

# 散尽浮华

安寻安放，不卑不亢；重剑无锋，大巧不工！

博客园 首页 新随笔 联系 管理 订阅 XML

随笔- 544 文章- 34 评论- 664

昵称: 散尽浮华

园龄: 3年4个月

粉丝: 2001

关注: 23

+加关注

## Mysql主从同步(1) - 概念和原理介绍 以及 主从/主主模式 部署记录

### Mysql复制概念

Mysql内建的复制功能是构建大型高性能应用程序的基础，将Mysql数据分布到多个系统上，这种分布机制是通过将Mysql某一台主机数据复制到其它主机（slaves）上，并重新执行一遍来实现的。复制过程中一个服务器充当主服务器，而一个或多个其它服务器充当从服务器。主服务器将更新写入二进制日志文件，并维护文件的一个索引以跟踪日志循环。这些日志可以记录发送到从服务器的更新。当一个从服务器连接主服务器时，它通知主服务器从服务器在日志中读取的最后一次成功更新的位置。从服务器接收从那时起发生的任何更新，然后封锁并等待主服务器通知新的更新。

#### Mysql支持哪些复制

- 基于语句的复制：在主服务器执行SQL语句，在从服务器执行同样语句。MySQL默认采用基于语句的复制，效率较高。一旦发现没法精确复制时，会自动退基于行的复制。
- 基于行的复制：把改变的内容复制过去，而不是把命令在从服务器上执行一遍。从mysql5.0开始支持
- 混合类型的复制：默认采用基于语句的复制，一旦发现基于语句的无法精确的复制时，就会采用基于行的复制。

#### Mysql复制解决的问题

- 数据分布 (Data distribution )
- 负载均衡(load balancing)
- 据备份(Backups) ， 保证数据安全
- 高可用性和容错行(High availability and failover)
- 实现读写分离，缓解数据库压力

### Mysql主从复制原理

master服务器将数据的改变都记录到二进制binlog日志中，只要master上的数据发生改变，则将其改变写入二进制日志；salve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/O Thread请求master二进制事件，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/O Thread和SQL Thread将进入睡眠状态，等待下一次被唤醒。

#### 需要理解：

- 从库会生成两个线程，一个I/O线程，一个SQL线程；
- 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；
- 主库会生成一个log dump线程，用来给从库I/O线程传binlog；
- 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；

#### 这里注意几点：

- 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）；
- 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；
- 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；
- 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）；
- master和slave两节点时间需同步；

#### Mysql复制流程图

2019年7月						
<	日	一	二	三	四	五
	30	1	2	3	4	5
	7	8	9	10	11	12
	14	15	16	17	18	19
	21	22	23	24	25	26
	28	29	30	31	1	2
	4	5	6	7	8	9

### 搜索

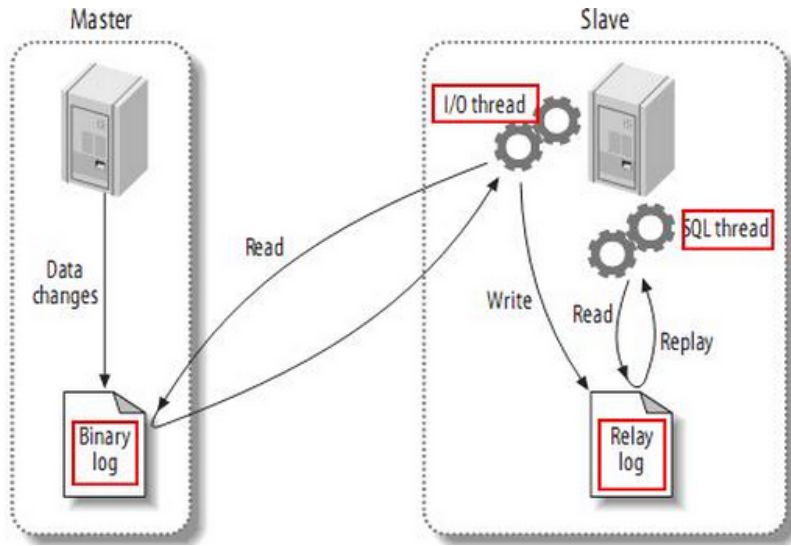
<input type="text"/>	<input type="button" value="找找看"/>
<input type="text"/>	<input type="button" value="谷歌搜索"/>

### 常用链接

我的随笔  
我的评论  
我的参与  
最新评论  
我的标签

### 随笔分类

Ansible(3)  
Apache(6)  
Ceph(5)  
ClusterShell(1)  
DNS(5)  
Docker(27)  
DRBD(2)  
Elasticsearch(6)  
Etcd  
Expect(2)  
Fabric(1)  
FastDFS(1)  
FTP(4)  
GlusterFS (5)  
Haproxy(6)  
IP SAN(1)  
Iptables(6)  
Jenkins(8)  
Jira and Confluence(7)  
Jumpserver(4)  
Kafka(2)  
LB+HA(23)  
LDAP(2)



如上图所示:

