MYSQL REPLICATION 主从复制全方位解决方案

惨绿少年 Linux运维, MySQL, 数据库 0评论 来源：本站原创 55℃ 字体： 小 中 大

1.1 主从复制基础概念

在了解主从复制之前必须要了解的就是数据库的二进制日志(binlog),主从复制架构大多基于二进制日志进行，二进制日志相关信息参考：http://www.cnblogs.com/clsn/p/8087678.html#_label6

1.1.1 二进制日志管理说明

二进制日志在哪？如何设置位置和命名?

在my.cnf文件中使用 log-bin = 指定；命名规则为 mysql-bin.000000 （后为6位数字）

二进制日志位置

```
mysql> show variables like '%log_bin%' ;
+---------------------------------+-----------------------------------------+
| Variable_name                   | Value                                   |
+---------------------------------+-----------------------------------------+
| log_bin                         | ON                                      |
| log_bin_basename                | /application/mysql/data/mysql-bin       |
| log_bin_index                   | /application/mysql/data/mysql-bin.index |
| log_bin_trust_function_creators | OFF                                     |
| log_bin_use_v1_row_events       | OFF                                     |
| sql_log_bin                     | ON                                      |
+---------------------------------+-----------------------------------------+
6 rows in set (0.06 sec)
```

日志命名

```
mysql> show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |      2979 |
| mysql-bin.000002 |       120 |
+------------------+-----------+
2 rows in set (0.00 sec)
```

二进制日志记录什么?

二进制日志中记录的是一个个完成的事件

二进制日志格式是怎样的?

推荐使用row格式

查看当前使用的日志 格式。

```
mysql> show variables like '%format%';
+-------------------------+-------------------+
| Variable_name           | Value             |
```

```
+--------------------------+-------------------+
| binlog_format            | ROW               |
| date_format              | %Y-%m-%d          |
| datetime_format          | %Y-%m-%d %H:%i:%s |
| default_week_format      | 0                 |
| innodb_file_format       | Antelope          |
| innodb_file_format_check | ON                |
| innodb_file_format_max   | Antelope          |
| time_format              | %H:%i:%s          |
+--------------------------+-------------------+
8 rows in set (0.00 sec)
```

二进制日志如何滚动?

每次重启都会刷新日志，也可以通过命令进行刷新 reset master;

二进制日志用来干嘛?

备份恢复

起始点的备份恢复

二进制日志的操作命令?

查看都有哪些二进制日志

```
mysql> show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |      2979 |
| mysql-bin.000002 |       167 |
| mysql-bin.000003 |       120 |
+------------------+-----------+
3 rows in set (0.00 sec)
```

查看当前使用的二进制日志文件

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000003 |      120 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

*binlog*相关详情参照：*http://www.cnblogs.com/clsn/p/8087678.html#_label6*

1.1.2 mysql传统备份方式和缺陷

1、二进制日志备份

2、mysqldump

a)必须有数据库服务器完成逻辑工作，需要更多地cpu周期

b)逻辑备份还原速度慢：需要MySQL加载和解释语句、转化存储格式、重建引擎

3、xtrabackup

a)文件大

b)不总是可以跨平台、操作系统和MySQL版本

1.1.3 MySQL主从复制能为我们做什么

高可用、辅助备份、分担负载

1.2 MySQL主从复制介绍

1.2.1 复制技术

作用

> 1.保证数据安全(异机实时备份)
>
> 2.保证服务持续运行（宕机接管）

主从复制实现基本原理

> 1.自带功能，复制是 MySQL 的一项功能，允许服务器将更改从一个实例复制到另一个实例。
>
> 2.主服务器将所有数据和结构更改记录到二进制日志中。
>
> 3.从属服务器从主服务器请求该二进制日志并在本地应用其内容。即通过把主库的binlog传送到从库，从新解析应用到从库。

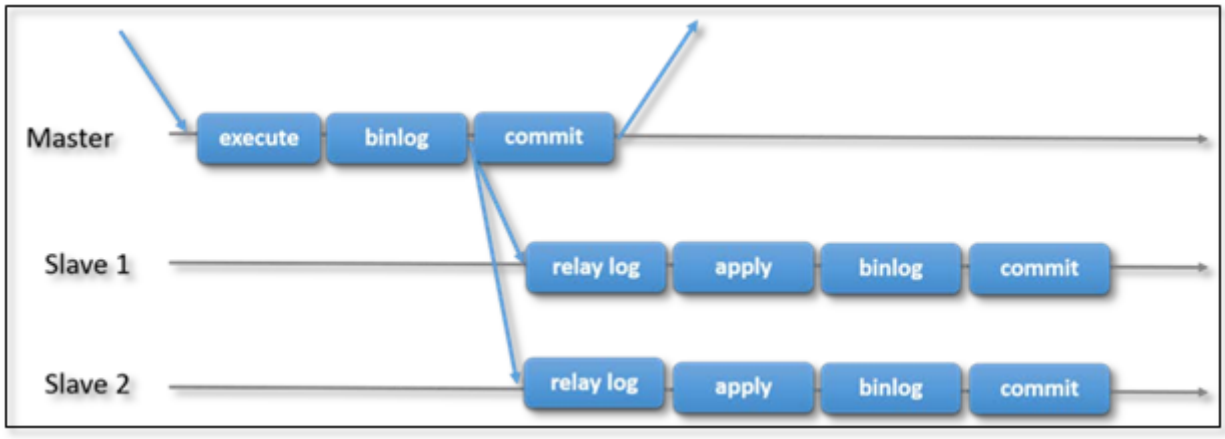
1.2.2 复制架构

mysql复制的应用常见场景:

> 应用场景1：从服务器作为主服务器的实时数据备份
>
> 应用场景2：主从服务器实现读写分离，从服务器实现负载均衡
>
> 应用场景3：把多个从服务器根据业务重要性进行拆分访问

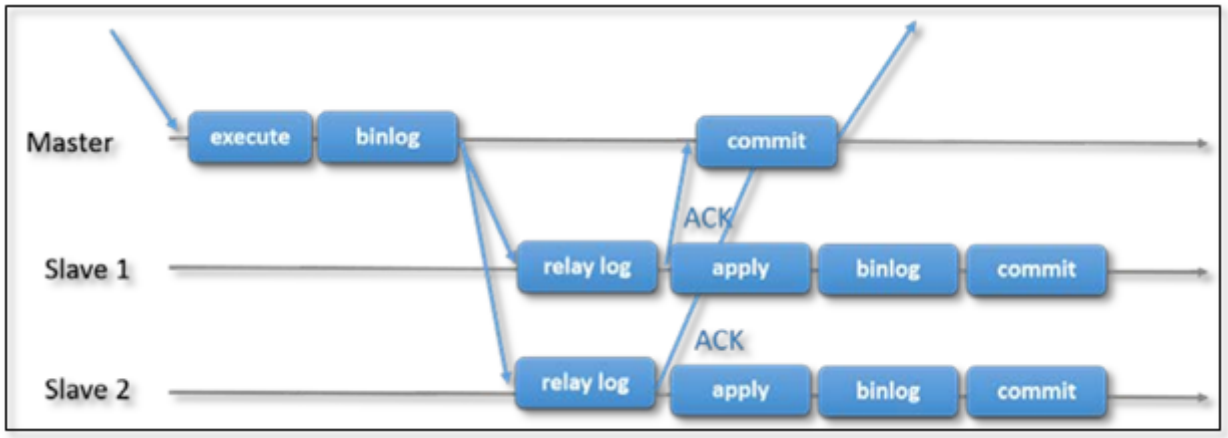
1.2.2.1 主–从复制

传统的 MySQL 复制提供了一种简单的主–从复制方法。 有一个主，以及一个或多个从。 主节点执行和提交事务，然后将它们（异步地）发送到从节点，以重新执行（在基于语句的复制中）或应用（在基于行的复制中）。 这是一个 shared-nothing 的系统，默认情况下所有 server 成员都有一个完整的数据副本。



(图)MySQL 异步复制

还有一个半同步复制，它在协议中添加了一个同步步骤。 这意味着主节点在提交时需要等待从节点确认它已经接收到事务。只有这样，主节点才能继续提交操作。



(图)MySQL 异步复制

在上面的两个图片中，可以看到传统异步 MySQL 复制协议（以及半同步）的图形展示。 蓝色箭头表示在不同 server 之间或者 server 与 client 应用之间的信息交互。

1.2.3 MySQL主从复制原理介绍

复制过程:

> 1、开启binlog日志，通过把主库的binlog传送到从库，从新解析应用到从库。
>
> 2、复制需要3个线程（dump、io、sql）完成，5.6从库多个sql。
>
> 3、复制是异步的过程。主从复制是异步的逻辑的SQL语句级的复制。

复制前提:

> 1、主服务期一定要打开二进制日志
>
> 2、必须两台服务器（或者是多个实例）

3、从服务器需要一次数据初始化

3.1如果主从服务器都是新搭建的话，可以不做初始化

3.2如果主服务器已经运行了很长时间了，可以通过备份将主库数据恢复到从库。

4、主库必须要有对从库复制请求的用户。

5、从库需要有relay-log设置，存放从主库传送过来的二进制日志 show variables like ‘%relay%’;

