# 数据库读写分离

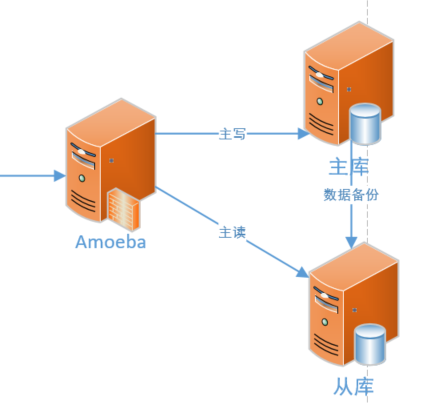
## Amoeba

### 介绍

Amoeba是一个以MySQL为底层数据存储，并对应用提供MySQL协议接口的proxy。它集中地响应应用的请求，依据用户事先设置的规则，将SQL请求发送到特定的数据库上执行。基于此可以实现负载均衡、[读写分离](https://baike.baidu.com/item/%E8%AF%BB%E5%86%99%E5%88%86%E7%A6%BB)、高可用性等需求。与MySQL官方的MySQL Proxy相比，作者强调的是amoeba配置的方便（基于XML的配置文件，用SQLJEP语法书写规则，比基于lua脚本的MySQL Proxy简单）。

Amoeba相当于一个SQL请求的[路由器](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1%E5%99%A8" \t "_blank)，目的是为负载均衡、读写分离、高可用性提供机制，而不是完全实现它们。用户需要结合使用MySQL的 Replication等机制来实现副本同步等功能。amoeba对底层数据库连接管理和路由实现也采用了可插拨的机制，第三方可以开发更高级的策略类来替代作者的实现。这个程序总体上比较符合[KISS原则](https://baike.baidu.com/item/KISS%E5%8E%9F%E5%88%99" \t "_blank)的思想。

### 实现读写分离

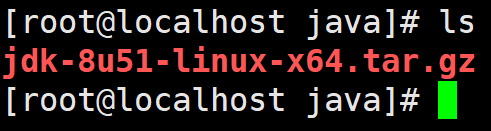


说明:实现数据库的读写分离,提供数据库处理能力,减少宕机的风险.

当用户有更新操作时,访问主数据库.当用户有读操作时访问从数据库.并且主从实现数据同步.实现读和写分离.高升数据库性能.

