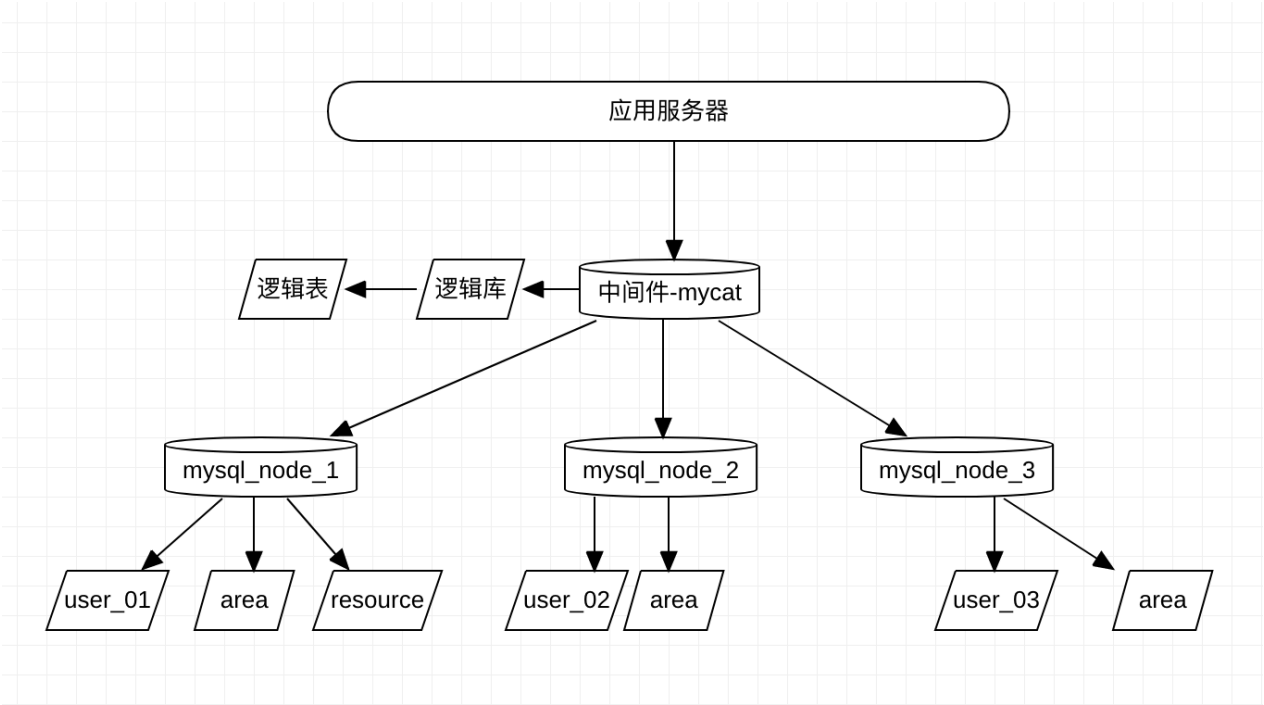
Mycat介绍

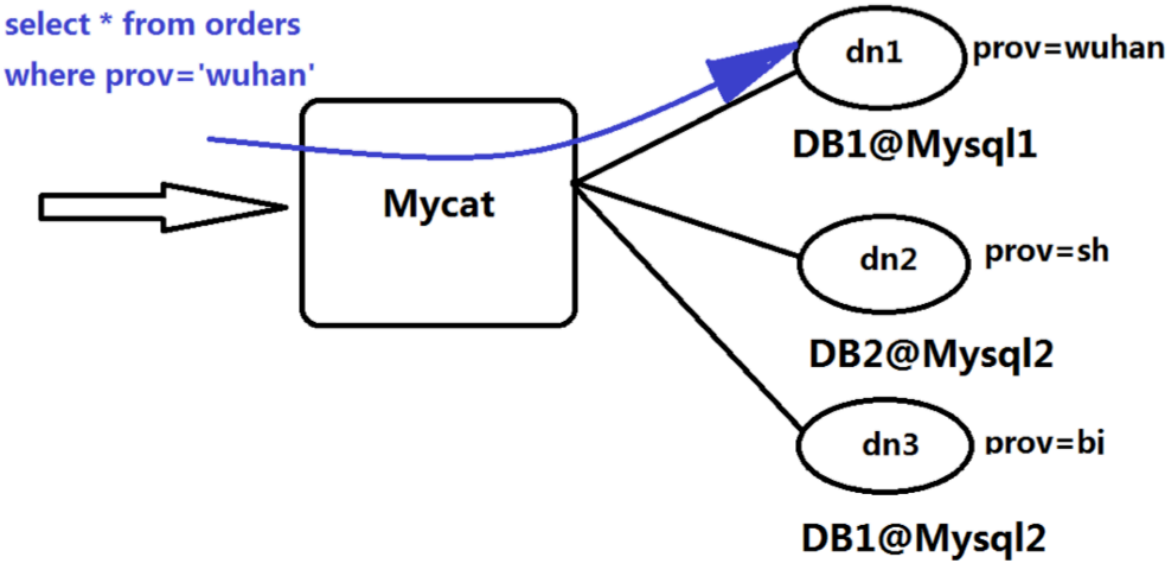
1、mycat是什么

mycat是一个实现了 MySQL 协议的 Server，前端用户可以把它看作是一个数据库代理，介于数据库与应用之间，进行数据处理与交互的中间服务，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，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了整个完整的数据库存储。



2、mycat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。



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。

如果上述 SQL 改为 select * from Orders where prov in ('wuhan','beijing')，那么，SQL 就会发给 MySQL1 与 MySQL2 去执行，然后结果集合并后输出给用户。但通常业务中我们的 SQL 会有 Order By 以及 Limit 翻页语法，此时就涉及到结果集在 Mycat 端的二次处理，这部分的代码也比较复杂，而最复杂的则属两个 表的 Jion 问题，为此，Mycat 提出了创新性的 ER 分片、全局表等

3、mycat中相关概念

3.1、 schema(逻辑库)

数据库中间件，通常对实际应用来说，并不需要知道中间件的存在，业务开发人员只需要知道 数据库的概念，所以数据库中间件可以被看做是一个或多个数据库集群构成的逻辑库。

在云计算时代，数据库中间件可以以多租户的形式给一个或多个应用提供服务， 每个应用访问的可能是一个 独立或者是共享的物理库，常见的如阿里云数据库服务器 RDS。

3.2、 逻辑表

既然有逻辑库，那么就会有逻辑表，分布式数据库中，对应用来说，读写数据的表就是逻辑表。逻辑表，可以是数据切分后，分布在一个或多个分片库中， 也可以不做数据切分，不分片， 只有一个表构成。