- Mysql复制过程的第一部分就是master记录二进制日志。在每个事务更新数据完成之前, master在二进制日志记录这些改变。MySQL将事务串行的写入二进制日志, 即使事务中的 语句都是交叉执行的。在事件写入二进制日志完成后, master通知存储引擎提交事务;
- 第二部分就是slave将master的binary log拷贝到它自己的中继日志。首先, slave开始一个工作线程(I/O线程)。I/O线程在master上打开一个普通的连接, 然后开始binlog dump process。Binlog dump process从master的二进制日志中读取事件, 如果已经跟上master, 它会睡眠并等待master产生新的事件。I/O线程将这些事件写入中继日志;
- SQL slave thread (SQL从线程) 处理该过程的最后一步。SQL线程从中继日志读取事件, 并重放其中的事件而更新slave的数据, 使其与master中的数据一致。只要该线程与 I/O线程保持一致, 中继日志通常会位于OS的缓存中, 所以中继日志的开销很小;
- 此外, 在master中也有一个工作线程: 和其它MySQL的连接一样, slave在master中打开一个连接也会使得master开始一个线程。复制过程有一个很重要的限制, 即复制在slave上是串行化的, 也就是说master上的并行更新操作不能在slave上并行操作。

#### Mysql复制模式

主从复制: 主库授权从库远程连接, 读取binlog日志并更新到本地数据库的过程; 主库写数据后, 从库会自动同步过来(从库跟着主库变);

主主复制: 主从相互授权连接, 读取对方binlog日志并更新到本地数据库的过程; 只要对方数据改变, 自己就跟着改变;

#### Mysql主从复制优点

在从服务器可以执行查询工作(即我们常说的读功能), 降低主服务器压力; (主库写, 从库读, 降压)

在从服务器进行备份, 避免备份期间影响主服务器服务; (确保数据安全)

当主服务器出现问题时, 可以切换到从服务器。(提升性能)

#### Mysql主从复制工作流程关键细节

1) MySQL支持单向、异步复制, 复制过程中一个服务器充当主服务器, 而一个或多个其它服务器充当从服务器。MySQL复制基于主服务器在二进制日志中跟踪所有对数据库的更改(更新、删除等等)。因此, 要进行复制, 必须在主服务器上启用二进制日志。每个从服务器从主服务器接收主服务器上已经记录到其二进制日志的保存的更新。当一个从服务器连接主服务器时, 它通知主服务器定位到从服务器在日志中读取的最后一次成功更新的位置。从服务器接收从那时起发生的任何更新, 并在本机上执行相同的更新。然后封锁并等待主服务器通知新的更新。从服务器执行备份不会干扰主服务器, 在备份过程中主服务器可以继续处理更新。

2) MySQL使用3个线程来执行复制功能, 其中两个线程(Sql线程和IO线程)在从服务器, 另外一个线程(IO线程)在主服务器。当发出START SLAVE时, 从服务器创建一个I/O线程, 以连接主服务器并让它发送记录在其二进制日志中的语句。主服务器创建一个线程将二进制日志中的内容发送到从服务器。该线程可以即为主服务器上SHOW PROCESSLIST的输出中的Binlog Dump线程。从服务器I/O线程读取主服务器Binlog Dump线程发送的内容并将该数据拷贝到从服务器数据目录中的本地文件中, 即中继日志。第3个线程是SQL线程, 由从服务器创建, 用于读取中继日志并执行日志中包含的更新。在从服务器上, 读取和执行更新语句被分成两个独立的任务。当从服务器启动时, 其I/O线程可以很快地从主服务器索取所有二进制日志内容, 即使SQL线程执行更新的远远滞后。

### mysql 主从复制注意事项小

## 结

#### Mysql主从数据完成同步的过程

- 1) 在Slave 服务器上执行start slave命令开启主从复制开关, 开始进行主从复制。
- 2) 此时, Slave服务器的IO线程会通过主从复制已经授权的复制用户权限请求连接master服务器, 并请求从执行binlog日志文件的指定位置(日志文件名和位置就是在配置主从复制服务时执行change master命令指定的)之

LVM(3)  
LVS(5)  
Maven/Nexus(1)  
Memcached(4)  
MongoDB(11)  
MooseFS(2)  
MQ-消息队列(5)  
Mysql(65)  
NFS(1)  
Nginx(51)  
PHP(10)  
Puppet(3)  
Python(12)  
Redis(17)  
Rsync(8)  
Saltstack(4)  
Samba(3)  
Shell(34)  
Squid(4)  
SSH(7)  
Supervisor/Monit(3)  
Tomcat(11)  
Ubuntu(9)  
Varnish(1)  
VPN(6)  
Zookeeper(2)  
安全性能(30)  
版本控制(31)  
常规运维(86)  
监控系统(43)  
日志分析(8)  
虚拟化(30)  
邮件服务(3)

## 随笔档案

2019年7月 (4)  
2019年6月 (1)  
2019年4月 (2)  
2019年3月 (6)  
2019年2月 (6)  
2019年1月 (6)  
2018年12月 (9)  
2018年11月 (7)  
2018年10月 (10)  
2018年9月 (9)  
2018年8月 (12)  
2018年7月 (11)  
2018年6月 (2)  
2018年5月 (12)  
2018年4月 (11)  
2018年3月 (8)  
2018年2月 (12)  
2018年1月 (21)  
2017年12月 (17)  
2017年11月 (12)  
2017年10月 (6)  
2017年9月 (10)  
2017年8月 (7)  
2017年7月 (5)  
2017年6月 (6)  
2017年5月 (8)  
2017年4月 (10)  
2017年3月 (11)  
2017年2月 (15)  
2017年1月 (36)  
2016年12月 (41)  
2016年11月 (34)  
2016年10月 (30)