### 配置JDK

1.上传JDK文件

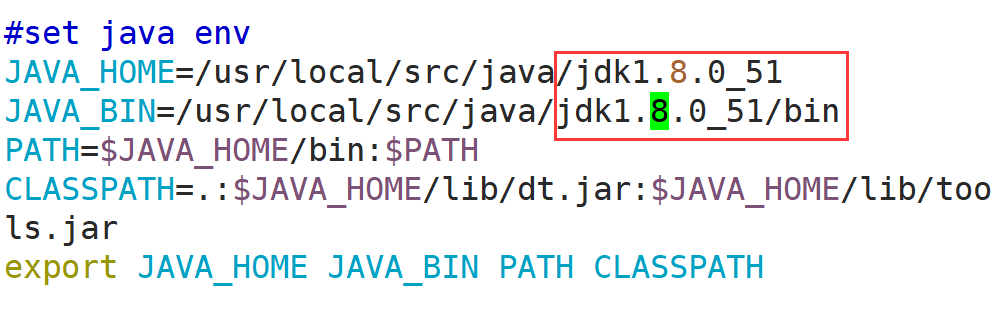


2.解压JDK

tar -xvf jdk-8u51-linux-x64.tar.gz

3.修改系统环境变量

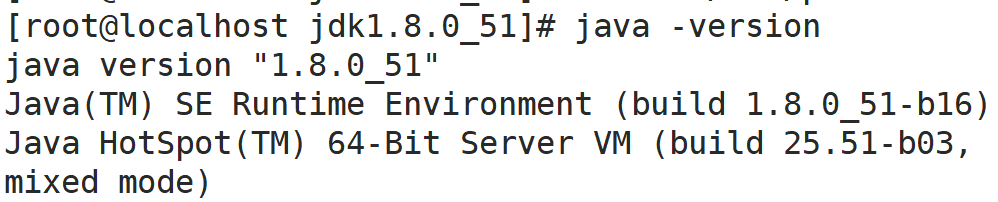
vim /etc/profile



4.让环境变量生效

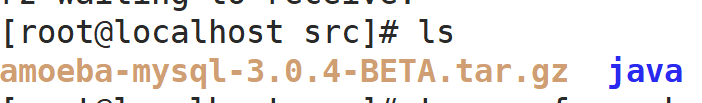
source /etc/profile

5.检测JDK是否生效



### 安装Amoeba

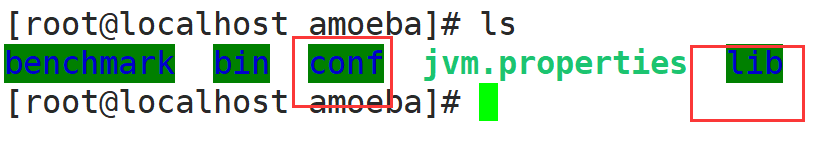
1. 上传文件



1. 解压文件

tar -xvf amoeba-mysql-3.0.4-BETA.tar.gz

1. 文件结构

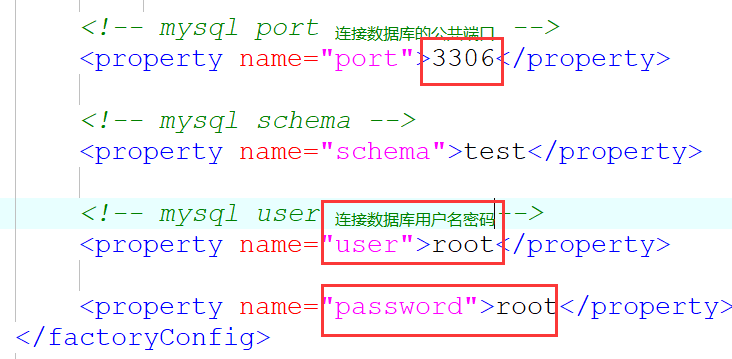


### Amoeba配置策略

1. 配置主库和从库的用户名和密码
2. 配置读写策略
3. 配置Amoeba自身配置文件 IP:用户名和密码:端口号
4. 需要修改jvm内存.

## 编辑dbService.xml

1. 配置公共的数据库链接



1. 定义数据库链接

<!--配置主库-->

<dbServer name="master" parent="abstractServer">

<factoryConfig>

<!-- mysql ip -->

<property name="ipAddress">192.168.126.162</property>

</factoryConfig>

</dbServer>

<dbServer name="slave01" parent="abstractServer">

<factoryConfig>

<!-- mysql ip -->

<property name="ipAddress">192.168.126.163</property>

</factoryConfig>

</dbServer>

1. 定义连接池

<dbServer name="multiPool" virtual="true">

<poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">

<!-- Load balancing strategy: 1=ROUNDROBIN , 2=WEIGHTBASED 挑选当前负载少的进行链接 , 3=HA-->