3.2.1、分片表

分片表，是指那些原有的很大数据的表，需要切分到多个数据库的表，这样，每个分片都有一部分数据，所 有分片构成了完整的数据。

例如在 mycat 配置中的 t_node 就属于分片表， 数据按照规则被分到 dn1,dn2 两个分片节点 (dataNode) 上。

```
<table name="t_node" primaryKey="vid" autoIncrement="true" dataNode="dn1,dn2"
rule="rule1" />
```

3.2.2、非分片表

一个数据库中并不是所有的表都很大，某些表是可以不用进行切分的，非分片是相对分片表来说的， 就是那 些不需要进行数据切分的表。

如下配置中 t_node，只存在于分片节点（dataNode）dn1 上。

```
<table name="t_node" primaryKey="vid" autoIncrement="true" dataNode="dn1" />
```

3.2.3、 ER 表

关系型数据库是基于实体关系模型（Entity-Relationship Model)之上， 通过其描述了真实世界中事物与关 系，Mycat 中的 ER 表即是来源于此。 根据这一思路，提出了基于 E-R 关系的数据分片策略，子表的记录与所关 联的父表记录存放在同一个数据分片上，即子表依赖于父表， 通过表分组（Table Group）保证数据 Join 不会跨 库操作。

表分组（Table Group）是解决跨分片数据 join 的一种很好的思路，也是数据切分规划的重要一条规则。

3.2.3、全局表

一个真实的业务系统中，往往存在大量的类似字典表的表，这些表基本上很少变动，字典表具有以下几个特 性：

- 变动不频繁
- 数据量总体变化不大
- 数据规模不大，很少有超过数十万条记录。

对于这类的表，在分片的情况下， 当业务表因为规模而进行分片以后，业务表与这些附属的字典表之间的关 联，就成了比较棘手的问题，所以 Mycat 中通过数据冗余来解决这类表的 join ，即所有的分片都有一份数据的拷 贝，所有将字典表或者符合字典表特性的一些表定义为全局表 。