后开始发送binlog日志内容

3) Master服务器接收到来自Slave服务器的IO线程的请求后, 其上负责复制的IO线程会根据Slave服务器的IO线程请求的信息分批读取指定binlog日志文件指定位置之后的binlog日志信息, 然后返回给Slave端的IO线程。返回的信息中除了binlog日志内容外, 还有在Master服务器端记录的IO线程。返回的信息中除了binlog中的下一个指定更新位置。

4) 当Slave服务器的IO线程获取到Master服务器上IO线程发送的日志内容、日志文件及位置点后, 会将binlog日志内容依次写到Slave端自身的Relay Log (即中继日志) 文件 (Mysql-relay-bin.xxx) 的最末端, 并将新的binlog文件名和位置记录到master-info文件中, 以便下一次读取master端新binlog日志时能告诉Master服务器从新binlog日志的指定文件及位置开始读取新的binlog日志内容

5) Slave服务器端的SQL线程会实时检测本地Relay Log 中IO线程新增的日志内容, 然后及时把Relay LOG 文件中的内容解析成sql语句, 并在自身Slave服务器上按解析SQL语句的位置顺序执行应用这样sql语句, 并在relay-log.info中记录当前应用中继日志的文件名和位置点。

#### 主从复制条件

- 开启Binlog功能
- 主库要建立账号
- 从库要配置master.info (CHANGE MASTER to...相当于配置密码文件和Master的相关信息)
- start slave 开启复制功能

#### 主从复制时需要理解

- 3个线程, 主库IO, 从库IO和SQL及作用
- master.info (从库) 作用
- relay-log 作用
- 异步复制
- binlog作用 (如果需要级联需要开启Binlog)

#### 主从复制时注意事项

- 主从复制是异步逻辑的SQL语句级的复制
- 复制时, 主库有一个I/O线程, 从库有两个线程, I/O和SQL线程
- 实现主从复制的必要条件是主库要开启记录binlog功能
- 作为复制的所有Mysql节点的server-id都不能相同
- binlog文件只记录对数据库有更改的SQL语句 (来自主库内容的变更), 不记录任何查询 (select, show) 语句

#### 彻底解除主从复制关系

- stop slave;
- reset slave; 或直接删除master.info和relay-log.info这两个文件;
- 修改my.cnf删除主从相关配置参数。

#### 让slave不随MySQL自动启动

- 1 | 修改my.cnf 在[mysqld]中增加"skip-slave-start"选项。

主从复制实现后, 使用mysqldump备份数据时, 注意按照下面方式

```
1 | # mysqldump --master-data --single-transaction --user=username --password=password dbna
2 | 这样就可以保留 file 和 position 的信息, 在新搭建一个slave时, 还原完数据库, file 和 position 的信
```

需要限定同步哪些数据库, 有3个思路

- 在执行grant授权的时候就限定数据库;
- 在主服务器上限定binlog\_do\_db = 数据库名;
- 主服务器上不限定数据库, 在从服务器上限定replicate-do-db = 数据库名;

如果想实现"主-从(主)-从" 这样的链式结构, 需要设置

- ```
1 | log-slave-updates          #只有加上它, 从前一台机器上同步过来的数据才能同步到下一台机器。
2 | log-bin=/opt/mysql/binlogs/bin-log          #二进制日志也必须开启
3 | log-bin-index=/opt/mysql/binlogs/bin-log.index
4 | expire_logs_days=14      # 还可以设置一个log保存周期
```

## Mysql主从同步时过滤部分库或表的设置

### 置

#### Mysql复制过滤

让从节点仅仅复制指定的数据库, 或指定数据库的指定数据表。主服务器有10个数据库, 而从节点只需要同步其中的一两个数据库。这个时候就需要复制过滤。复制过滤器可以在主节点中实现, 也可以在从节点中实现。Mysql主从同步部分数据有两个思路: master只发送需要的; Slave只接收想要的。

2016年9月 (35)  
2016年8月 (35)  
2016年7月 (37)  
2016年6月 (37)  
2016年3月 (3)

## Linux加油站

Ansible中文权威指南  
AWK使用手册  
Centos的epel源下载  
Ceph开源社区  
Codis教程  
Django基础教程  
Docker基础学习  
Docker镜像-hub.docker  
Elasticsearch 中文社区  
Git基础教程  
Go语言学习速查手册  
Go语言中文网  
HTTP Status Codes  
IT牛人博客-运维学习  
Kubernetes TLS bootstrapping 那点事  
Kubernetes 集群部署参考  
Kubernetes 中文社区  
Linux命令大全  
Linux运维日志  
Linux专注监控的博客  
Mysql源码下载  
Nginx官方配置(Modules reference)  
Open-falcon社区文档  
OpenResty--Nginx和LuaJIT的Web平台  
Prometheus 操作指南  
Prometheus 中文手册  
Puppet安装版本下载  
Python3-cookbook  
Python学习- 推荐网站  
Python学习资料-中文电子书分享  
Python自动化运维-案例源码  
Python自动化运维之路  
Rancher 部署文档  
Redis命令参考  
RUNOOB.COM - 编程学习  
Shell传递的参数说明  
Squid中文权威指南  
Supervisor教程  
Swarm管理-Linux运维日志  
Tengine参考文档  
Zabbix完整监控 - 配置教程  
阿里开源镜像  
博客在线-LVS  
监控ElasticSearch性能指标  
精通Python自动化脚本  
每天一个linux命令  
鸟哥的Linux私房菜  
网易开源镜像  
网站速度测试-17ce  
我的攻防安全指南  
以优雅姿态监控 Kubernetes  
优质博客-运维  
域名信息查询平台  
运维派  
值得学习的牛人博客

