常用命令:

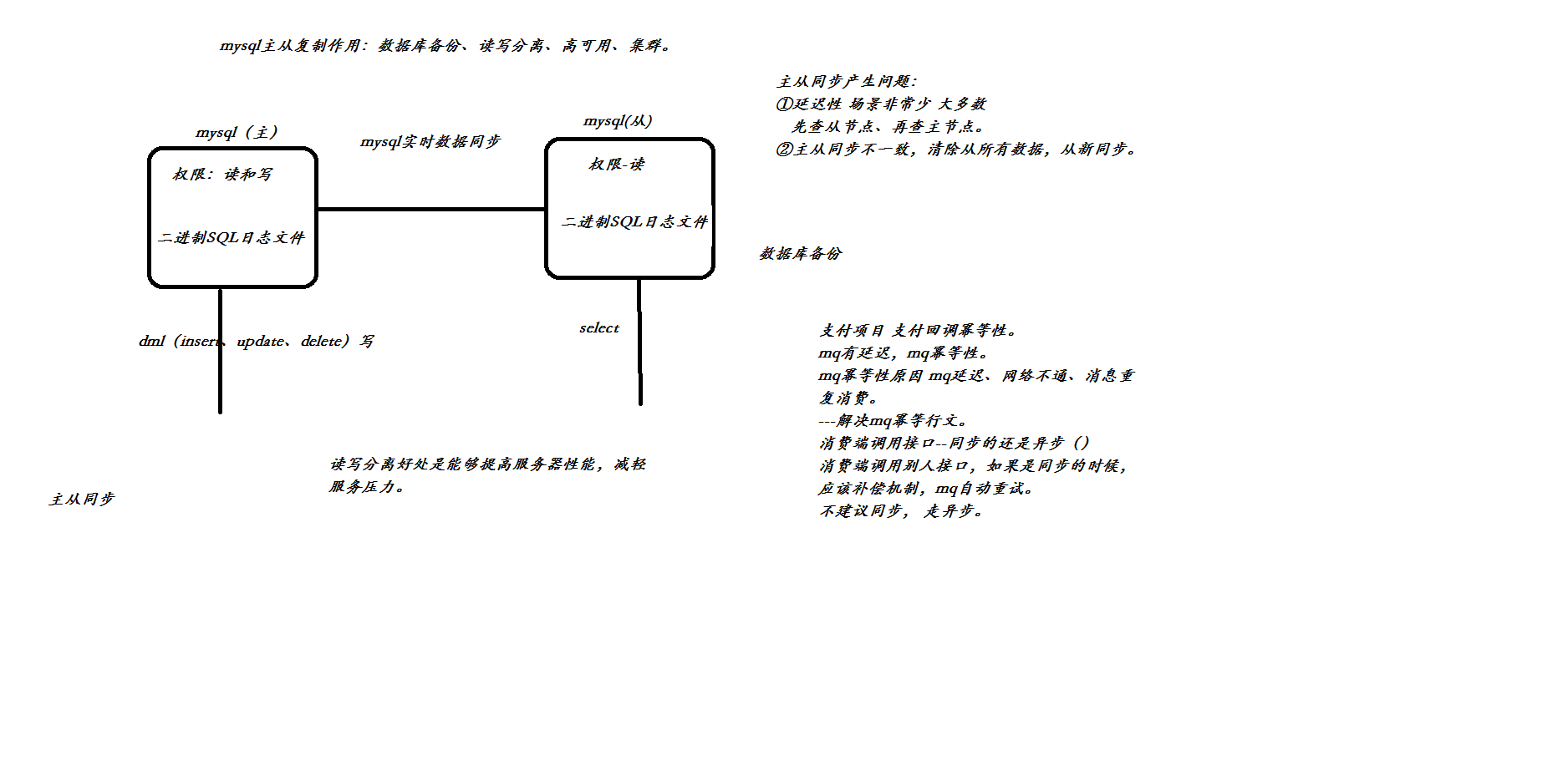
service mysqld start

service iptables stop

mysql –u root

# 主从复制





## 概念

影响MySQL-A数据库的操作，在数据库执行后，都会写入本地的日志系统A中。

 假设，实时的将变化了的日志系统中的数据库事件操作，在MYSQL-A的3306端口，通过网络发给MYSQL-B。

 MYSQL-B收到后，写入本地日志系统B，然后一条条的将数据库事件在数据库中完成。

 那么，MYSQL-A的变化，MYSQL-B也会变化，这样就是所谓的MYSQL的复制，即MYSQL replication。

 在上面的模型中，MYSQL-A就是主服务器，即master，MYSQL-B就是从服务器，即slave。

 日志系统A，其实它是MYSQL的日志类型中的二进制日志，也就是专门用来保存修改数据库表的所有动作，即bin log。【注意MYSQL会在执行语句之后，释放锁之前，写入二进制日志，确保事务安全】

