# MyCat概述

## 历史

2013年阿里的Cobar在社区使用过程中发现存在一些比较严重的问题，及其使用限制，经过Mycat发 人第一次改良，第一代改良版——Mycat诞生。

Mycat开源以后，一些Cobar的用户参与了Mycat的 开发，最终Mycat发展成为一个由众多软件公司的实力派架构师和资深开发人员维护的社区型开源软件。

2014年Mycat首次在上海的《中华架构师》大会上对外宣讲，更多的人参与进来，随后越来越多的项目采用了Mycat。

2015年5月，由核心参与者们一起编写的第一本官方权威指南《Mycat权威指南》电子版发布，累计超过500本，成为开源项目中的首创。

2015年10月为止，Mycat项目总共有16个Committer。 截至2015年11月，超过300个项目采用Mycat，涵盖银行、电信、电子商务、物流、移动应用、O2O的众多领域和公司。

## 功能

Mycat是什么？从定义和分类来看，它是一个开源的分布式数据库系统，是一个实现了MySQL协议的的Server，前端用户可以把它看作是一个数据库代理，用MySQL客户端工具和命令行访问，而其后端可以用MySQL原生（Native）协议与多个MySQL服务器通信，也可以用JDBC协议与大多数主流数据库服务器通信，其核心功能是分表分库，即将一个大表水平分割为N个小表，存储在后端MySQL服务器里或者其他数据库里。

## 原理

Mycat的原理中最重要的一个动词是“拦截”，它拦截了用户发送过来的SQL语句，首先对SQL语句做了一些特定的分析：如分片分析、路由分析、读写分离分析、缓存分析等，然后将此SQL发往后端的真实数据库，并将返回的结果做适当的处理，最终再返回给用户。



上述图片里，Orders表被分为三个分片datanode（简称dn)，这三个分片是分布在两台MySQL Server上(DataHost)，即datanode=database@datahost方式，因此你可以用一台到N台服务器来分片，分片规则为（sharding rule)典型的字符串枚举分片规则，一个规则的定义是分片字段（sharding column)+分片函数(rule function)，这里的分片字段为prov而分片函数为字符串枚举方式。

当Mycat收到一个SQL时，会先解析这个SQL，查找涉及到的表，然后看此表的定义，如果有分片规则，则获取到SQL里分片字段的值，并匹配分片函数，得到该SQL对应的分片列表，然后将SQL发往这些分片去执行，最后收集和处理所有分片返回的结果数据，并输出到客户端。以select \* from Orders where prov=?语句为例，查到prov=wuhan，按照分片函数，wuhan返回dn1，于是SQL就发给了MySQL1，去取DB1上的查询结果，并返回给用户。

## 应用场景

Mycat发展到现在，适用的场景已经很丰富，而且不断有新用户给出新的创新性的方案，以下是几个典型的应用场景：

* 单纯的读写分离，此时配置最为简单，支持读写分离，主从切换
* 分表分库，对于超过1000万的表进行分片，最大支持1000亿的单表分片
* 多租户应用，每个应用一个库，但应用程序只连接Mycat，从而不改造程序本身，实现多租户化
* 报表系统，借助于Mycat的分表能力，处理大规模报表的统计
* 替代Hbase，分析大数据
* 作为海量数据实时查询的一种简单有效方案，比如100亿条频繁查询的记录需要在3秒内查询出来结果，除了基于主键的查询，还可能存在范围查询或其他属性查询，此时Mycat可能是最简单有效的选择

# 概念原理

## 分库分片

* 垂直切分

一个数据库由很多表的构成，每个表对应着不同的业务，垂直切分是指按照业务将表进行分类，分布到不同的数据库上面，这样也就将数据或者说压力分担到不同的库上面，如下图：



* 水平切分

相对于垂直拆分，水平拆分不是将表做分类，而是按照某个字段的某种规则来分散到多个库之中，每个表中包含一部分数据。简单来说，我们可以将数据的水平切分理解为是按照数据行的切分，就是将表中的某些行切分到一个数据库，而另外的某些行又切分到其他的数据库中，如图



## 读写分离

数据库读写分离对于大型系统或者访问量很高的互联网应用来说，是必不可少的一个重要功能。 从数据库的角度来说，对于大多数应用来说，从集中到分布，最基本的一个需求不是数据存储的瓶颈，而是在于计算的瓶颈，即SQL查询的瓶颈，我们知道，正常情况下，Insert SQL就是几十个毫秒的时间内写入完成，而系统中的大多数Select SQL则要几秒到几分钟才能有结果，很多复杂的SQL，其消耗服务器CPU的能力超强，不亚于死循环的威力。在没有读写分离的系统上，很可能高峰时段的一些复杂SQL查询就导致数据库服务器CPU爆表，系统陷入瘫痪，严重情况下可能导致数据库崩溃。因此，从保护数据库的角度来说，我们应该尽量避免没有主从复制机制的单节点数据库。

对于MySQL来说，标准的读写分离是主从模式，一个写节点Master后面跟着多个读节点，读节点的数量取决于系统的压力，通常是1-3个读节点的配置，如下图所示：



MySQL支持更多的主从复制的拓扑关系，如下图所示，但通常我们不会采用双向主从同步以及环状的拓扑：



# 安装配置

## 环境准备

1) JDK下载

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

注：必须JDK7或更高版本.