## 最新评论

1. Re:CentOS下Proftpd环境部署并使用  
虚拟用户登录 - 运维笔记



## master主节点

在主节点的二进制日志中仅记录与指定数据库（数据表）相关的事件日志，但是主节点的二进制日志不完整，没有记录所有对主节点的修改操作。如果要使用该方式，则在主节点的配置文件中添加如下参数：

```
1 binlog_do_db="***,***,***";      #数据库白名单列表，二进制日志记录的数据库（多数据库用逗号隔开或重
2 binlog_ignore_db="***,***,***";  #数据库黑名单列表，二进制日志中忽略的数据库（多数据库用逗号隔开
```

## slave从节点

从服务器SQL Thread在Replay中继日志中的事件时仅读取于特定数据库相关的事件，并应用于本地。（但是浪费I/O,浪费带宽）推荐使用从节点复制过滤相关设置项：

```
1 replicate_do_db = "webdb";      #复制库的白名单。设定需要复制的数据库（多数据库使用逗号隔开或重
2 replicate_ignore_db = "mysql";  #复制库的黑名单。设定需要忽略的复制数据库（多数据库使用逗号隔
3 replicate_do_table = "webdb.user"; #复制表的白名单。设定需要复制的表（多数据库使用逗号隔开或重复设
4 replicate_ignore_table = "webdb.uw"; #复制表的黑名单。设定需要忽略的复制的表（多数据库使用逗号隔开或
5
6 replicate-wild-do-table          #同replication-do-table功能一样，但是可以通配符。更高级别的
7 replicate-wild-ignore-table      #同replication-ignore-table功能一样，但是可以加通配符。
```

当在主库存在的库而从库不存在的库同步时，会出现sql错误，这时候可以排除或者从库手动导入主库数据库；

从库可以使用通配符“库名.%”方式过滤主从同步时某个库的设置

```
1 replicate-wild-do-table=webdb.%      #只复制webdb库下的所有表
2 replicate-wild-ignore-table=mysql.%  #忽略mysql库下的所有表
```

特别注意：生产库上一般不建议设置过滤规则，如果非要设置，强烈建议从库使用通配符方式过滤某个库

```
1 replicate-wild-do-table= "库名.%"
2 replicate-wild-ignore-table= "库名.%"
```

而不建议从库使用DB方式过滤某个库：

```
1 replicate_do_db = "库名"
2 replicate_ignore_db = "库名"
```

## mysql主从/主主同步环境部署记录

### 1) 环境描述

```
1 mysql的安装可以参考：http://www.cnblogs.com/kevingrace/p/6109679.html
2 Centos6.8版本
3 master: 182.148.15.238
4 slave: 182.148.15.237
5
6 注意下面几点：
7 1) 要保证同步服务期间之间的网络联通。即能相互ping通，能使用对方授权信息连接到对方数据库（防火墙开放3306
8 2) 关闭selinux。
9 3) 同步前，双方数据库中需要同步的数据要保持一致。这样，同步环境实现后，再次更新的数据就会如期同步了。
```

### 2) 主从复制实现过程记录

```
1 为了测试效果，先在master机器上创建测试库
2 mysql> CREATE DATABASE huanqiu CHARACTER SET utf8 COLLATE utf8_general_ci;
3 Query OK, 1 row affected (0.00 sec)
4
5 mysql> use huanqiu;
6 Database changed
7 mysql> create table if not exists haha (id int(10) PRIMARY KEY AUTO_INCREMENT,name va
8 Query OK, 0 rows affected (0.02 sec)
9
10 mysql> insert into huanqiu.haha values(1,"wangshibo"),(2,"guohuihui");
11 Query OK, 2 rows affected (0.00 sec)
```

向博主学习

--少林功夫好

2. Re:Linux下FTP虚拟账号环境部署记录 (vsftpd)

博主确实博学。

--少林功夫好

3. Re:Linux下PAM模块学习总结

牛逼好文

--少林功夫好

4. Re:Docker容器基础介绍

杨过大侠你好！你原文的句子“不要在容器中存储数据 - 容器可能被停止，销毁，或替换。一个运行在容器中的程序版本1.0，应该很容易被1.1的版本替换且不影响或损失数据。有鉴于此，如.....

--darenzhoudba

5. Re:Centos下安装破解confluence6.3的操作记录

@nq\_linux在配置界面里更改默认语言为中文...

--散尽浮华

6. Re:Centos下安装破解confluence6.3的操作记录

楼主，你好。按照你的教程安装成功了，非常感谢。但是有点奇怪的是，我在最后一步执行完之后，页面就变成英文了（最开始的一步里面，我已经选择了中文，并且一路都是中文）。请问能给点指点吗？...

--nq\_linux

7. Re:linux下安装php的imagick扩展模块（附php升级脚本）

@与威尔已更新~...