6、在第一次的时候，从库需要change master to 去连接主库。

7、change master信息需要存放到master.info中 show variables like ‘%master_info%’;

8、从库怎么知道，主库发生了新的变化?通过relay-log.info记录的已经应用过的relay-log信息。

9、在复制过程中涉及到的线程

从库会开启一个IO thread(线程)，负责连接主库，请求binlog，接收binlog并写入relay-log。

从库会开启一个SQL thread(线程)，负责执行relay-log中的事件。

主库会开启一个dump thrad(线程)，负责响应从IO thread的请求。

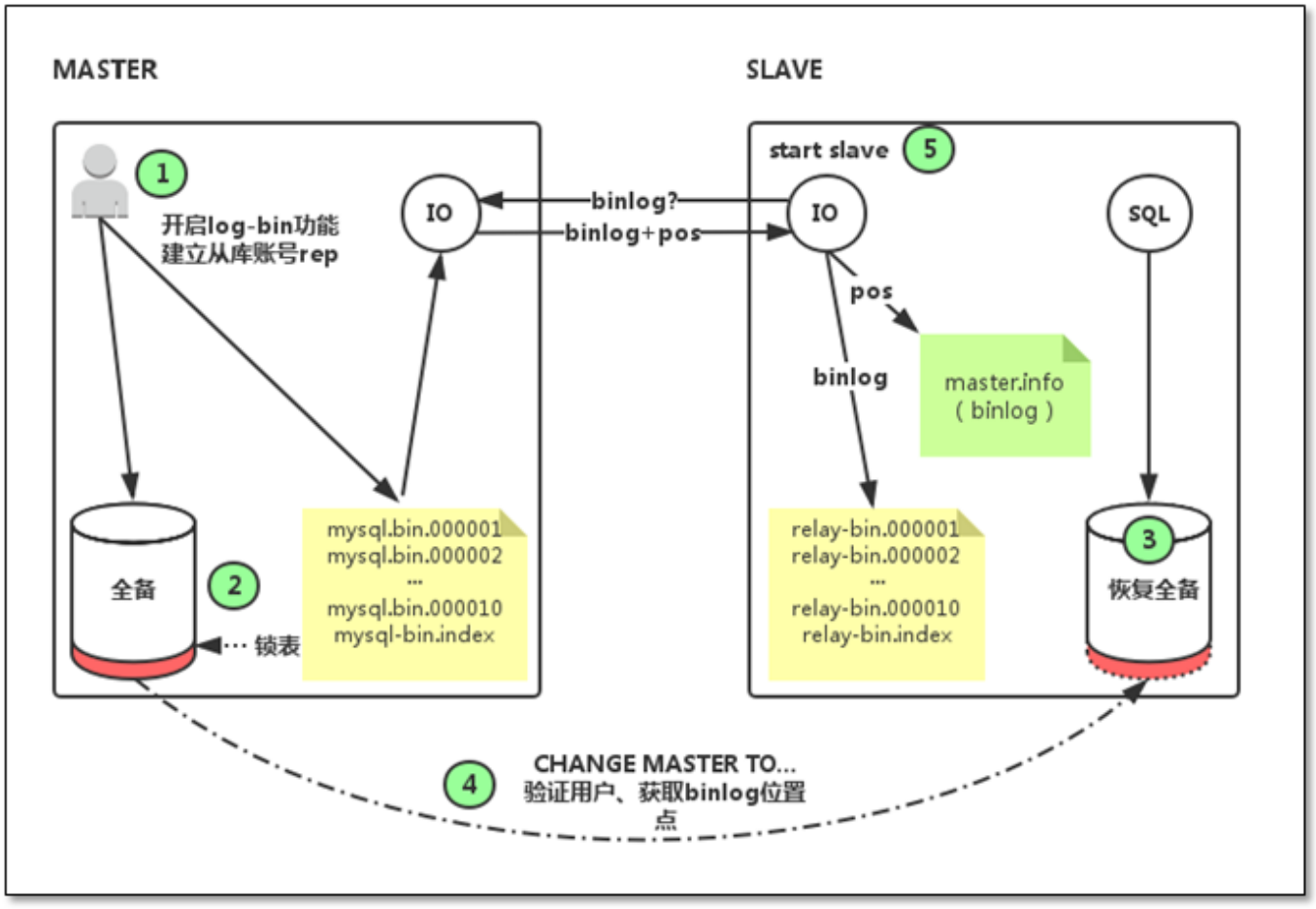
主从怎么实现的?

1、通过二进制日志

2、至少两台（主、从）

3、主服务器的二进制日志“拿”到从服务器上再运行一遍。

4、通过网络连接两台机器，一般都会出现延迟的状态。也可以说是异步的。



1.2.4 执行原理–第一次开启主从过程

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 evnets，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,默认情况下，已经应用过的relay会自动被清理purge。
>
> 到此位置，一次主从复制就完成

一旦主从运行起来：

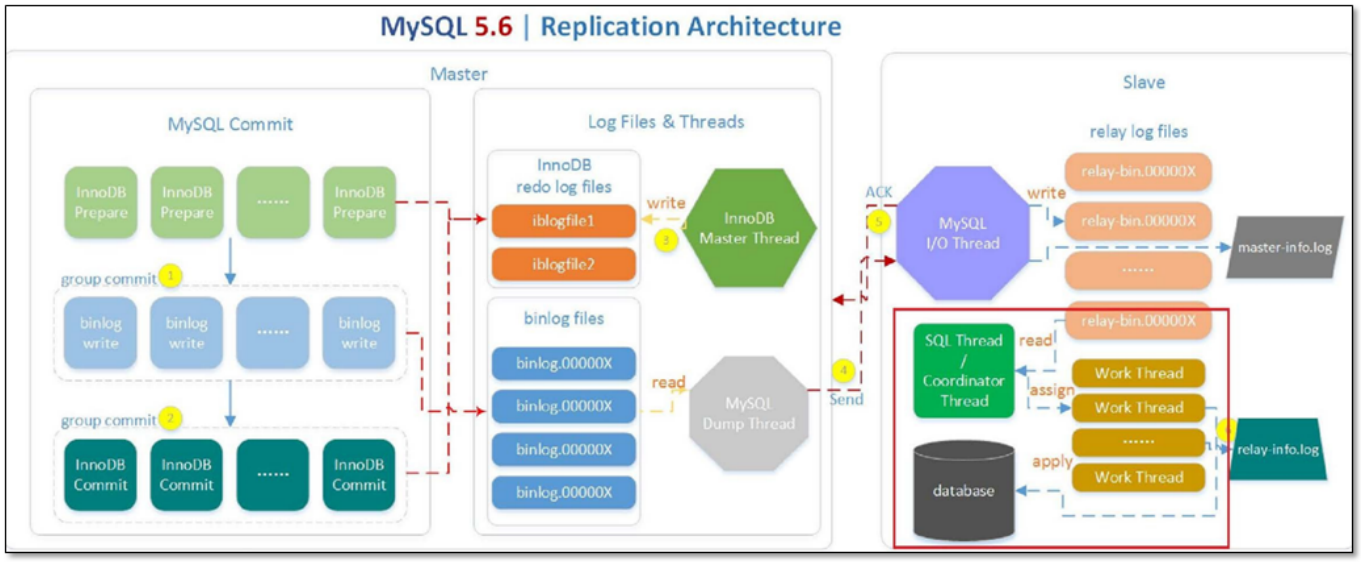
就不需要手工执行change master to，

因为信息都会被存放到master.info

（user、password、port、ip，上次获取过的binlog信息file和position）中

其他的过程都是一样的

1.2.4.1 详细的mysql replication 过程



1.3 主从搭建配置

本次主从搭建使用mysql多实例进行实验。多实例配置参考文档进行配置：*http://www.cnblogs.com/clsn/p/8038964.html#_label8*

1.3.1 多实例数据库slave配置

系统环境说明：

```
[root@db02 ~]# cat /etc/redhat-release
CentOS release 6.9 (Final)
[root@db02 ~]# uname -r
2.6.32-696.el6.x86_64
[root@db02 ~]# /etc/init.d/iptables status
iptables: Firewall is not running.  # 注意：务必关闭防火墙（iptables  selinux）
[root@db02 ~]# getenforce
Disabled
[root@db02 ~]# mysql --version
mysql  Ver 14.14 Distrib 5.6.36, for Linux (x86_64) using  EditLine wrapper
```

1、启动多实例数据库

```
[root@db02 ~]# /data/3306/mysql start
Starting MySQL...
[root@db02 ~]# /data/3307/mysql start
Starting MySQL...
```

2、配置文件说明：

***master* 配置文件说明：**

```
[root@db02 ~]# cat /data/3306/my.cnf
[client]
port            = 3306
socket          = /data/3306/mysql.sock

[mysqld]
user    = mysql
port    = 3306
socket  = /data/3306/mysql.sock
basedir = /application/mysql
datadir = /data/3306/data
log-bin = /data/3306/mysql-bin
server-id = 6  # server id 不能相同
skip_name_resolve = 0 # 跳过域名解析参数


[mysqld_safe]
log-error=/data/3306/mysql_3306.err
pid-file=/data/3306/mysqld.pid
```