<property name="loadbalance">1</property>

<!-- Separated by commas,such as: server1,server2,server1 -->

<property name="poolNames">slave01,master,slave01</property>

</poolConfig>

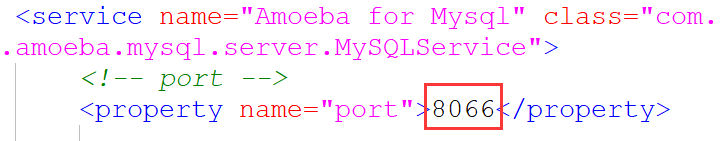
</dbServer>

1. 上传修改文件

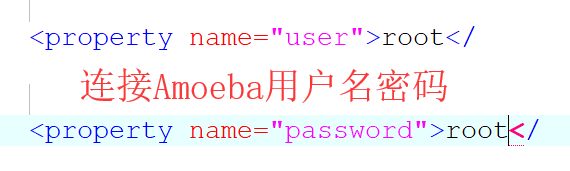
说明:将修改后的文件通过FZ工具上传到Linux系统下的Amoeba中

### 编辑Amoeba.xml

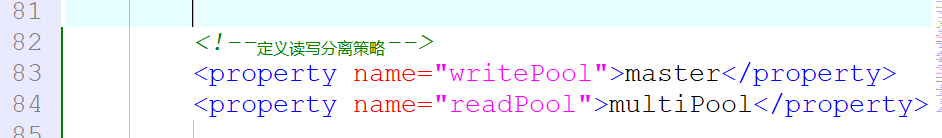
1. 定义链接端口



1. 配置用户名密码



1. 读写分离策略



### 修改JVM内存大小

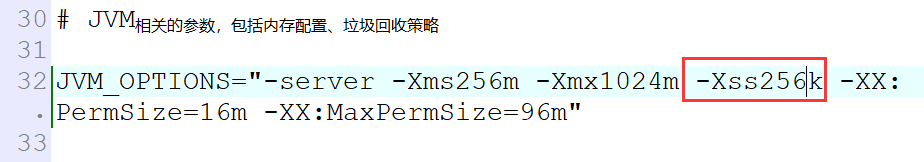
JVM\_OPTIONS="-server -Xms256m -Xmx1024m -Xss196k -XX:PermSize=16m -XX:MaxPermSize=96m"

-Xms256m JVM初始化内存大小

-Xmx1024m JVM最大内存

-Xss196k 指定线程空间大小

修改线程大小

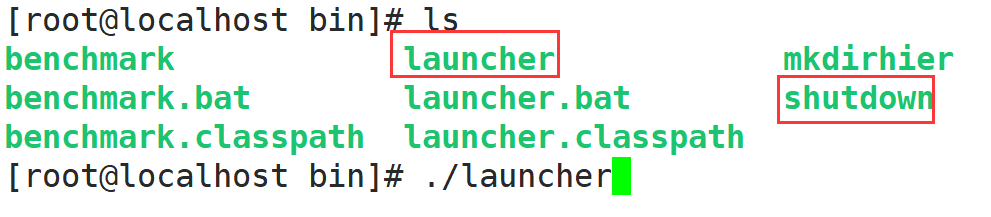


### 启动Amoeba

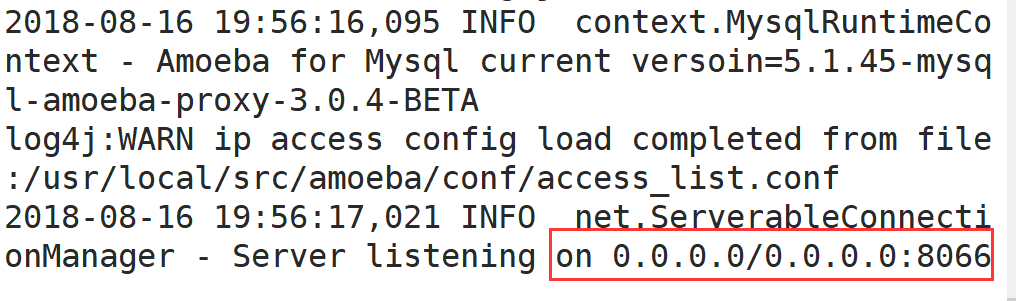
1. 关闭防火墙
2. 启动命令

./launcher

./shutdown



如果出现8066端口表示服务启动成功.



## 数据库双机热备形式

### 说明

根据数据库主从的配置.当主数据库做更新操作时,从库复制备份数据.当主库宕机时,通过某些技术手段可以数据库的高可用,实现故障迁移.当用户再有写入操作时,应该将数据写入从库中.如果主库修复完成重启数据库时,主库发现从库的数据较多.则数据库备份将失效.最终导致数据不一致的问题.