--散尽浮华

8. Re:linux下安装php的imagick扩展模块（附php升级脚本）

第一个百度网盘下载链接。。。。。。。。

--与威尔

9. Re:linux下监控某个目录是否被更改非常好

--renzhehongyi

10. Re:Centos下安装破解confluence6.3的操作记录

楼主百度云都失效了能在提供下嘛

--小小飞侠

## 阅读排行榜

1. Git忽略提交规则 - .gitignore配置运维总结(380442)
2. ELK实时日志分析平台环境部署--完整记录(159689)
3. 完整部署CentOS7.2+OpenStack+kv m 云平台环境（1）--基础环境搭建(125819)
4. mysql数据库误删除后的数据恢复操作说明(114606)
5. Linux终端复用神器-Tmux使用梳理(11384)
6. Mysql之binlog日志说明及利用binlog日志恢复数据操作记录(67212)
7. mysql表名忽略大小写问题记录(65962)
8. MySQL两种存储引擎：MyISAM和InnoDB 简单总结(63757)
9. Gitlab利用Webhook实现Push代码后的jenkins自动构建(62580)

10. 执行git push出现"Everything up-to-date"(62327)

## 评论排行榜

1. 完整部署CentOS7.2+OpenStack+kv m 云平台环境 (1) --基础环境搭建(119)
2. ELK实时日志分析平台环境部署--完整记录(24)
3. [原创]CI持续集成系统环境--Gitlab+G errit+Jenkins完整对接(20)
4. kvm虚拟化管理平台WebVirtMgr部署-完整记录(1)(18)
5. Centos下安装破解Jira7的操作记录(18)

## 推荐排行榜

1. Git忽略提交规则 - .gitignore配置运维总结(34)
2. ELK实时日志分析平台环境部署--完整记录(29)
3. 完整部署CentOS7.2+OpenStack+kv m 云平台环境 (1) --基础环境搭建(20)
4. Maven私服Nexus3.x环境构建操作记录(12)
5. SVN和Git对比梳理(12)
6. Mysql之binlog日志说明及利用binlog日志恢复数据操作记录(11)
7. 运维中的日志切割操作梳理 (Logrotate/python/shell脚本实现) (10)
8. mysql主从同步(3)-percona-toolkit工具 (数据一致性监测、延迟监控) 使用梳理(9)
9. 简单对比git pull和git pull --rebase的使用(9)
10. Redis哨兵模式 (sentinel) 学习总结及部署记录 (主从复制、读写分离、主从切换) (8)

```
12 Records: 2 Duplicates: 0 Warnings: 0
13
14 mysql> select * from huanqiu.haha;
15 +-----+
16 | id | name |
17 +-----+
18 | 1 | wangshibo |
19 | 2 | guohuihui |
20 +-----+
21 2 rows in set (0.00 sec)
22
23 -----
24 温馨提示:
25 修改库或表的字符集
26 mysql> alter database huanqiu default character set utf8; //修改huanqiu库的字符集
27 mysql> alter table huanqiu.haha default character set utf8; //修改huanqiu.haha表的字符
28
29 添加主键
30 mysql> Alter table huanqiu.haha add primary key(id); //将huanqiu.haha表的id添加主键
31 mysql> Alter table huanqiu.haha change id id int(10) not null auto_increment; //自增
32
33 删除主键时要先删除自增长,再删除主键
34 mysql> Alter table huanqiu.haha change id id int(10); //删除自增长
35 mysql> Alter table huanqiu.haha drop primary key; //删除主键
36 -----
37
38 下面是master数据库上的操作:
39 1) 设置master数据库的my.cnf文件 (在[mysqld]配置区域添加下面内容)
40 [root@master ~]# vim /usr/local/mysql/my.cnf
41 .....
42 server-id=1 #数据库唯一ID, 主从的标识号绝对不能重复。
43 log-bin=mysql-bin #开启bin-log, 并指定文件目录和文件名前缀
44 binlog-do-db=huanqiu #需要同步的数据库。如果是多个同步库, 就以此格式另写几行即可。如果不指明对某
45 binlog-ignore-db=mysql #不同步mysql系统数据库。如果是多个不同步库, 就以此格式另写几行; 也可以在
46 sync_binlog = 1 #确保binlog日志写入后与硬盘同步
47 binlog_checksum = none #跳过现有的采用checksum的事件, mysql5.6.5以后的版本中binlog_checks
48 binlog_format = mixed #bin-log日志文件格式, 设置为MIXED可以防止主键重复。
49
50 -----
51 温馨提示: 在主服务器上最重要的二进制日志设置是sync_binlog, 这使得mysql在每次提交事务的时候把二进制
52 sync_binlog这个参数是对于MySQL系统来说是至关重要的, 他不仅影响到Binlog对MySQL所带来的性能损耗, 而
53 sync_binlog=0, 当事务提交之后, MySQL不做fsync之类的磁盘同步指令刷新binlog_cache中的信息到磁盘,
54 sync_binlog=n, 当每进行n次事务提交之后, MySQL将进行一次fsync之类的磁盘同步指令来将binlog_cache
55
56 在MySQL中系统默认的设置是sync_binlog=0, 也就是不做任何强制性的磁盘刷新指令, 这时候的性能是最好的,
57
58 从以往经验和相关测试来看, 对于高并发事务的系统来说, "sync_binlog"设置为0和设置为1的系统写入性能差距
59 -----
60
61 2) 导出master数据库多余slave数据库中的数据, 然后导入到slave数据库中。保证双方在同步环境实现前的数据
62 导出数据库之前先锁定数据库
63 mysql> flush tables with read lock; #数据库只读锁定命令, 防止导出数据库的时候有数据写入。
64
65 导出master数据库中多余的huanqiu库(master数据库的root用户登陆密码: 123456)
66 [root@master ~]# mysqldump -uroot huanqiu -p123456 >/opt/huanqiu.sql
67 [root@master ~]# rsync -e "ssh -p22" -avpgolr /opt/huanqiu.sql 182.148.15.237:/opt
68
69 3) 设置数据同步权限
70 mysql> grant replication slave,replication client on *.* to slave@'182.148.15.237'
```

```

71 Query OK, 0 rows affected (0.02 sec)
72 mysql> flush privileges;
73 Query OK, 0 rows affected (0.00 sec)
74
75 -----
76 温馨提示:
77 权限查看方式
78 mysql> show grants;
79 mysql> show grants for slave@'182.148.1115.237';
80 -----
81
82 4) 查看主服务器master状态(注意File与Position项, 从服务器需要这两项参数)
83 mysql> show master status;
84 +-----+-----+-----+-----+-----+
85 | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
86 +-----+-----+-----+-----+-----+
87 | mysql-bin.000007 |      120 | huanqiu      | mysql             |
88 +-----+-----+-----+-----+-----+
89 1 row in set (0.00 sec)
90
91
92 下面是slave数据库上的操作:
93 1) 设置slave数据库的my.cnf配置文件
94 [root@master ~]# vim /usr/local/mysql/my.cnf
95 .....
96 server-id=2    #设置从服务器id, 必须于主服务器不同
97 log-bin=mysql-bin    #启动MySQL二进制日志系统
98 replicate-do-db=huanqiu    #需要同步的数据库名。如果不指明同步哪些库, 就去掉这行, 表示所有库的同步
99 replicate-ignore-db=mysql    #不同步mysql系统数据库
100 slave-skip-errors = all    #跳过所有的错误错误, 继续执行复制操作
101
102 -----
103 温馨提示:
104 当只针对某些库的某张表进行同步时, 如下, 只同步huanqiu库的haha表和huanpc库的heihei表:
105 replicate-do-db = huanqiu
106 replicate-wild-do-table = huanqiu.haha    //当只同步几个或少数表时, 可以这样设置。注意这要
107 replicate-do-db = huanpc
108 replicate-wild-do-table = huanpc.heihei    //如果同步的库的表比较多时, 就不能这样——指定了
109 -----
110
111 2) 在slave数据库中导入从master传过来的数据。
112 mysql> CREATE DATABASE huanqiu CHARACTER SET utf8 COLLATE utf8_general_ci;    #先创建一
113 mysql> use huanqiu;
114 mysql> source /opt/huanqiu.sql;    #导入master中多余的数据。
115 .....
116
117 3) 配置主从同步指令
118 mysql> stop slave;    #执行同步前, 要先关闭slave
119 mysql> change master to master_host='182.148.15.238',master_user='slave',master_pass
120
121 mysql> start slave;
122 mysql> show slave status \G;
123 .....
124 ***** 1. row *****
125         Slave_IO_State: Waiting for master to send event
126         Master_Host: 182.148.15.238
127         Master_User: slave
128         Master_Port: 3306
129         Connect_Retry: 60