***slave* 配置文件说明：**

```
[root@db02 ~]# cat /data/3307/my.cnf
[client]
port            = 3307
socket          = /data/3307/mysql.sock

[mysqld]
user    = mysql
port    = 3307
socket  = /data/3307/mysql.sock
basedir = /application/mysql
datadir = /data/3307/data
log-bin = /data/3307/mysql-bin
server-id = 7  # server id 不能相同
skip_name_resolve = 0  # 跳过域名解析参数
read_only = 1  # 从库只读 （非root用户 ）

[mysqld_safe]
log-error=/data/3307/mysql_3307.err
pid-file=/data/3307/mysqld.pid
```

3、在主库创建复制用户

登陆到主数据库中：

```
mysql -uroot -p123 -S /data/3306/mysql.sock
```

创建授权用户，注意是slave用户。

```
grant replication slave  on *.* to repl@'10.0.0.%' identified by '123';
```

4、初始化从库数据

备份主库当前数据

```
 mysqldump -uroot -p123 -A -B -F --master-data=2  -S /data/3306/mysql.sock >/tmp/full.sql
```

部分参数说明：（详细参照*http://www.cnblogs.com/clsn/p/8138015.html#_label2*）

-F 刷新二进制日志

–master-data [= #]这会导致二进制日志的位置和文件名被追加到输出中。

如果等于1，则将其打印为CHANGE MASTER命令; 如果等于2，那么该命令将以注释符号为前缀。

到从库进行恢复

```
 mysql -uroot -p123 -S /data/3307/mysql.sock
```

恢复备份的数据

```
set sql_log_bin=0;
source /tmp/full.sql
```

5、开启从库复制

查看备份的当前使用的文件及POS号

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000012 |      120 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

登入数据库，进行slave配置。

```
mysql -uroot -p123 -S /data/3307/mysql.sock
CHANGE MASTER TO
  MASTER_HOST='10.0.0.52',
  MASTER_USER='repl',
  MASTER_PASSWORD='123',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='mysql-bin.000012',
  MASTER_LOG_POS=120;
start  slave;  # 启动从库复制
```

该配置想关说明可以通过 help 获得。

```
mysql> help CHANGE MASTER TO
CHANGE MASTER TO
  MASTER_HOST='master2.mycompany.com',
  MASTER_USER='replication',
  MASTER_PASSWORD='bigs3cret',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='master2-bin.001',
  MASTER_LOG_POS=4,
  MASTER_CONNECT_RETRY=10;
```

1.3.2 测试主从同步

查看slave库的状态

主要查看 Slave_IO_Running 与 Slave_SQL_Running 是否都为Yes

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.0.0.52
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000010
          Read_Master_Log_Pos: 842
               Relay_Log_File: 3307-relay-bin.000018
                Relay_Log_Pos: 283
        Relay_Master_Log_File: mysql-bin.000010
             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: 842
              Relay_Log_Space: 455
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
```

```
38                Last_IO_Error:
39               Last_SQL_Errno: 0
40               Last_SQL_Error:
41   Replicate_Ignore_Server_Ids:
42              Master_Server_Id: 6
43                   Master_UUID: 4f344556-e0ab-11e7-9138-000c29d60ab3
44              Master_Info_File: /data/3307/data/master.info
45                     SQL_Delay: 0
46           SQL_Remaining_Delay: NULL
47       Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O thread
48            Master_Retry_Count: 86400
49                   Master_Bind:
50       Last_IO_Error_Timestamp:
51      Last_SQL_Error_Timestamp:
52                Master_SSL_Crl:
53            Master_SSL_Crlpath:
54            Retrieved_Gtid_Set:
55             Executed_Gtid_Set:
56                 Auto_Position: 0
57 1 row in set (0.00 sec)
```

View Code 从库状态信息

在主库进行操作，在从库验证

```
[root@db02 ~]# mysql -uroot -p123  -S /data/3306/mysql.sock
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)
mysql> create database clsn;
Query OK, 1 row affected (0.00 sec)
```

在从库上可以看到该数据库已创建

```
[root@db02 ~]# mysql -uroot -p123 -S /data/3307/mysql.sock
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| clsn               |
| mysql              |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

至此*mysql*主从复制就搭建完成

1.3.3 忘记数据库密码?

```
shell> /application/mysql/bin/mysqld_safe  --defaults-file=/data/3306/my.cnf    --skip-grant-tab
mysql> update user set password=password('123') where user='root' and host='localhost';
mysql> flush privileges;
```

1.3.4 主从复制状态失败的原因?

```
Last_IO_Error: error reconnecting to master 'repl@10.0.0.52:3306' - retry-time: 60  retries: 1
```

原因:

> 1、主机没启动，或者宕机，检查主库的状态。
>
> 2、网络通信问题，使用ping命令进行检查；或使用mysql命令进行shell端登陆测试
>
> 3、防火墙，selinux（务必检查）。
>
> 4、复制用户和密码、端口号、地址有问题，使用mysql命令进行shell端登陆测试。
>
> 5、mysql自动解析，会将连接的IP解析成主机名（skip-name-resolve = 0）写入my.cnf文件即可。
>
> 6、从库IO异常关闭，通过show slave status \G 进行查看。

1.4 MySQL主从复制常见问题

1.4.1 从库binlog落后主库binlog?

从库记录的已经主库已经给我传送的binlog事件的坐标，一般在繁忙的生产环境下会落后于主库

```
show master status\G  --- 主
show slave status \G  --- 从
Master_Log_File: mysql-bin.000007
Read_Master_Log_Pos: 729
```

落后太远的原因:

> 硬件条件有关的，机器磁盘IO性能不足。
>
> 主要还是网络问题，网络传输的性能。
>
> 主库存放二进制日志的存储性能太低，建议binlog日志存咋SSD中。

> 主库DUMP线程太繁忙，主要发生在一主多从的环境下。
>
> 从库IO线程太忙
>
> 人为控制（delay节点、延时节点 ）

1.4.2 主库update，从库迟迟的没有更新。

特殊情况：日志已经传过来了，数据并没有同步

一般情况：

> 1、没开启SQL线程
>
> 2、传的东西有问题（你要做的事情，我提前已经做了，不想重复做了，然后他就死了）
>
> 3、SQL线程忙。
>
> 4、人为控制了【delay(从库)节点、延时节点，一般生产设置为3-6小时之间，可以保证过去3-6小时之间的误操作，可以避免】。

1.4.3 主从复制延时配置(从库配置)

停止从库复制

```
mysql>stop slave;
Query OK, 0 rows affected (0.01 sec)
```

修改延时参数，MASTER_DELAY，单位位S （秒）。

```
mysql>CHANGE MASTER TO MASTER_DELAY = 30;
Query OK, 0 rows affected (0.07 sec)
```

启动从库复制

```
mysql>start slave;
Query OK, 0 rows affected (0.07 sec)
```

查看配置是否生效

```
mysql> show slave status \G
……
              SQL_Delay: 30
```

1.4.4 从库安全配置（其他用户只读）

修改my.cnf配置文件，添加只读参数

```
read_only = 1    ====> 控制普通用户
innodb_read_only = 1  ====> 控制root用户，正常情况不要加
```

添加完成后重启数据库

```
mysql> show variables like '%read_only%';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| innodb_read_only | OFF   |
| read_only        | ON    |
| tx_read_only     | OFF   |
+------------------+-------+
3 rows in set (0.00 sec)
```

延时从库: delay节点、延时节点

1.4.5 主从复制故障及解决（跳过错误）

命令行设置

```
stop slave; #<==临时停止同步开关。
set global sql_slave_skip_counter = 1 ; #<==将同步指针向下移动一个，如果多次不同步，可以重复操作
start slave;
```

在配置文件修改，设置要跳过的pos

```
/etc/my.cnf
slave-skip-errors = 1032,1062,1007
```

在mysql中可以跳过某些错误,但是最好的解决办法，重新搭建主从复制。

1.4.6 延时节点概念 –> SQL线程延时?

```
Last_SQL_Errno: 0
Last_SQL_Error:
```

原因:

> 1、主库做操作的对象，在从库不存在
>
> 2、主库做操作的对象属性不一致。
>
> 3、主库做操作的对象，从库已经存在
>
> ……

1.4.7 Slave_*_Running： ?

> 1、Slave_IO_Running I/O 线程正在运行、未运行还是正在运行但尚未连接到主服务器。可能值分别为Yes、No 或 Connecting。
>
> 2、Slave_SQL_Running SQL 线程当前正在运行、未运行,可能值分别为 Yes、No 主服务器日志坐标：
>
> 3、Master_Log_File 和 Read_Master_Log_Pos 标识主服务器二进制日志中 I/O 线程已经传输的最近事件的坐标。
>
> 4、如果Master_Log_File和Read_Master_Log_Pos 的值远远落后于主服务器上的那些值，这表示主服务器与从属服务器之间事件的网络传输可能存在延迟。

1.4.8 中继日志坐标

> a) Relay_Log_File 和 Relay_Log_Pos 列标识从属服务器中继日志中 SQL 线程已经执行的最近事件的坐标。这些坐标对应于 Relay_Master_Log_File 和 Exec_Master_Log_Pos 列标识的主服务器二进制日志中的坐标。
>
> b) 如果 Relay_Master_Log_File 和 Exec_Master_Log_Pos 列的输出远远落后于 Master_Log_File 和Read_Master_Log_Pos 列（表示 I/O 线程的坐标），这表示 SQL 线程（而不是 I/O 线程）中存在延迟。即，它表示复制日志事件快于执行这些事件。