2) MySQL下载

http://dev.mysql.com/downloads/mysql/5.6.html#downloads

注：MyCAT支持多种数据库接入，如：MySQL、SQLServer、Oracle、MongoDB等，推荐使用MySQL做集群。

3) MyCAT下载

[http://dl.mycat.io/](http://www.mycat.io/)

注：建议下载1.6-RELEASE稳定版本。

## 安装

解压缩完成安装,Linux下copy文件到/usr/local/mycat目录

Mycat目录如下：

bin：存放运行程序

conf：存放配置文件

lib：运行依赖的Jar文件

logs：存放日志文件

catlet：Mycat的扩展功能

conf目录下主要配置文件：

| **文件** | **说明** |
| --- | --- |
| server.xml | Mycat的配置文件，设置账号、参数等 |
| schema.xml | Mycat对应的物理数据库和数据库表的配置 |
| rule.xml | Mycat分片（分库分表）规则 |

运行mycat  
/usr/local/mycat/bin/mycat restart

使用Mycat后，连接数据库是不变的，默认端口是8066。连接方式和普通数据库一样，如：jdbc:mysql://192.168.0.2:8066/

或命令行：

mysql -uroot -p123456 -h127.0.0.1 -P8066 -DTESTDB

## 分库分表

* 配置文件schema.xml

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<!-- 数据库配置，与server.xml中的数据库对应 -->

<schema name="lunch" checkSQLschema="false" sqlMaxLimit="100">

<table name="lunchmenu" dataNode="dn1" />

<table name="restaurant" dataNode="dn1" />

<table name="userlunch" dataNode="dn1" />

<table name="users" dataNode="dn1" />

<table name="dictionary" primaryKey="id" autoIncrement="true" dataNode="dn1,dn2" rule="mod-long" />

</schema>

<!-- 分片配置 -->

<dataNode name="dn1" dataHost="test1" database="lunch" />

<dataNode name="dn2" dataHost="test2" database="lunch" />

<!-- 物理数据库配置 -->

<dataHost name="test1" maxCon="1000" minCon="10" balance="0" writeType="0" dbType="mysql" dbDriver="native">

<heartbeat>select user();</heartbeat>

<writeHost host="hostM1" url="192.168.0.2:3306" user="root" password="123456">

</writeHost>

</dataHost>

<dataHost name="test2" maxCon="1000" minCon="10" balance="0" writeType="0" dbType="mysql" dbDriver="native">

<heartbeat>select user();</heartbeat>

<writeHost host="hostS1" url="192.168.0.3:3306" user="root" password="123456">

</writeHost>

</dataHost>

</mycat:schema>

## 读写分离

* 配置文件schema.xml

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<!-- 数据库配置，与server.xml中的数据库对应 -->

<schema name="lunch" checkSQLschema="false" sqlMaxLimit="100">

<table name="lunchmenu" dataNode="dn1" />

<table name="restaurant" dataNode="dn1" />

<table name="userlunch" dataNode="dn1" />

<table name="users" dataNode="dn1" />

<table name="dictionary" primaryKey="id" autoIncrement="true" dataNode="dn1" />

</schema>

<!-- 分片配置 -->

<dataNode name="dn1" dataHost="test1" database="lunch" />

<!-- 物理数据库配置 -->

<dataHost name="test1" maxCon="1000" minCon="10" balance="1" writeType="0" dbType="mysql" dbDriver="native">

<heartbeat>select user();</heartbeat>

<writeHost host="hostM1" url="192.168.0.2:3306" user="root" password="123456">

<readHost host="hostM1" url="192.168.0.3:3306" user="root" password="123456">

</readHost>

</writeHost>

</dataHost>

</mycat:schema>

* 主从复制

Mycat并不提供主从复制功能，必须通过Mysql自身的复制机制实现：

主库创建用户并授权

GRANT REPLICATION SLAVE ON \*.\* TO 'repl'@'%' IDENTIFIED BY 'mysql';

然后执行quit命令退出mysql服务，执行如下命令重启mysql：

service mysql restart

修改mysql配置文件mysqld.cnf

--Master库配置文件添加如下内容

server-id=1

log-bin=master-bin

log-bin-index=master-bin.index

binlog-do-db = db1

binlog-ignore-db = mysql

binlog-ignore-db = information\_schema

binlog\_format = mixed #binlog日志格式，mysql默认采用，如果从服务器slave有别的slave要复制那么该slave也需要该项 #

expire\_logs\_days = 7

执行show master status;查看Master状态

--Slave库配置文件

server-id=2

relay-log-index=slave-relay-bin.index

relay-log=slave-relay-bin

replicate-do-db = db1

replicate-ignore-db = mysql

replicate-ignore-db = information\_schema

log\_slave\_updates = 1

--slave连接master

change master to master\_host='10.10.8.76',

master\_port=3306,

master\_user='repl',

master\_password='mysql',

master\_log\_file='master-bin.000002',

master\_log\_pos=0;

start slave

执行show slave status;查看Slave状态