracle is a registered trademark of Oracle Corporation and/or its
9 affiliates. Other names may be trademarks of their respective
10 owners.
11
12 Type 'help;' or '\h' for help. Type '\c' to clear the current input
   statement.
13
14 mysql>
```

可以访问“help”表，来看MySQL管理员模式都能做些什么。可以使用SQL语句来访问

```
1 mysql> select * from help;
2 +-----+-----+
3 | command                | description
4 +-----+-----+
5 | SELECT * FROM help      | shows this help
6 | SELECT * FROM backends  | lists the backends and their state
7 | SET OFFLINE $backend_id | offline backend server, $backend_id is
   backend_ndx's id |
8 | SET ONLINE $backend_id  | online backend server, ...
```

```

 9 | ADD MASTER $backend      | example: "add master 127.0.0.1:3306", ...
   |
10 | ADD SLAVE $backend        | example: "add slave 127.0.0.1:3306", ...
   |
11 | REMOVE BACKEND $backend_id | example: "remove backend 1", ...
   |
12 | SELECT * FROM clients     | lists the clients
   |
13 | ADD CLIENT $client        | example: "add client 192.168.1.2", ...
   |
14 | REMOVE CLIENT $client     | example: "remove client 192.168.1.2", ...
   |
15 | SELECT * FROM pwds        | lists the pwds
   |
16 | ADD PWD $pwd              | example: "add pwd user:raw_password", ...
   |
17 | ADD ENPWD $pwd            | example: "add enpwd user:encrypted_password",
... |
18 | REMOVE PWD $pwd           | example: "remove pwd user", ...
   |
19 | SAVE CONFIG               | save the backends to config file
   |
20 | SELECT VERSION            | display the version of Atlas
   |
21 +-----+-----+
    -----+
22 16 rows in set (0.00 sec)
23
24 mysql>

```

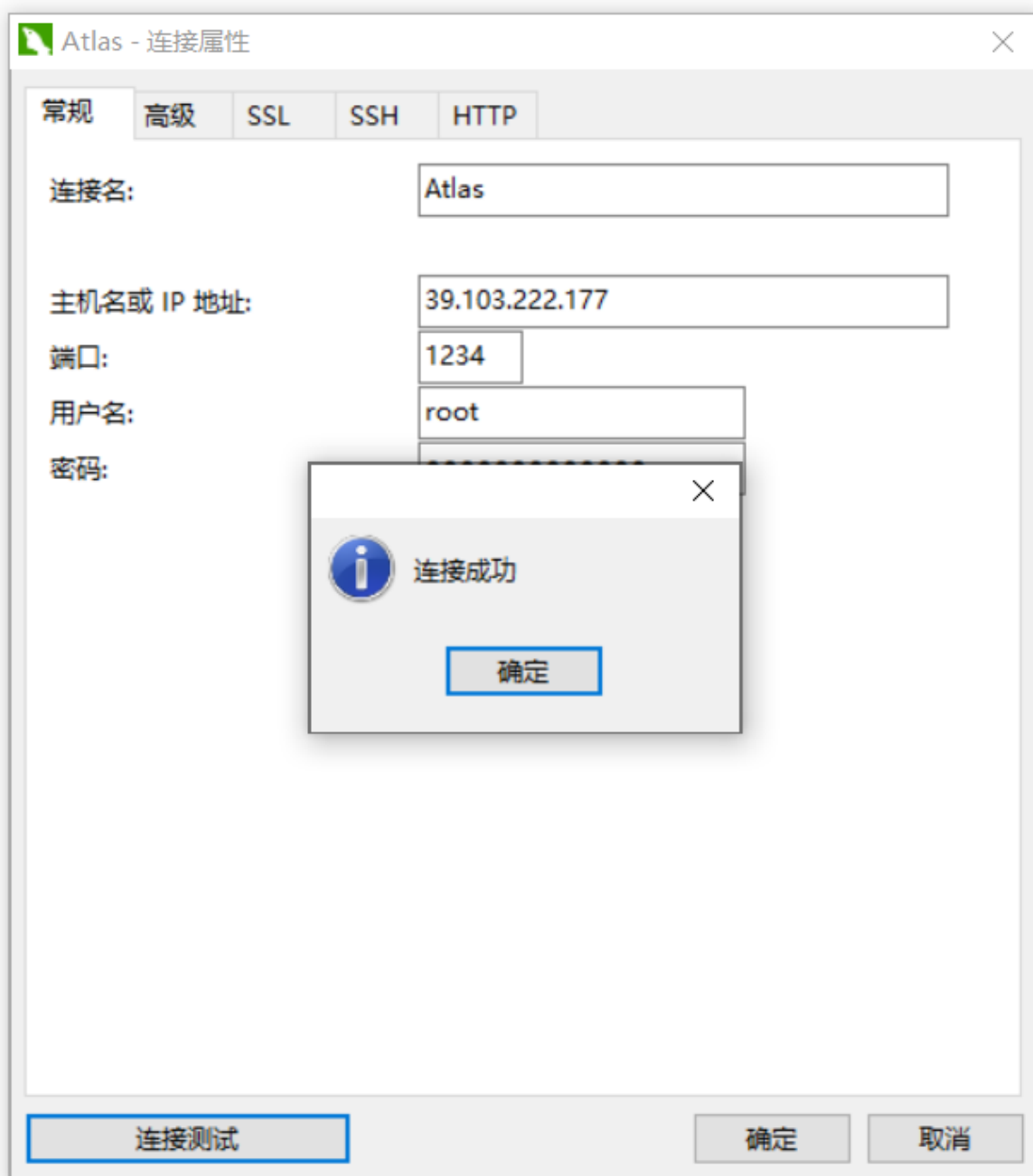
也可以使用工作接口来访问，使用命令 `mysql -h127.0.0.1 -P1234 -uroot -proot`

```

1 [root@localhost bin]# mysql -h127.0.0.1 -P1234 -uroot -proot
2 Welcome to the MySQL monitor.  Commands end with ; or \g.
3 Your MySQL connection id is 1
4 Server version: 5.0.81-log
5
6 Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.
7
8 Oracle is a registered trademark of Oracle Corporation and/or its
9 affiliates. Other names may be trademarks of their respective
10 owners.
11
12 Type 'help;' or '\h' for help. Type '\c' to clear the current input
    statement.
13
14 mysql>

```

如果工作接口可以进入了，就可以在Windows平台下，使用Navicat来连接数据库，填写对应的host，Port，用户名，密码就可以。



## 2.4.5 测试

```
1 # 进入Atlas的管理模式
2 mysql -h127.0.0.1 -P2345 -uuser -ppwd
3 # 查看所有节点
4 SELECT * FROM backends;
```

```
mysql> SELECT * FROM backends;
```

backend_ndx	address	state	type
1	39.103.222.177:3306	up	rw
2	39.99.131.178:3306	up	ro

```
2 rows in set (0.00 sec)
```