1.4.9 单一主从需要改变的地方

从库的作用

> 1、相当于实时备份
>
> 2、使用从库备份
>
> 3、一主多从应对读多的业务需求

如果，从库只做备份服务器用，那么主库的压力会不减反增。因为，所有的业务都在主库实现，读和写，dump线程读取并投递binlog

解决方案:

> （1）可不可以挪走一部分读业务到从库，读写分离

> （2） 一主多从应对读多的业务需求，一旦发展成这个架构，dump线程投递binlog的压力更大
>
> （3） 多级主从，采用中间库缓解主库dump的压力，会出现中间库瓶颈的问题，选择blackhole引擎，看性能与安全的权衡
>
> （4）双主模型：缓解，数据一致性难保证
>
> （5）环装复制

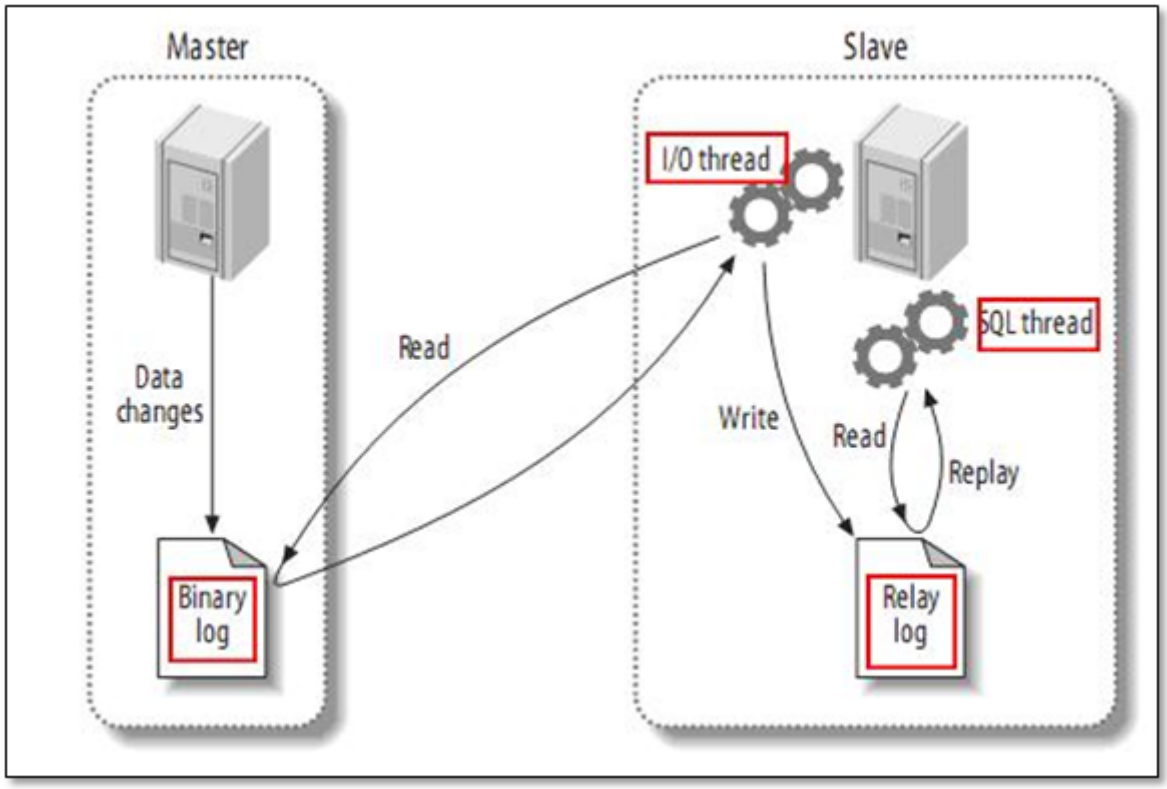
1.5 【生产案例】主从复制事故

1.5.1 发生背景

> 1、有一台已经工作很久的单机mysql数据。在2017年12月24日的平安夜，我司购物网站宕机了。机器物损坏，系统硬盘报废。
>
> 2、在接到一条短信告知服务器宕机，数据库连不上。当时的我一脸懵逼的还在开party，谁能想到在这样一个阖家欢乐的时刻发生这样的事情。
>
> 3、随之我火速赶回公司处理事故。首先更换硬盘，从备份服务器上拉取备份数据，用备份恢复宕机的时刻数据，经历40分钟后所有应用恢复正常。
>
> 4、经历这次事故，决心修改数据库架构，我跟领导承诺，保证改完之后，出现类似故障能在5-10分钟恢复业务。把原来的停机时间缩短4-8倍。

1.5.2 搭建流程

1.5.2.1 架构设计



修改架构采用数据库主从同步，能保证数据的安全，提高事故发生的恢复速度。

1.5.2.2 架构实施

（1）准备一台新机器，配置、系统、环境等与原数据库保持一致。

（2）在主库检查binlog开关，没有开启将其开启，检查server_id 与 auto.cnf文件中的uuid 是否唯一。

（3）主库创建授权复制的用户，授权 replication slave。

（4）备份主库上现有数据，恢复到从库中，推荐使用mysqldump，在访问低谷的时候做。

（5）在从库上开启binlog和relaylog,server_id。

（6）在从库配置change master to 信息：在第一次开启主从的时候，告诉从库user password host port，复制binlog的起点file、position。

（7）start slave 开启主从复制。

到此历经千辛万苦主从复制搭建完成。

1.5.3 测试主从切换

> （1） 主从的可用性测试：在主库中插入数据，在从库查看有没有。
>
> （2） 主从快速恢复演练
>
> a） 在一个月黑风高夜选一个业务不繁忙时间点，人工宕掉主库。
>
> b） 将从库定为主库，查看从库的日志量（master.info、relay-log.info）

c） 判断主从日志的差距（master.info,show master status）

d） 恢复后发现偏差，就人为登录到主库（备份服务器也行）中，把截取差距的binlog日志，并传送到从库进行数据补偿。

e） 此时从库数据现在已经和主库一致。

f） reset master，reset slave

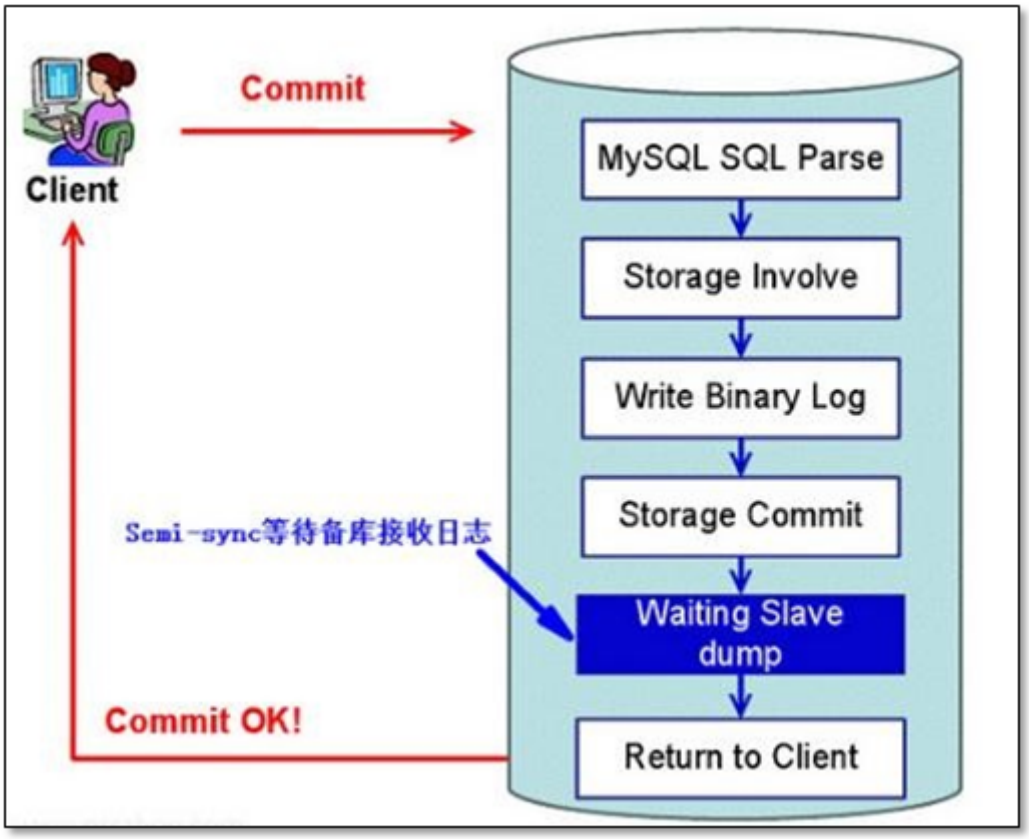
g） 应用割接到从库，将应用数据库IP指向从库IP，测试应用。

（3）小结：经历里这次测试，主从见的切换历时6分32秒，比之前缩短许多，但是感觉还差点什么，以后再补吧。

1.6 mysql半同步复制

MySQL复制默认是异步复制，存在一定的概率备库与主库的数据是不对等的,如果Master宕机，事务在Master上已提交，但很可能这些事务没有传到任何的Slave上，此时Slave也可能会丢失事务。在半同步复制的架构下，当master在将自己binlog发给slave上的时候，要确保slave已经接受到了这个二进制日志以后，才会返回数据给客户端。