数据冗余是解决跨分片数据 join 的一种很好的思路，也是数据切分规划的另外一条重要规则。

3.3、分片节点(dataNode)

数据切分后，一个大表被分到不同的分片数据库上面，每个表分片所在的数据库就是分片节点（dataNode）。

3.4、节点主机(dataHost)

数据切分后，每个分片节点（dataNode）不一定都会独占一台机器， 同一机器上面可以有多个分片数据库， 这样一个或多个分片节点（dataNode）所在的机器就是节点主机（dataHost）,为了规避单节点主机并发数限 制，尽量将读写压力高的分片节点（dataNode）均衡的放在不同的节点主机（ dataHost）。

3.5、 分片规则(rule)

前面讲了数据切分， 一个大表被分成若干个分片表，就需要一定的规则，这样按照某种业务规则把数据分到 某个分片的规则就是分片规则，数据切分选择合适的分片规则非常重要，将极大的避免后续数据处理的难度。

3.6、 全局序列号(sequence)

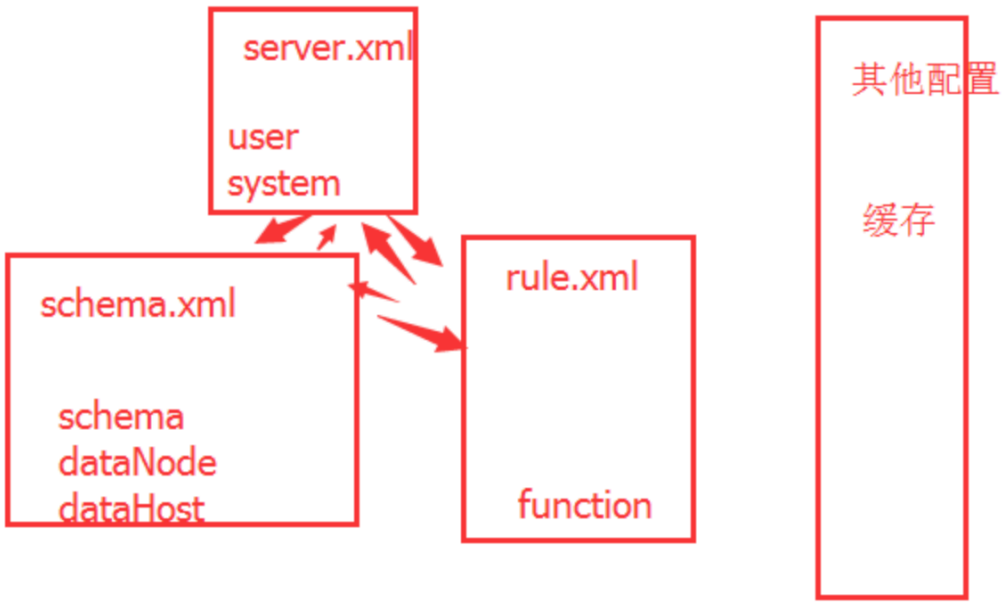
数据切分后，原有的关系数据库中的主键约束在分布式条件下将无法使用， 因此需要引入外部机制保证数据 唯一性标识，这种保证全局性的数据唯一标识的机制就是全局序列号（ sequence）。

4、mycat目录

目录	描述
bin	bin 程序目录，存放了 window 版本和 linux 版本，除了提供封装成服务的版本之外，也提供了 nowrap 的 shell 脚本命令，方便大家选择和修改，进入到 bin 目录： Linux 下运行：./mycat console, 首先要 chmod +x * mycat 支持的命令{ console
conf	目录下存放配置文件， server.xml 是 Mycat 服务器参数调整和用户授权的配置文件，schema.xml 是逻 辑库定义和表以及分片定义的配置文件，rule.xml 是分片规则的配置文件，分片规则的具体一些参数信息单独存 放为文件，也在这个目录下，配置文件修改，需要重启 Mycat 或者通过 9066 端口 reload.
lib	主要存放 mycat 依赖的一些 jar 文件.
logs	日志存放在 logs/mycat.log 中，每天一个文件，日志的配置是在 conf/log4j.xml 中，根据自己的需要，可 以调整输出级别为 debug，debug 级别下，会输出更多的信息，方便排查问题.