 日志系统B，并不是二进制日志，由于它是从MYSQL-A的二进制日志复制过来的，并不是自己的数据库变化产生的，有点接力的感觉，称为中继日志，即relay log。

 可以发现，通过上面的机制，可以保证MYSQL-A和MYSQL-B的数据库数据一致，但是时间上肯定有延迟，即MYSQL-B的数据是滞后的。

【即便不考虑什么网络的因素，MYSQL-A的数据库操作是可以并发的执行的，但是MYSQL-B只能从relay log中读一条，执行下。因此MYSQL-A的写操作很频繁，MYSQL-B很可能跟不上。】

## 解决问题

数据如何不被丢失

备份

读写分离

数据库负载均衡

高可用

## 服务器准备

|  |
| --- |
| 192.168.110.177 主服务器 master  192.168.110.178 从服务器slave |

## 修改主(master)服务器

|  |
| --- |
| vi /etc/my.cnf 新增以下内容  server\_id=177 ###服务器id  log-bin=mysql-bin ###开启日志文件 |

### 重启服务器

service mysqld start

service iptables stop

### 主服务器给从服务器账号授权

|  |
| --- |
| GRANT REPLICATION SLAVE ON \*.\* to 'mysync'@'%' identified by 'q123456'; |
| //一般不用root帐号，&ldquo;%&rdquo;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。 |

### 登录主服务器的mysql，查询master的状态

|  |
| --- |
| show master status; |

如果结果为null,则主服务器my.cf没有配置好.

## 修改从(slave)服务器

|  |
| --- |
| server\_id=178  log-bin=mysql-bin  binlog\_do\_db=test |

|  |
| --- |
| change master to master\_host='192.168.110.177',master\_user='mysync',master\_password='q123456',  master\_log\_file='mysql-bin.000002',master\_log\_pos=343; |

### 启动同步

|  |
| --- |
| start slave |

### 检查从服务器复制功能状态

SHOW SLAVE STATUS

|  |
| --- |
|  |

Slave\_IO\_Running: Yes    //此状态必须YES  
Slave\_SQL\_Running: Yes     //此状态必须YES

# 读写分离

## 什么是读写分离

在数据库集群架构中，让主库负责处理事务性查询，而从库只负责处理select查询，让两者分工明确达到提高数据库整体读写性能。当然，主数据库另外一个功能就是负责将事务性查询导致的数据变更同步到从库中，也就是写操作。

## 读写分离的好处

 1）分摊服务器压力，提高机器的系统处理效率

    读写分离适用于读远比写的场景，如果有一台服务器，当select很多时，update和delete会被这些select访问中的数据堵塞，等待select结束，并发性能并不高，而主从只负责各自的写和读，极大程度的缓解X锁和S锁争用；

   假如我们有1主3从，不考虑上述1中提到的从库单方面设置，假设现在1分钟内有10条写入，150条读取。那么，1主3从相当于共计40条写入，而读取总数没变，因此平均下来每台服务器承担了10条写入和50条读取（主库不承担读取操作）。因此，虽然写入没变，但是读取大大分摊了，提高了系统性能。另外，当读取被分摊后，又间接提高了写入的性能。所以，总体性能提高了，说白了就是拿机器和带宽换性能；

  2）增加冗余，提高服务可用性，当一台数据库服务器宕机后可以调整另外一台从库以最快速度恢复服务

## 主从复制原理

依赖于二进制日志，binary-log.

二进制日志中记录引起数据库发生改变的语句

Insert 、delete、update、create table

## Scale-up与Scale-out区别

Scale Out是指Application可以在水平方向上扩展。一般对数据中心的应用而言，Scale out指的是当添加更多的机器时，应用仍然可以很好的利用这些机器的资源来提升自己的效率从而达到很好的扩展性。

Scale Up是指Application可以在垂直方向上扩展。一般对单台机器而言，Scale Up值得是当某个计算节点（机器）添加更多的CPU Cores，存储设备，使用更大的内存时，应用可以很充分的利用这些资源来提升自己的效率从而达到很好的扩展性。

## MyCat

### 什么是  Mycat

是一个开源的分布式数据库系统，但是因为数据库一般都有自己的数据库引擎，而Mycat并没有属于自己的独有数据库引擎，所有严格意义上说并不能算是一个完整的数据库系统，只能说是一个在应用和数据库之间起桥梁作用的中间件。

在Mycat中间件出现之前，MySQL主从复制集群，如果要实现读写分离，一般是在程序段实现，这样就带来了一个问题，即数据段和程序的耦合度太高，如果数据库的地址发生了改变，那么我的程序也要进行相应的修改，如果数据库不小心挂掉了，则同时也意味着程序的不可用，而对于很多应用来说，并不能接受；

  引入Mycat中间件能很好地对程序和数据库进行解耦，这样，程序只需关注数据库中间件的地址，而无需知晓底层数据库是如何提供服务的，大量的通用数据聚合、事务、数据源切换等工作都由中间件来处理；

  Mycat中间件的原理是对数据进行分片处理，从原有的一个库，被切分为多个分片数据库，所有的分片数据库集群构成完成的数据库存储，有点类似磁盘阵列中的RAID0.