半同步复制介于异步复制和全同步复制之间，主库在执行完客户端提交的事务后不是立刻返回给客户端，而是等待至少一个从库接收到并写到relay log中才返回给客户端。相对于异步复制，半同步复制提高了数据的安全性，同时它也造成了一定程度的延迟，这个延迟最少是一个TCP/IP往返的时间。所以，半同步复制最好在低延时的网络中使用。



半同步复制的原理图

在5.6中加入了group commit特性之后，性能不比传统的异步复制差。

1.6.1 半同步复制的潜在问题

客户端事务在存储引擎层提交后，在得到从库确认的过程中，主库宕机了，此时，可能的情况有两种

事务还没发送到从库上

此时，客户端会收到事务提交失败的信息，客户端会重新提交该事务到新的主上，当宕机的主库重新启动后，以从库的身份重新加入到该主从结构中，会发现，该事务在从库中被提交了两次，一次是之前作为主的时候，一次是被新主同步过来的。

事务已经发送到从库上

此时，从库已经收到并应用了该事务，但是客户端仍然会收到事务提交失败的信息，重新提交该事务到新的主上。

1.6.2 半同步架构搭建

加载使用的插件

```
主库:
INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';
从从:
INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';
```

查看是否加载成功

```
show plugins;
```

启动半同步复制

```
主库:
SET GLOBAL rpl_semi_sync_master_enabled = 1;
从库:
SET GLOBAL rpl_semi_sync_slave_enabled = 1;
```

重启从库上的IO线程

```
STOP SLAVE IO_THREAD;
START SLAVE IO_THREAD;
```

查看是否在运行

```
主库:
show status like 'Rpl_semi_sync_master_status';
从库:
show status like 'Rpl_semi_sync_slave_status';
```

测试半同步复制

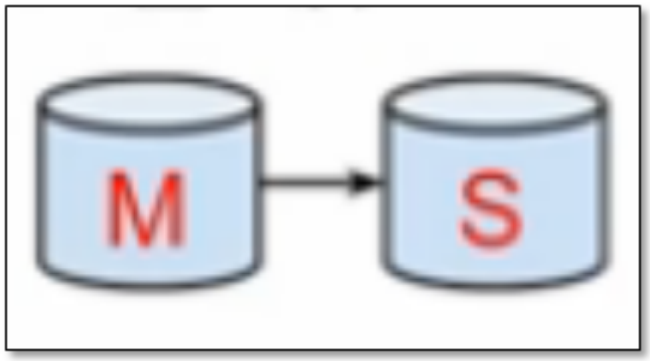
```
查看延迟时间:
show variables like '%rpl_semi_sync%';
从库:
stop slave;
主库:
create database clsn;
```