```

```

130      Master_Log_File: mysql-bin.000007
131      Read_Master_Log_Pos: 120
132      Relay_Log_File: mysql-relay-bin.000002
133      Relay_Log_Pos: 279
134      Relay_Master_Log_File: mysql-bin.000007
135      Slave_IO_Running: Yes
136      Slave_SQL_Running: Yes
137      Replicate_Do_DB: huanqiu
138      Replicate_Ignore_DB: mysql
139      .....
140      Seconds_Behind_Master: 0
141

```

如上，当IO和SQL线程的状态均为Yes，则表示主从已实现同步了！

查看slave数据库中的数据情况

```

145 mysql> show databases;
146 +-----+
147 | Database          |
148 +-----+
149 | information_schema |
150 | huanqiu            |
151 | mysql              |
152 | performance_schema |
153 | test               |
154 +-----+
155 5 rows in set (0.00 sec)

```

```

157 mysql> select * from huanqiu.haha;
158 +-----+
159 | id | name      |
160 +-----+
161 | 1  | wangshibo |
162 | 2  | guohuihui |
163 +-----+
164 2 rows in set (0.00 sec)

```

下面测试下Mysql主从同步的效果

现在主数据库上写入新数据

```

168 mysql> unlock tables;      #解锁，否则新数据无法写入
169 mysql> insert into huanqiu.haha values(100,"anhui");
170 Query OK, 1 row affected (0.00 sec)

```

然后在slave数据库上查看，发现master上新写入的数据已经同步过来了

```

173 mysql> select * from huanqiu.haha;
174 +-----+
175 | id | name      |
176 +-----+
177 | 1  | wangshibo |
178 | 2  | guohuihui |
179 | 100 | anhui     |
180 +-----+
181 3 rows in set (0.00 sec)

```

至此，主从同步环境已经实现！

Mysql主从环境部署一段时间后，发现主从不同步时，如何进行数据同步至一致？

有以下两种做法：

1) 参考：[mysql主从同步\(2\)-问题梳理](#) 中的第（4）步的第二种方法

2) 参考: [mysql主从同步\(3\)-percona-toolkit工具 \(数据一致性监测、延迟监控\) 使用梳理](#)