4.1、配置mycat

mycat 的主要配置在conf目录下，下面图片描述了Mycat最重要的3大配置文件：



4.1.1 server.xml

此配置文件在mycat的conf目录下，定义用户以及系统相关变量，如端口等.

```
<user name="user">
    <property name="password">123456</property>
    <property name="schemas">test</property>
    <property name="readOnly">true</property>
    <property name="benchmark">0</property>
    <property name="usingDecrypt">1</property>
    <privileges check="false">
      <schema name="test" dml="0110" >
        <table name="tb01" dml="0000"></table>
        <table name="tb02" dml="1111"></table>
      </schema>
    </privileges>
  </user>
```

基本属性配置

属性名称	默认值	描述
password		设置用户的密码
readOnly		限制用户对数据库是否可读写的权限，true为只读，false为读写
schemas		文本来控制用户可放问的 schema，多个schema用，号隔开
benchmark		： benchmark 基准, 当前端的整体 connection 数达到基准值是, 对来自该账户的请求开始拒绝连接，0 或不设表示不限制
usingDecrypt	0	是否对密码加密默认 0 否 如需要开启配置 1，同时使用加密程序对密码加密

加密命令方法是执行mycat的jar程序：

```
java -cp Mycat-server-1.6.7.1-release.jar org.opencloudb.util.DecryptUtil
0:user:password
```

子权限管理

对用户的 schema 及 下级的 table 进行精细化的 DML 权限控制，privileges 节点中的 check 属性是用于标识是否开启 DML 权限检查， 默认 false 标识不检查，如果 privileges 节点不配置，就等同于 check=false。

参数	说明	事例(禁止增删改查)
dml	insert,update,select,delete	000

dml权限顺序为：insert(新增),update(修改),select(查询),delete(删除),0000--> 1111,0为禁止权限，1为开启权限。 例如对schema设置权限、对table设置权限

设置了 schema , 但只设置了个别 table 或 未设置 table 的 DML，自动继承 schema 的 DML 属性

防火墙配置管理

防火墙就是网络层对请求的地址进行限制，主要是从安全角度来保证Mycat不被匿名IP进行访问，通过白名单和 SQL 黑名单对服务器进行详细对权限控制。

```
<firewall>
     <whitehost>
        <host host="1*7.0.0.*" user="root"/>
        <host host="1*8.0.0.*" user="root"/>
     </whitehost>
       <blacklist check="false">
       </blacklist>
  </firewall>
```

4.1.2 schema.xml配置

管理着 MyCat 的逻辑库、表、分片规则、DataNode 以 及 DataSource。

schema标签

schema 标签用于定义 MyCat 实例中的逻辑库，MyCat 可以有多个逻辑库，每个逻辑库都有自己的相关配 置。可以使用 schema 标签来划分这些不同的逻辑库。如果不配置 schema 标签，所有的表配置，会属于同一个默认的逻辑库。

逻辑库的概念和 MYSQL 数据库中 Database 的概念相同，我 们在查询这两个不同的逻辑库中表的时候需要切换到该逻辑库下才可以查询到所需要的表。

```
<schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn2">
</schema>
```

属性	说明
name	逻辑数据库名，与server.xml中的schema对应
checkSQLschema	数据库前缀相关设置，建议看文档，这里暂时设为folse
sqlMaxLimit	当该值设置为某个数值时。每条执行的 SQL 语句， 如果没有加上 limit 语句，MyCat 也会自动的加上所对应的值。设置该值的话， MyCat 默认会把查询到的信息全部都展示出来，造成过多的输出。所以，在正常使用中，还是建议加上一个值， 用于减少过多的数据返回。**当然 SQL 语句中也显式的指定 limit 的大小，不受该属性的约束，如果运行的 schema 为非拆分库的，那么该属性不会生效。 需要手动添加 limit 语句**
dataNode	该属性用于绑定逻辑库到某个具体的 database 上；配置默认分片，只需要配置需要分片的表即可。**注意没有配置在分片里面的表工具查看无法显示，但是可以正常使用**

table标签:

```
<table name="travelrecord" primaryKey="id" autoIncrement='false'
dataNode="dn1,dn2,dn3" rule="auto-sharding-long" ></table>
```