如果创建库的时间是设置的时间就成功了。

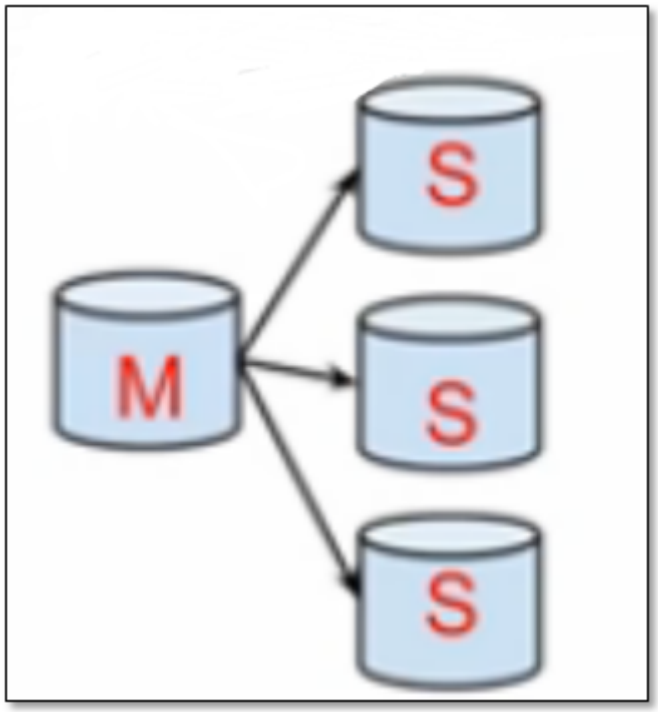
1.7 主从复制架构的演变

1.7.1 基本结构

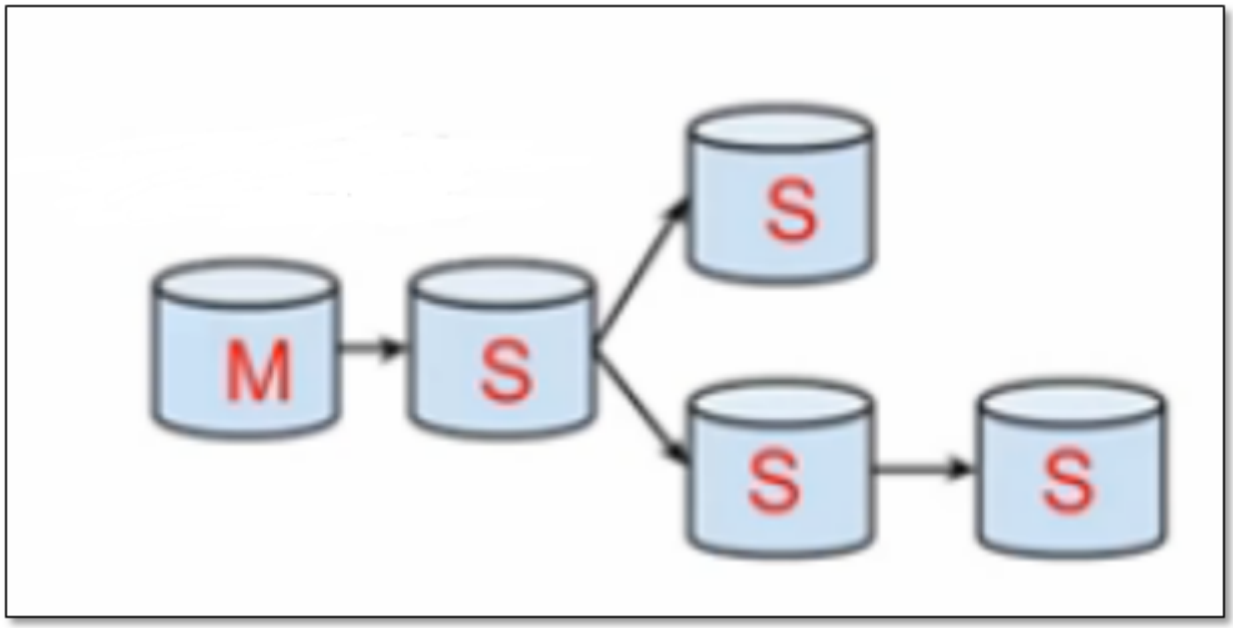
(1) 一主一从



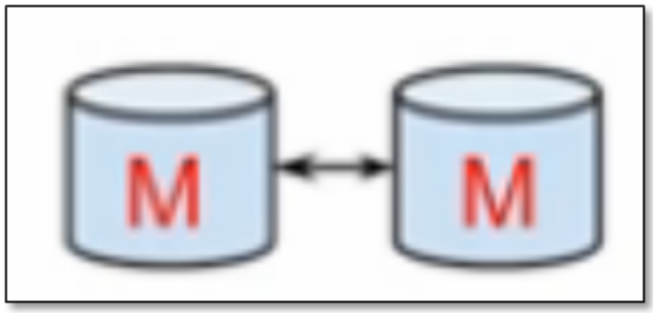
(2) 一主多从



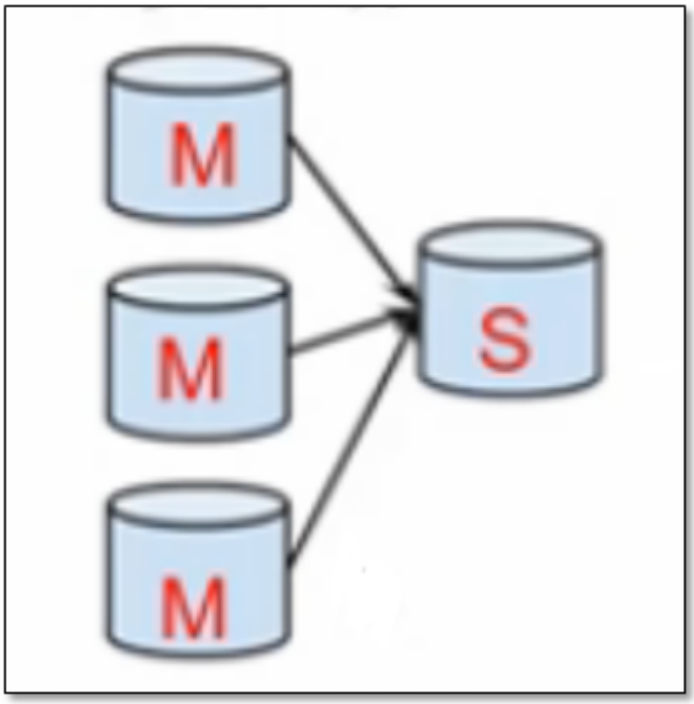
(3) 多级主从



（4）双主

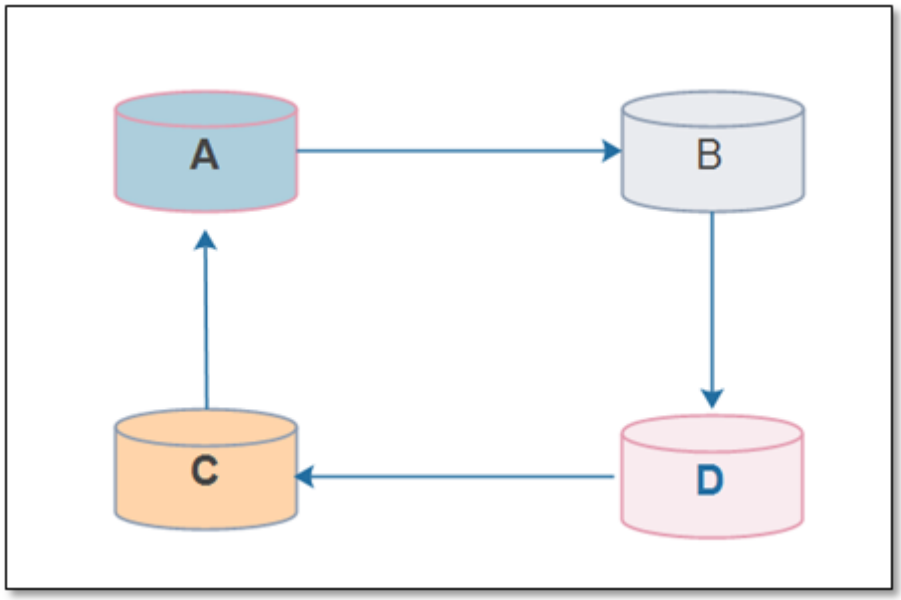


（5）多主一从（5.7之后开始支持）



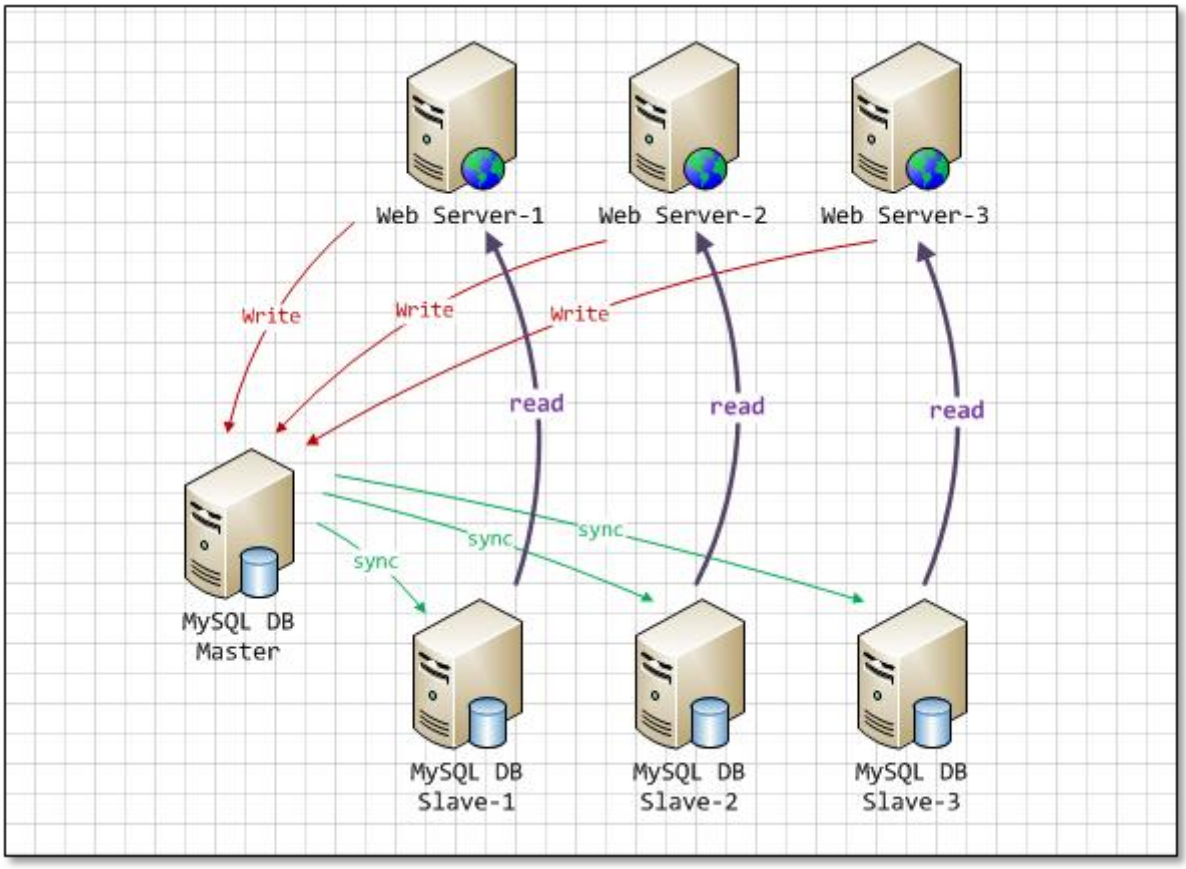
（6）循环复制

对于循环数据库镜像，就是多个数据库A、B、C、D等，对其中任一个数据库的修改，都要同时镜像到其它的数据库里。 replicate–same–server–id = **0**



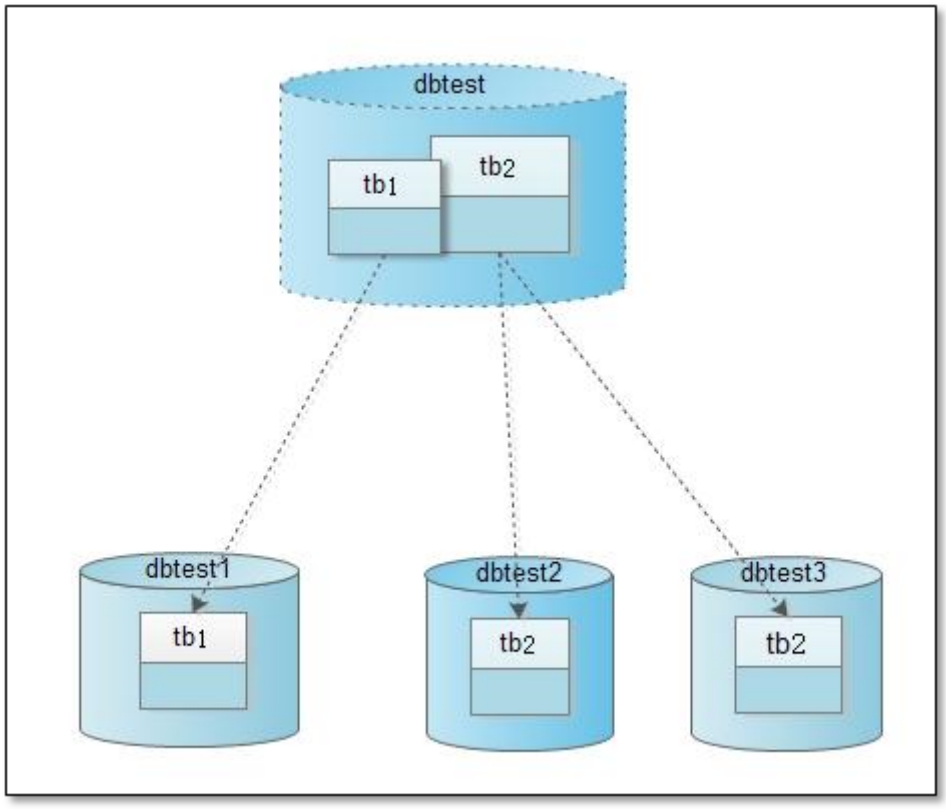
1.7.2 高级应用架构演变

（1）读写分离——MySQL proxy、amoeba、xx-dbproxy等。



一般来说都是通过主从复制（Master-Slave）的方式来同步数据，再通过读写分离（MySQL-Proxy）来提升数据库的并发负载能力这样的方案来进行部署与实施的。