### 双机热备

说明:

A: 主库: 192.168.126.162

B: 从库: 192.168.126.163

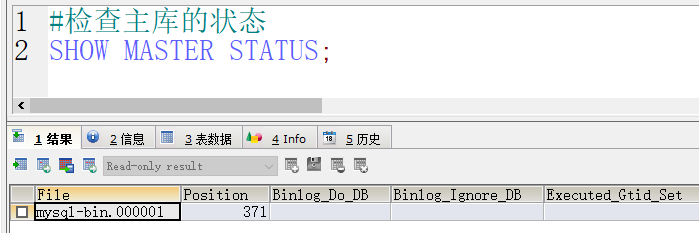
备份策略:

B备份A的数据

A备份B的数据

### 备份步骤:

1. 查询主库状态



1. 数据备份

#实现主从挂载

change MASTER to MASTER\_HOST="192.168.126.163",

MASTER\_PORT=3306,

MASTER\_USER="root",

MASTER\_PASSWORD="root",

MASTER\_LOG\_FILE="mysql-bin.000001",

MASTER\_LOG\_POS=371

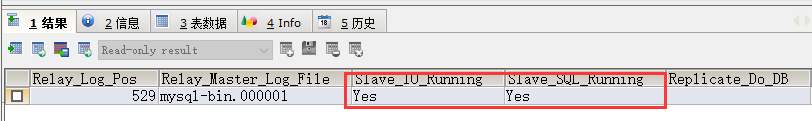
#开启主从服务

start SLAVE

#检查主从状态

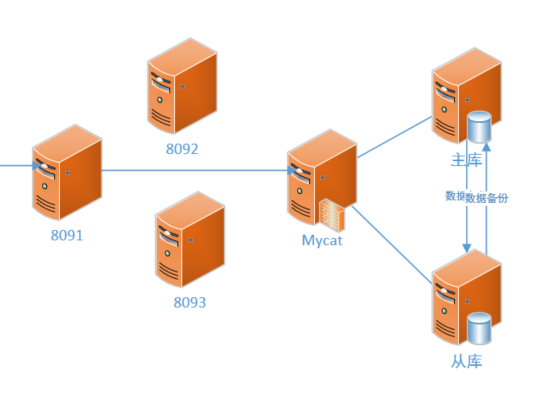
show SLAVE status

状态检查.



## Mycat数据库高可用实现

### 架构设计

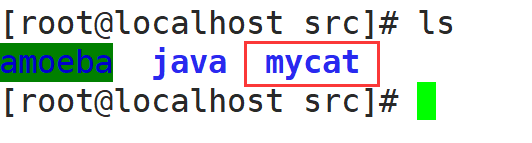


### Mycat介绍



### Mycat安装

1. 上传jar包后解压



### 编辑Server.xml

<!--用户标签-->

<user name="root">

<property name="password">root</property>

<!--与schema.xml中的配置相同 注意数据库的大小写-->

<property name="schemas">jtdb</property>

</user>

### 编辑Schema.xml

<!--name属性是自定义的 dataNode表示数据库的节点信息-->

<schema name="jtdb" checkSQLschema="false" sqlMaxLimit="100" dataNode="jtdb"/>

<!--定义节点名称/节点主机/数据名称-->

<dataNode name="jtdb" dataHost="localhost1" database="jtdb" />

<!--参数介绍-->

<!--balance 0表示所有的读操作都会发往writeHost主机 -->

<!--1表示所有的读操作发往readHost和闲置的主节点中-->

<!--writeType=0 所有的写操作都发往第一个writeHost主机-->

<!--writeType=1 所有的写操作随机发往writeHost中-->

<!--dbType 表示数据库类型 mysql/oracle-->

<!--dbDriver="native" 固定参数 不变-->

<!--switchType=-1 表示不自动切换, 主机宕机后不会自动切换从节点-->

<!--switchType=1 表示会自动切换(默认值)如果第一个主节点宕机后,Mycat会进行3次心跳检测,如果3次都没有响应,则会自动切换到第二个主节点-->

<!--并且会更新/conf/dnindex.properties文件的主节点信息 localhost1=0 表示第一个节点.该文件不要随意修改否则会出现大问题-->

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select 1</heartbeat>

<!--配置第一台主机主要进行写库操作,在默认的条件下Mycat主要操作第一台主机在第一台主机中已经实现了读写分离.因为默认写操作会发往137的数据库.读的操作默认发往141.如果从节点比较忙,则主节点分担部分压力.