**温馨提示:** 在实际业务场景中, mysql主从同步时最好别过滤库, 即最好进行基于整个数据库的同步配置。如果业务库比较多的情况下, 可使用mysql多实例方式进行同步, 一个业务库对应一个mysql实例, 每个mysql实例做基于它的整个数据库的同步配置。使用过滤库或过滤表的方式进行主从同步配置, 后续会带来一些比较麻烦的坑。

```

1  从库同步主库的命令调整为:
2  "change master to master_host = '主数据库ip', master_port = 主数据库mysql端口, master_use
3
4  针对上面的主从配置, 调整为基于整个数据库的主从同步配置, 调整后的配置:
5
6  主数据库的my.cnf配置:
7  server-id=1
8  log-bin=mysql-bin
9  sync_binlog = 1
10 binlog_checksum = none
11 binlog_format = mixed
12
13 从数据库的my.cnf配置:
14 server-id=2
15 log-bin=mysql-bin
16 slave-skip-errors = all
17
18 从库同步主库的命令如下 (即不需要跟master_log_file 和 master_log_pos=120)
19 mysql> change master to master_host = '182.148.15.238', master_port = 3306, master_use

```

### 3) 主主复制实现过程记录

```

1  根据上面的主从环境部署, master和slave已经实现同步, 即在master上写入新数据, 自动同步到slave。而从库
2  下面就说下Mysql主主复制环境, 在slave上更新数据时, master也能自动同步过来。
3  -----
4  温馨提示:
5  在做主主同步前, 提醒下需要特别注意的一个问题:
6  主主复制和主从复制有一些区别, 因为多主中都可以对服务器有写权限, 所以设计到自增长重复问题, 例如:
7  出现的问题 (多主自增长ID重复)
8  1) 首先在A和B两个库上创建test表结构;
9  2) 停掉A, 在B上对数据表test (存在自增长属性的ID字段) 执行插入操作, 返回插入ID为1;
10 3) 然后停掉B, 在A上对数据表test (存在自增长属性的ID字段) 执行插入操作, 返回的插入ID也是1;
11 4) 然后 同时启动A, B, 就会出现主键ID重复
12
13 解决方法:
14 只要保证两台服务器上的数据库里插入的自增长数据不同就可以了
15 如: A插入奇数ID, B插入偶数ID, 当然如果服务器多的话, 还可以自定义算法, 只要不同就可以了
16 在下面例子中, 在两台主主服务器上加入参数, 以实现奇偶插入!
17 记住: 在做主主同步时需要设置自增长的两个相关配置, 如下:
18 auto_increment_offset      表示自增长字段从那个数开始, 取值范围是1 .. 65535。这个就是序号。如:
19 auto_increment_increment    表示自增长字段每次递增的量, 其默认值是1, 取值范围是1 .. 65535。如:
20
21 在主主同步配置时, 需要将两台服务器的:
22 auto_increment_increment    增长量都配置为2
23 auto_increment_offset       分别配置为1和2。这是序号, 第一台从1开始, 第二台就是2, 以此类推! 这样
24 才可以避免两台服务器同时做更新时自增长字段的值之间发生冲突。(针对的是有自增长属性的字段)
25 -----
26
27 现在记录下主主同步的操作过程:
28 1) 在master上的my.cnf配置:

```



```

29 [root@master ~]# vim /usr/local/mysql/my.cnf
30 server-id = 1
31 log-bin = mysql-bin
32 binlog-ignore-db = mysql,information_schema
33 sync_binlog = 1
34 binlog_checksum = none
35 binlog_format = mixed
36 auto-increment-increment = 2
37 auto-increment-offset = 1
38 slave-skip-errors = all
39
40 [root@master ~]# /etc/init.d/mysql restart
41 Shutting down MySQL. SUCCESS!
42 Starting MySQL.. SUCCESS!
43
44 数据同步授权 (iptables防火墙开启3306端口, 要确保对方机器能使用下面权限连接到本机mysql)
45 mysql> grant replication slave,replication client on *.* to slave@'182.148.15.237' id
46 mysql> flush privileges;
47
48 最好将库锁住, 仅仅允许读, 以保证数据一致性; 待主主同步环境部署后再解锁; 锁住后, 就不能往表里写数据, 但
49 mysql> FLUSH TABLES WITH READ LOCK; //注意该参数设置后, 如果自己同步对方数据, 同步前一定要
50 Query OK, 0 rows affected (0.00 sec)
51
52 mysql> show master status;
53 +-----+-----+-----+-----+-----+
54 | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
55 +-----+-----+-----+-----+-----+
56 | mysql-bin.000001 | 1970 | | | |
57 +-----+-----+-----+-----+-----+
58 1 row in set (0.00 sec)
59
60 2) slave数据库上
61 [root@slave ~]# vim /usr/local/mysql/my.cnf
62 server-id = 2
63 log-bin = mysql-bin
64 binlog-ignore-db = mysql,information_schema
65 sync_binlog = 1
66 binlog_checksum = none
67 binlog_format = mixed
68 auto-increment-increment = 2
69 auto-increment-offset = 2
70 slave-skip-errors = all
71
72 [root@slave ~]# /etc/init.d/mysql restart
73 Shutting down MySQL. SUCCESS!
74 Starting MySQL.. SUCCESS!
75
76 数据同步授权 (iptables防火墙开启3306端口, 要确保对方机器能使用下面权限连接到本机mysql)
77 同理, slave也要授权给master机器远程同步数据的权限
78 mysql> grant replication slave ,replication client on *.* to slave@'182.148.15.238' i
79 mysql> flush privileges;
80
81 mysql> FLUSH TABLES WITH READ LOCK;
82 mysql> show master status;
83 +-----+-----+-----+-----+-----+
84 | File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
85 +-----+-----+-----+-----+-----+
86 | mysql-bin.000001 | 4136 | | | |
87 +-----+-----+-----+-----+-----+