（2）分库分表——cobar、自主研发等。

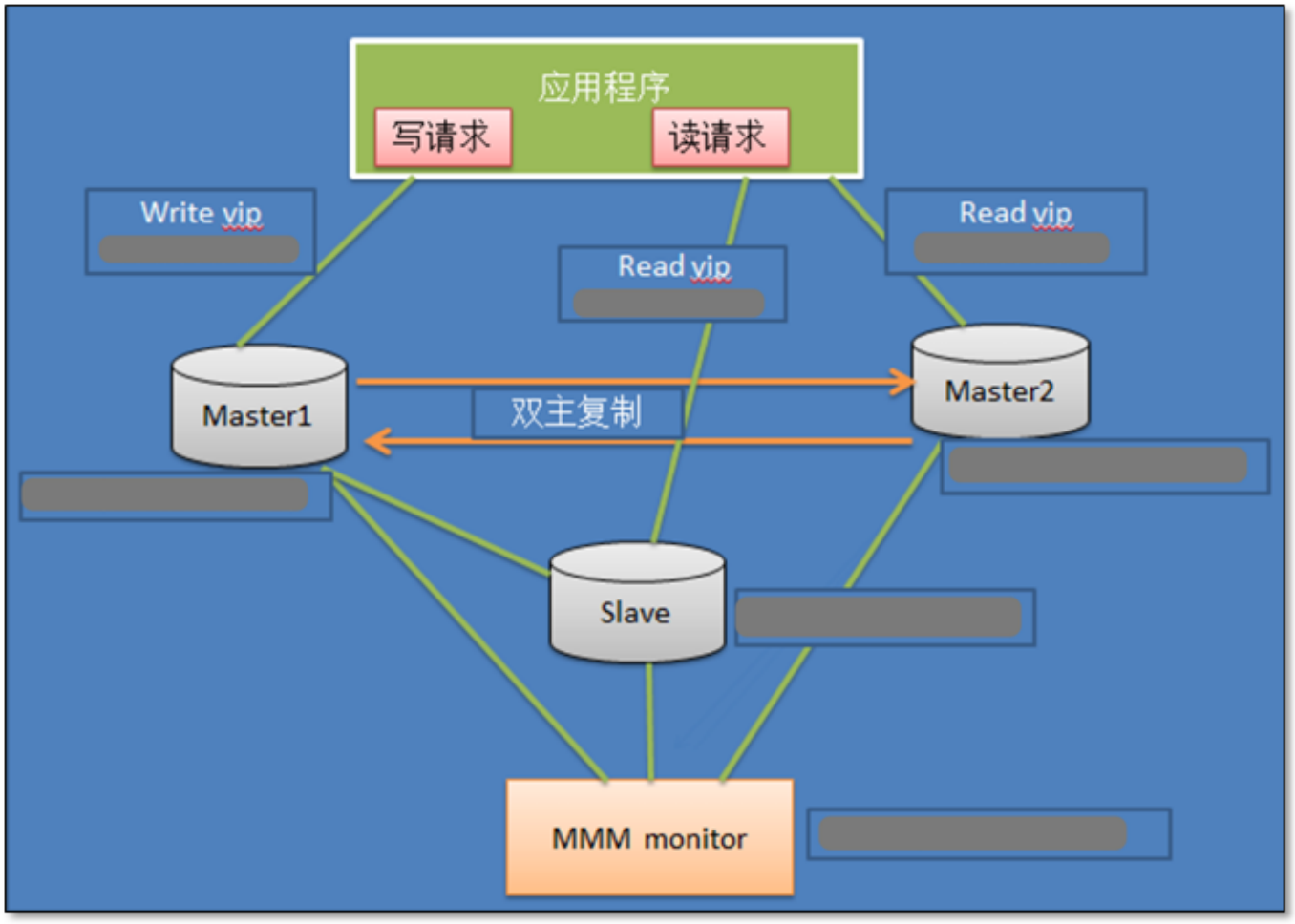


1) 系统对外提供的数据库名是dbtest,并且其中有两张表tb1和tb2。

2) tb1表的数据被映射到物理数据库dbtest1的tb1上。

3) tb2表的一部分数据被映射到物理数据库dbtest2的tb2上，另外一部分数据被映射到物理数据库dbtest3的tb2上。

（3）MMM架构——mysql-mmm（google）-使用极少



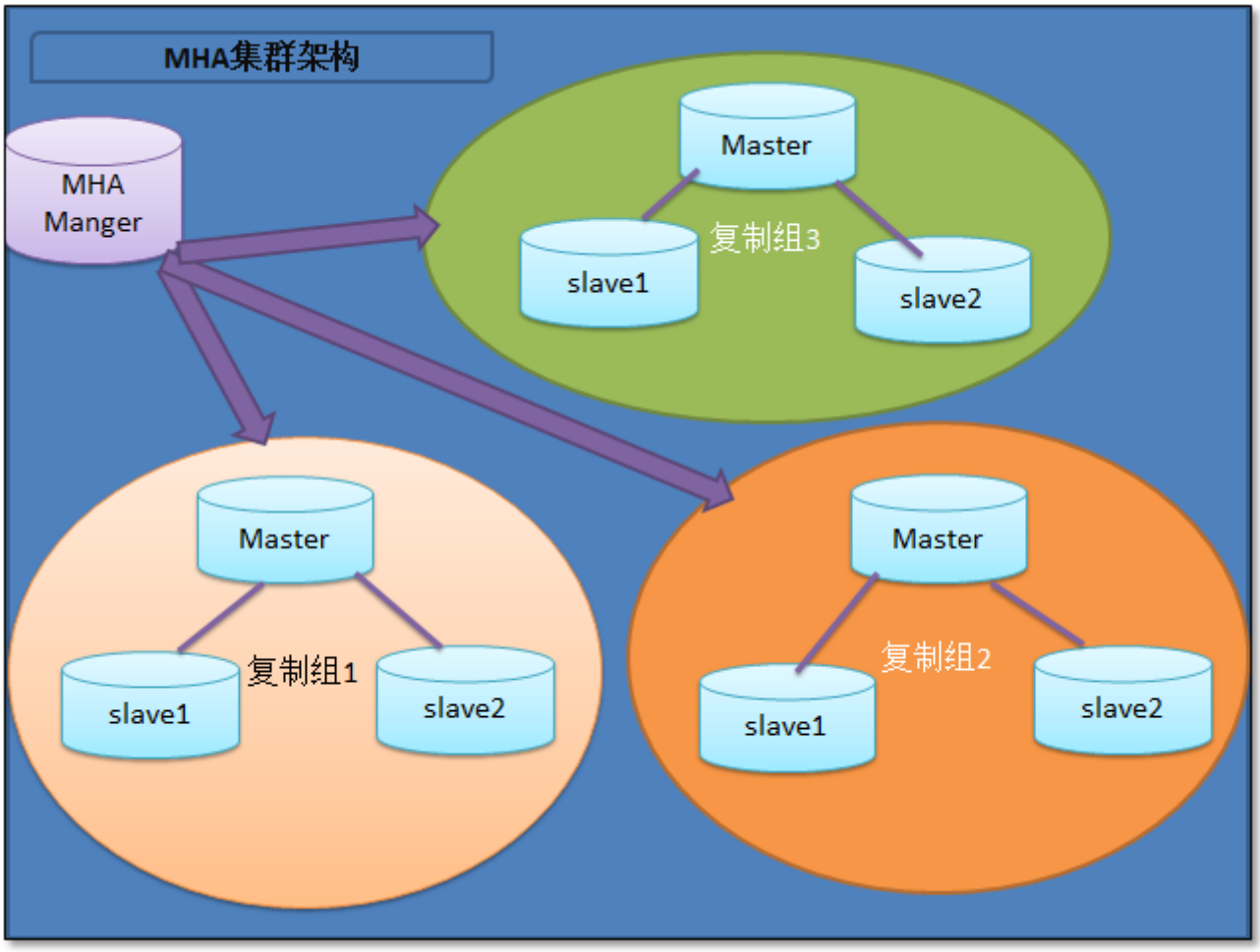
MMM（Master-Master replication manager for MySQL）是一套支持双主故障切换和双主日常管理的脚本程序。MMM使用Perl语言开发，主要用来监控和管理MySQL Master-Master（双主）复制，可以说是mysql主主复制管理器。

虽然叫做双主复制，但是业务上同一时刻只允许对一个主进行写入，另一台备选主上提供部分读服务，以加速在主主切换时刻备选主的预热，可以说MMM这套脚本程序一方面实现了故障切换的功能，另一方面其内部附加的工具脚本也可以实现多个slave的read负载均衡。

关于mysql主主复制配置的监控、故障转移和管理的一套可伸缩的脚本套件（在任何时候只有一个节点可以被写入），这个套件也能对居于标准的主从配置的任意数量的从服务器进行读负载均衡，所以你可以用它来在一组居于复制的服务器启动虚拟ip，除此之外，它还有实现数据备份、节点之间重新同步功能的脚本。