### Mycat安装

#### 创建表结构

|  |
| --- |
| **CREATE DATABASE IF NOT EXISTS `weibo\_simple`;**  -- ------------------------------------  -- Table structure for `t\_users` 用户表  -- ------------------------------------  **DROP TABLE IF EXISTS `t\_users`;**  **CREATE TABLE `t\_users` (**  **`user\_id`** varchar**(64) NOT NULL COMMENT '注册用户ID',**  **`user\_email`** varchar**(64) NOT NULL COMMENT '注册用户邮箱',**  **`user\_password`** varchar**(64) NOT NULL COMMENT '注册用户密码',**  **`user\_nikename`** varchar**(64) NOT NULL COMMENT '注册用户昵称',**  **`user\_creatime` datetime NOT NULL COMMENT '注册时间',**  **`user\_status` tinyint(1) NOT NULL COMMENT '验证状态 1：已验证 0：未验证',**  **`user\_deleteflag` tinyint(1) NOT NULL COMMENT '删除标记 1：已删除 0：未删除',**  **PRIMARY KEY (`user\_id`)**  **) ENGINE=InnoDB DEFAULT CHARSET=utf8;**  -- -------------------------------------  -- Table structure for `t\_message`微博表  -- -------------------------------------  **DROP TABLE IF EXISTS `t\_message`;**  **CREATE TABLE `t\_message` (**  **`messages\_id`** varchar**(64) NOT NULL COMMENT '微博ID',**  **`user\_id`** varchar**(64) NOT NULL COMMENT '发表用户',**  **`messages\_info`** varchar**(255) DEFAULT NULL COMMENT '微博内容',**  **`messages\_time` datetime DEFAULT NULL COMMENT '发布时间',**  **`messages\_commentnum`** int**(12) DEFAULT NULL COMMENT '评论次数',**  **`message\_deleteflag` tinyint(1) NOT NULL COMMENT '删除标记 1：已删除 0：未删除',**  **`message\_viewnum`** int**(12) DEFAULT NULL COMMENT '被浏览量',**  **PRIMARY KEY (`messages\_id`),**  **KEY `user\_id` (`user\_id`),**  **CONSTRAINT `t\_message\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `t\_users` (`user\_id`)**  **) ENGINE=InnoDB DEFAULT CHARSET=utf8;** |

#### 配置server.xml

|  |
| --- |
| <!-- 添加user -->  <user name="mycat">  <property name="password">mycat</property>  <property name="schemas">mycat</property>  </user>    <!-- 添加user -->  <user name="mycat\_red">  <property name="password">mycat\_red</property>  <property name="schemas">mycat</property>  <property name="readOnly">true</property>  </user> |

#### 配置schema.xml

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">  <mycat:schema xmlns:mycat="http://org.opencloudb/">  <!-- 与server.xml中user的schemas名一致 -->  <schema name="mycat" checkSQLschema="true" sqlMaxLimit="100">  <table name="t\_users" primaryKey="user\_id" dataNode="dn1" rule="rule1"/>  <table name="t\_message" type="global" primaryKey="messages\_id" dataNode="dn1" />  </schema>  <dataNode name="dn1" dataHost="jdbchost" database="weibo\_simple  " />    <dataHost name="jdbchost" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql" dbDriver="native" switchType="1"  slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <writeHost host="hostMaster" url="172.27.185.1:3306" user="root" password="root">  </writeHost>  <writeHost host="hostSlave" url="172.27.185.2:3306" user="root" password="root"/>  </dataHost>    </mycat:schema> |

#### 配置rule.xml文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!-- - - Licensed under the Apache License, Version 2.0 (the "License");  - you may not use this file except in compliance with the License. - You  may obtain a copy of the License at - - http://www.apache.org/licenses/LICENSE-2.0  - - Unless required by applicable law or agreed to in writing, software -  distributed under the License is distributed on an "AS IS" BASIS, - WITHOUT  WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. - See the  License for the specific language governing permissions and - limitations  under the License. -->  <!DOCTYPE mycat:rule SYSTEM "rule.dtd">  <mycat:rule xmlns:mycat="http://org.opencloudb/">  <tableRule name="rule1">  <rule>  <columns>user\_id</columns>  <algorithm>func1</algorithm>  </rule>  </tableRule>  <function name="func1" class="org.opencloudb.route.function.AutoPartitionByLong">  <property name="mapFile">autopartition-long.txt</property>  </function>  </mycat:rule> |

#### 为了更好地定位错误，修改log4j.xml

<level value="debug" />

双击startup\_nowrap.bat开始启动

# 常见问题

SHOW MASTER STATUS 如果为，则在my.ini文件中添加一行

log-bin=mysql-bin

给账号分配权限

grant all privileges on \*.\* to 'root'@'172.27.185.1' identified by 'root';

查询服务器server\_id

SHOW VARIABLES LIKE 'server\_id'