-->

<writeHost host="hostM1" url="192.168.126.162:3306" user="root" password="root">

<!--读数据库-->

<readHost host="hostS1" url="192.168.126.163:3306" user="root" password="root" />

</writeHost>

<!--定义第二台主机 由于数据库内部已经实现了双机热备.-->

<!--Mycat实现高可用.当第一个主机137宕机后.mycat会自动发出心跳检测.检测3次.-->

<!--如果主机137没有给Mycat响应则判断主机死亡.则回启东第二台主机继续为用户提供服务.-->

<!--如果137主机恢复之后则处于等待状态.如果141宕机则137再次持续为用户提供服务.-->

<!--前提:实现双机热备.-->

<writeHost host="hostM2" url="192.168.126.163:3306" user="root" password="root">

<readHost host="hostS1" url="192.168.126.162:3306" user="root" password="root" />

</writeHost>

</dataHost>

### 启动Mycat

说明:将配置文件修改完成后,上传配置文件到指定目录下



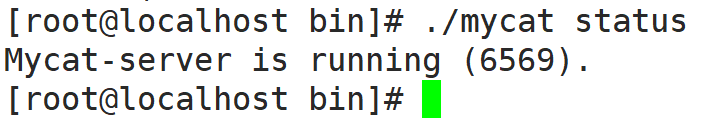
命令介绍:

./mycat start 启动

./mycat stop 停止

./mycat restart 重启

./mycat status 检测状态

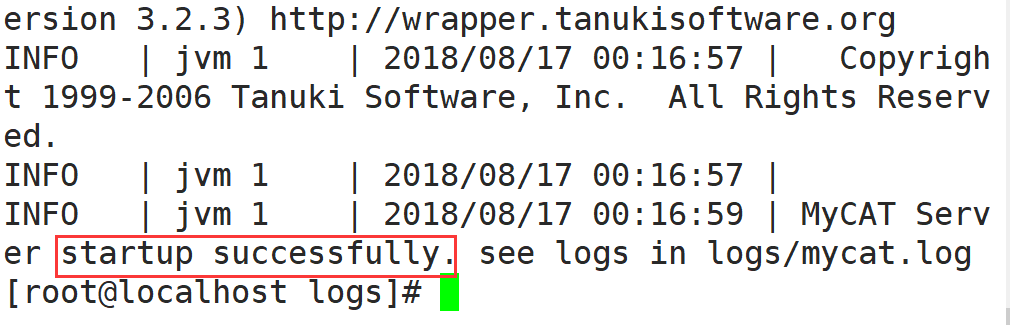


### 报错检查

说明:如果mycat启动错误需要检查日志文件是否正确

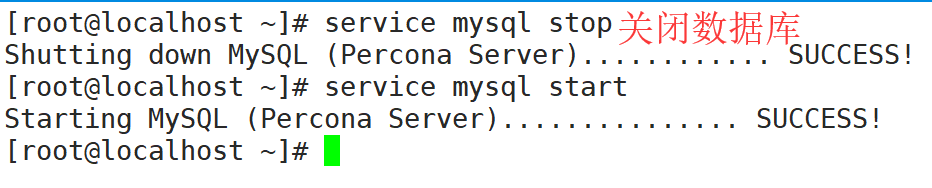
[root@localhost logs]# cat wrapper.log

日志状态:

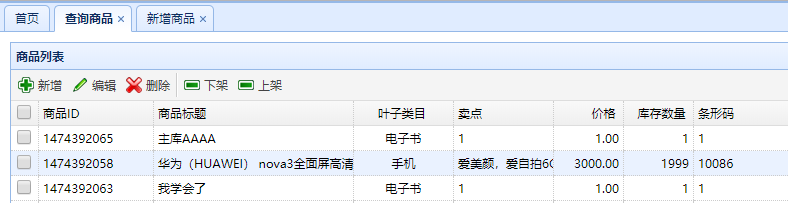


### 数据库高可用检查

说明:当程序正常启动后,将主库关闭



关闭主库后新增数据 查询是否正常.



之后启动mysql主库检查主库是否有信息