属性	说明
name	定义逻辑表的表名， 这个名字就如同在数据库中执行 create table 命令指定的名字一样，同个 schema 标 签中定义的名字必须唯一。
dataNode	定义这个逻辑表所属的 dataNode, 该属性的值需要和 dataNode 标签中 name 属性的值相互对应。
primaryKey	该逻辑表对应真实表的主键，例如：分片的规则是使用非主键进行分片的，那么在使用主键查询的时候，就 会发送查询语句到所有配置的 DN 上，如果使用该属性配置真实表的主键。难么 MyCat 会缓存主键与具体 DN 的 信息，那么再次使用非主键进行查询的时候就不会进行广播式的查询，就会直接发送语句给具体的 DN，但是尽管 配置该属性，如果缓存并没有命中的话，还是会发送语句给具体的 DN，来获得数据。
autoIncrement	是否自增
ruleRequired	该属性用于指定表是否绑定分片规则，如果配置为 true，但没有配置具体 rule 的话 ，程序会报错。
rule	该属性用于指定逻辑表要使用的规则名字，规则名字在 rule.xml 中定义， 必须与 tableRule 标签中 name 属 性属性值一一对应。
type	该属性定义了逻辑表的类型，目前逻辑表只有“全局表”和”普通表”两种类型。对应的配置： 全局表：global。 普通表：不指定该值为 globla 的所有表。
needAddLimit	指定表是否需要自动的在每个语句后面加上 limit 限制。由于使用了分库分表， 数据量有时会特别巨大。这时 候执行查询语句，如果恰巧又忘记了加上数量限制的话。那么查询所有的数据出来，也够等上一小会儿的

childTable 子表，具有E-R关系的表，即具有父子关系的表或者是一对多关系的表，例如订单表tbl_order和订单项表tbl_order_item, 子表中可以嵌套子表

属性名	值
joinKey	外键字段，例如tbl_order_item表中的字段order_id
parentKey	属性指定的值一般为与父表建立关联关系的列名。程序首先获取 joinkey 的值， 再通过 parentKey 属性指定 的列名产生查询语句，通过执行该语句得到父表存储在哪个分片上。从而确定子表存储的位置。

dataNode

标签定义了 MyCat 中的数据节点，也就是我们通常说所的数据分片。一个 dataNode 标签就是 一个独立的数据分片

```
<dataNode name="dn1" dataHost="lch3307" database="db1" ></dataNode>
```

属性	说明
name	节点名，与table中dataNode对应
datahost	该属性用于定义该分片属于哪个数据库实例的， 属性值是引用 dataHost 标签上定义的 name 属性。
database	该属性用于定义该分片属性哪个具体数据库实例上的具体库， 因为这里使用两个纬度来定义分片，就是：实 例 具体的库。因为每个库上建立的表和表结构是一样的。所以这样做就可以轻松的对表进行水平拆分。

dataHost

定义了具 体的数据库实例、读写分离配置和心跳语句

```
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="0" writeType="0"
dbType="mysql" dbDriver="native">
<heartbeat>select user()</heartbeat>
<!-- can have multi write hosts -->
<writeHost host="hostM1" url="localhost:3306" user="root" password="123456">
<!-- can have multi read hosts -->
<!-- <readHost host="hostS1" url="localhost:3306" user="root" password="123456"/> -->
</writeHost>
<!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->
</dataHost>
```

属性	说明
name	物理数据库名，与dataNode中dataHost对应
maxCon	指定每个读写实例连接池的最大连接
minCon	指定每个读写实例连接池的最小连接，初始化连接池的大小
balance	负载均衡类型，很重要的一个属性, 一般读写分离要设置成1。 0: 不开启读写分离机制，所有读操作都发送到当前可用的writeHost上 1: 全部的 readHost 与 stand by writeHost 参与 select 语句的负载均衡，简单的说，当双主双从模式(M1->S1，M2->S2，并且 M1 与 M2 互为主备)，正常情况下，M2,S1,S2 都参与 select 语句的负载 均衡。 2: 所有读操作都随机的在 writeHost、readhost 上分发 3: 所有读请求随机的分发到 wiriterHost 对应的 readhost 执行，writerHost 不负担读压力
writeType	写操作的负载均衡，一般设置为0，表示所有写操作发送到配置的第一个writeHost，第一个挂了切到还生存的第二个writeHost，重新启动后已切换后的为准，切换记录在配置文件中:dnindex.properties
switchType	主从切换类型 1：默认值，表示自动切换 2: 基于MySQL主从同步的状态决定是否切换(心跳语句为 show slave status) ，一般用于读写分离 3：基于 MySQL galary cluster 的切换机制(适合集群) 心跳语句为 show status like ‘wsrep%’
dbType	数据库的类型，有mysql、oracle、mongodb、spark等
dbDriver	一般设置为native