对于那些对数据的一致性要求很高的业务，非常不建议采用MMM这种高可用架构。

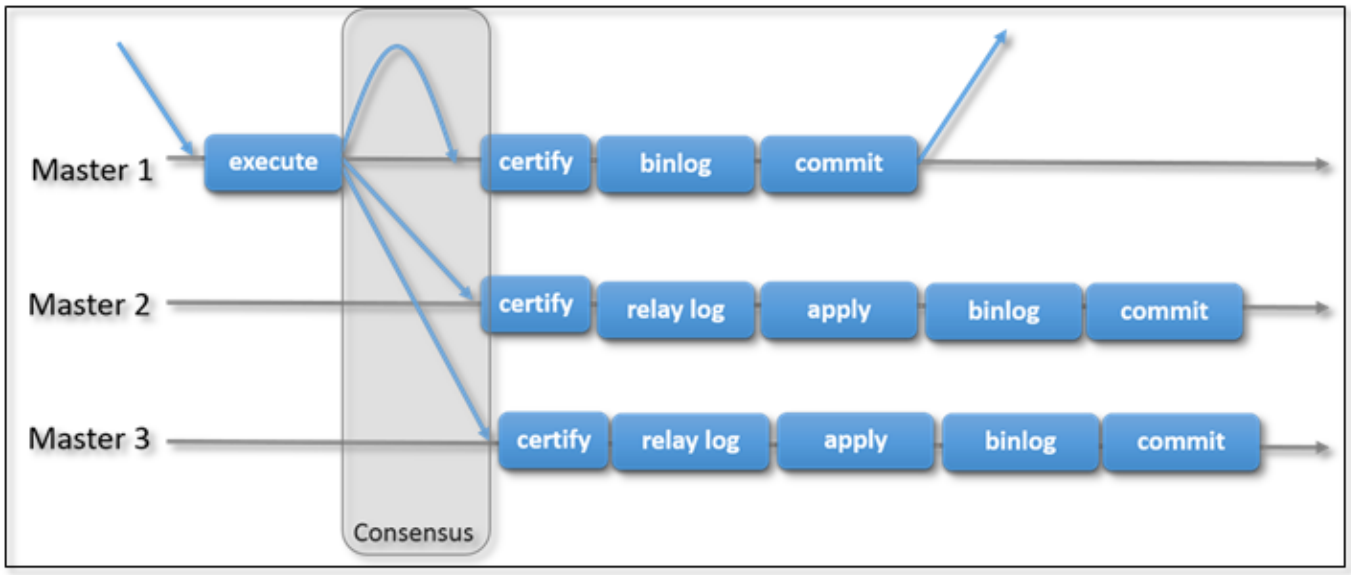
（4）MHA架构——mysql-master-ha（日本DeNa）



MHA（Master High Availability）目前在MySQL高可用方面是一个相对成熟的解决方案，它由日本DeNA公司youshimaton（现就职于Facebook公司）开发，是一套优秀的作为MySQL高可用性环境下故障切换和主从提升的高可用软件。在MySQL故障切换过程中，MHA能做到在0~30秒之内自动完成数据库的故障切换操作，并且在进行故障切换的过程中，MHA能在最大程度上保证数据的一致性，以达到真正意义上的高可用。

该软件由两部分组成：MHA Manager（管理节点）和MHA Node（数据节点）。

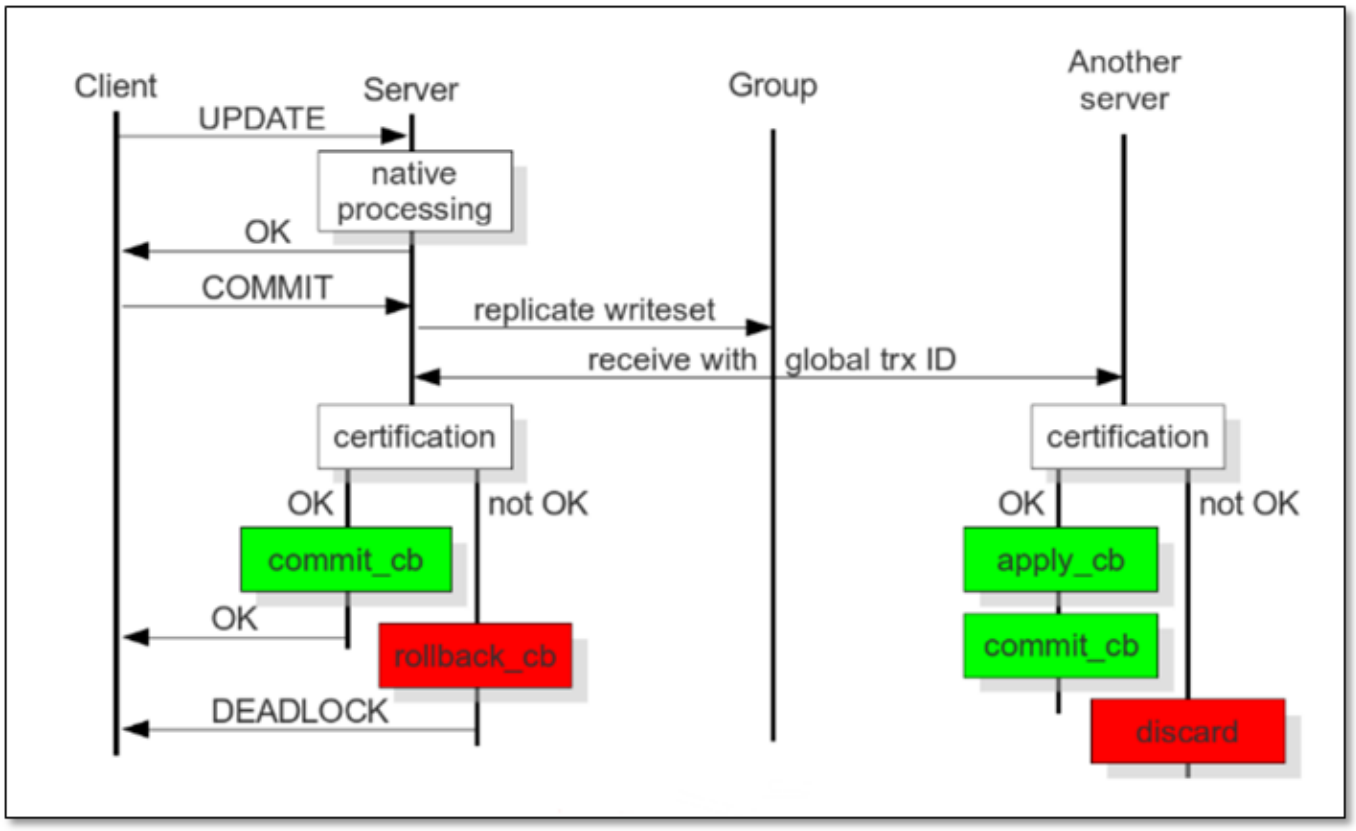
(5) MGR– 5.7 新特性 MySQL Group replication



基于传统异步复制和半同步复制的缺陷——数据的一致性问题无法保证，MySQL官方在5.7.17版本正式推出组复制（MySQL Group Replication，简称MGR）。

由若干个节点共同组成一个复制组，一个事务的提交，必须经过组内大多数节点（N / 2 + 1）决议并通过，才能得以提交。如上图所示，由3个节点组成一个复制组，Consensus层为一致性协议层，在事务提交过程中，发生组间通讯，由2个节点决议(certify)通过这个事务，事务才能够最终得以提交并响应。

（6）PXC、MySQL Cluster、galera cluster架构



在PXC中，一次数据写入在各个节点间的验证/回滚流程

***PXC*架构的优点：**

a） 服务高可用；

b） 数据同步复制(并发复制)，几乎无延迟；

c） 多个可同时读写节点，可实现写扩展，不过最好事先进行分库分表，让各个节点分别写不同的表或者库，避免让galera解决数据冲突；

d） 新节点可以自动部署，部署操作简单；

e） 数据严格一致性，尤其适合电商类应用；

f） 完全兼容MySQL；

1.7.3 分库分表简单实践

实践中使用的为world数据库，为mysql官方提供，详情参照：http://www.cnblogs.com/clsn/p/8087417.html#_label0

第一个里程碑:创建新表

```
CREATE TABLE `country_1` (
  `Code` char(3) NOT NULL DEFAULT '',
  `Name` char(52) NOT NULL DEFAULT '',
  `Continent` enum('Asia','Europe','North America','Africa','Oceania','Antarctica','South Americ
  PRIMARY KEY (`Code`),
  KEY `name_idx` (`Name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

第二个里程碑：从旧表导入数据到新表

```
insert into country_1(code,name,continent) select code,name,continent from country;
```

第三个里程碑：横向拆表

创建与导出表相同格式的新表1

```
create table country_1_p1 like country_1;
```

将数据的前100行插入到新表1中

```
insert into country_1_p1 select code,name,continent from country_1 order by code limit 100;
```

创建与新表1相同格式的新表2

```
create table country_1_p2 like country_1;
```

将100行之后的139行导入表2.

```
insert into country_1_p2 select code,name,continent from country_1 order by code limit 139 offse
```

1.8 参考文献

[1] http://blog.csdn.net/hguisu/article/details/7325124

[2] https://www.cnblogs.com/ivictor/p/5735580.html

[3] https://www.cnblogs.com/Aiapple/p/5792939.html

[4] http://heylinux.com/archives/1004.html 读写分离

[5] http://hualong.iteye.com/blog/2102798 Cobar逻辑层次图

[6] https://www.cnblogs.com/kevingrace/p/5662975.html MMM架构

[7] https://www.cnblogs.com/gomysql/p/3675429.html MHA架构

[8] http://blog.jobbole.com/70844/ MySQL在大型网站的应用架构演变

[9] https://www.cnblogs.com/dosmile/p/6681923.html MGR复制

[10] http://imysql.cn/tag/pxc PXC架构

[11] https://mp.weixin.qq.com/s/L1OsdKS6Vbw1pXIKbW3-NQ

赞0

如无特殊说明，文章均为本站原创，转载请注明出处

- 转载请注明来源：MySQL Replication 主从复制全方位解决方案
- 本文永久链接地址：https://www.nmtui.com/clsn/lx300.html

该文章由 惨绿少年 发布

惨绿少年Linux www.nmtui.com