```

```

88 1 row in set (0.00 sec)
89
90 3) 执行主主同步操作
91 先在slave数据库上做同步master的设置。(确保slave上要同步的数据, 提前在master上存在。最好双方数据保
92 mysql> unlock tables;      //先解锁, 将对方数据同步到自己的数据库中
93 mysql> slave stop;
94 mysql> change master to master_host='182.148.15.238',master_user='slave',master_passw
95
96 mysql> start slave;
97 mysql> show slave status \G;
98 ***** 1. row *****
99      Slave_IO_State: Waiting for master to send event
100      Master_Host: 182.148.15.238
101      Master_User: slave
102      Master_Port: 3306
103      Connect_Retry: 60
104      Master_Log_File: mysql-bin.000001
105      Read_Master_Log_Pos: 1970
106      Relay_Log_File: mysql-relay-bin.000003
107      Relay_Log_Pos: 750
108      Relay_Master_Log_File: mysql-bin.000001
109      Slave_IO_Running: Yes
110      Slave_SQL_Running: Yes
111      .....

```

这样就实现了slave -> master的同步环境。

```

116 再在master数据库上做同步slave的设置。(确保slave上要同步的数据, 提前在master上存在。最好双方数据保
117 mysql> unlock tables;
118 mysql> slave stop;
119 mysql> change master to master_host='182.148.15.237',master_user='slave',master_passw
120
121 mysql> start slave;
122 mysql> show slave status \G;
123 ***** 1. row *****
124      Slave_IO_State: Waiting for master to send event
125      Master_Host: 182.148.15.237
126      Master_User: slave
127      Master_Port: 3306
128      Connect_Retry: 60
129      Master_Log_File: mysql-bin.000001
130      Read_Master_Log_Pos: 4136
131      Relay_Log_File: mysql-relay-bin.000003
132      Relay_Log_Pos: 750
133      Relay_Master_Log_File: mysql-bin.000001
134      Slave_IO_Running: Yes
135      Slave_SQL_Running: Yes
136      .....

```

这样就实现了master -> slave的同步环境。至此, 主主双向同步环境已经实现!

测试下Mysql主主同步的效果

在master上写入新数据

```

142 mysql> select * from huanqiu.haha;
143 +-----+-----+
144 | id | name |
145 +-----+-----+
146 | 1 | wangshibo |

```

```
147 | 2 | guohuihui |
148 | 100 | anhui |
149 +-----+-----+
150 3 rows in set (0.00 sec)
151
152 mysql> insert into huanqiu.haha values(15,"guoconggong");
153
154 在slave数据库中查看，发现master新写入的数据已经同步过来了
155 mysql> select * from huanqiu.haha;
156 +-----+-----+
157 | id | name |
158 +-----+-----+
159 | 1 | wangshibo |
160 | 2 | guohuihui |
161 | 15 | guoconggong |
162 | 100 | anhui |
163 +-----+-----+
164 3 rows in set (0.00 sec)
165
166 在slave上删除数据
167 mysql> delete from huanqiu.haha where id=100;
168
169 在master数据库中查看
170 mysql> select * from huanqiu.haha;
171 +-----+-----+
172 | id | name |
173 +-----+-----+
174 | 1 | wangshibo |
175 | 2 | guohuihui |
176 | 15 | guoconggong |
177 +-----+-----+
178 3 rows in set (0.00 sec)
```




\*\*\*\*\*当你发现自己的才华撑不起野心时，就请安静下来学习吧\*\*\*\*\*

分类: [Mysql](#)

好文要顶

关注我

收藏该文



[散尽浮华](#)  
[关注 - 23](#)  
[粉丝 - 2001](#)  
[+加关注](#)

5

0

« 上一篇: [分布式监控系统Zabbix--完整安装记录 \(7\) -使用percona监控MySQL](#)  
» 下一篇: [linux下EOF写法梳理](#)

posted @ 2017-01-06 15:51 [散尽浮华](#) 阅读(15353) 评论(2) [编辑](#) [收藏](#)

评论

#1楼 2017-11-19 20:54 | 蜗Amazon牛

<https://www.cnblogs.com/hellotracy/articles/5183057.html>

支持(0) 反对(0)

#2楼 2018-01-27 10:55 | 东北小狐狸

写的真好，拜读了。感谢

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】超50万C++/C#源码：大型实时仿真组态图形源码

【前端】SpreadJS表格控件，可嵌入系统开发的在线Excel

【推荐】程序员问答平台，解决您开发中遇到的技术难题



#### 相关博文：

- [mysql数据库主从同步](#)
- [mysql 主从同步](#)
- [Mariadb--主从同步](#)
- [mysql主从同步](#)
- [mysql主从复制](#)

#### 最新新闻：

- [2019年Q2中国智能手机成绩单：OV、小米、苹果等均下滑](#)
  - [全球首款 OPPO瀑布屏真机横空出世：弯曲度高达88度](#)
  - [7月29日是复读机发明28周年纪念日](#)
  - [住友开发出在轮胎内发电的方法](#)
  - [加州理工微机器人实现体内实时成像与控制，或将奋战抗癌前线](#)
- » [更多新闻...](#)