writeHost和readHost

这两个标签都指定后端数据库的相关配置给 mycat，用于实例化后端连接池。 唯一不同的是，writeHost 指 定写实例、readHost 指定读实例，组着这些读写实例来满足系统的要求。

在一个 dataHost 内可以定义多个 writeHost 和 readHost。但是，如果 writeHost 指定的后端数据库宕机， 那么这个 writeHost 绑定的所有 readHost 都将不可用。另一方面，由于这个 writeHost 宕机系统会自动的检测 到，并切换到备用的 writeHost 上去。

属性名	值
host	主机名，主一般使用Master中的M后缀来结尾，M后面使用1、2、3等序号标识
url	配置实际物理数据库的ip和端口，例如url="127.0.0.1:3306"
user	物理数据库的用户名
password	物理数据库的密码
weight	权重 配置在 readhost 中作为读节点的权重
usingDecrypt	是否对密码加密默认 0 否 如需要开启配置 1，同时使用加密程序对密码加密

加密命令为**(执行 mycat jar 程序（1.4.1 以后))**：

```
java -cp Mycat-server-1.4.1-dev.jar org.opencloudb.util.DecryptUtil
1:host:user:password
Mycat-server-1.4.1-dev.jar 为 mycat download 下载目录的 jar
1:host:user:password 中 1 为 db 端加密标志，host 为 dataHost 的 host 名称
```

Mycat主从分离只是在读的时候做了处理，写入数据的时候，只会写入到writehost，需要通过数据库本身做主从同步将数据同步到readhost

> 至于其他的场景，如同时主从和分表分库也是支持的了，只要了解这个实现以后再去修改配置，都是可以实现的。而热备及故障专业官方推荐使用haproxy配合一起使用

4.1.3 rule.xml

定义了我们对表进行拆分所涉及到的规则定义。 我们可以灵活的对表使用不同的分片算法， 或者对表使用相同的算法但具体的参数不同。这个文件里面主要有 tableRule 和 function 这两个标签。在具体使用过程中可以按照需求添加 tableRule 和 function。

5、服务启动与启动设置

5.1 安装环境

MyCAT 是使用 JAVA 语言进行编写开发，使用前需要先安装 JAVA 运行环境(JRE),由于 MyCAT 中使用了 JDK7 中的一些特性，所以要求必须在 JDK7 以上的版本上运行。

5.2 服务启动

5.2.1、Linux

MyCAT 在 Linux 中部署启动时，首先需要在 Linux 系统的环境变量中配置 MYCAT_HOME,操作方式如下：

1) vi /etc/profile,在系统环境变量文件中增加 MYCAT_HOME=/usr/local/Mycat

2) 执行 source /etc/profile 命令，使环境变量生效。

如果是在多台 Linux 系统中组建的 MyCAT 集群，那需要在 MyCAT Server 所在的服务器上配置对其他 ip 和 主机名的映射，配置方式如下：

vi /etc/hosts

例如：我有 4 台机器，配置如下：

IP 主机名：

```
192.168.100.2 sam_server_1
192.168.100.3 sam_server_2
192.168.100.4 sam_server_3
192.168.100.5 sam_server_4
```

编辑完后，保存文件。

经过以上两个步骤的配置，就可以到/usr/local/Mycat/bin 目录下执行：

./mycat start

即可启动 mycat 服务！

5.2.2 windows

MyCAT 在 windows 中部署时，建议放在某个盘符的根目录下，如果不是在根目录下，请尽量不要放在包含 中文的目录下

如：D:\Mycat-server-1.6.7.1-win\

命令行方式启动：

从 cmd 中执行命令到达 D:\ Mycat-server-1.6.7.1-win\bin 目录下，执行 startup_nowrap.bat 即可启动 MyCAT 服务。

注：执行此命令时， 需要确保 windows 系统中已经配置好了 JAVA 的环境变量，并可执行 java 命令。jdk 版 本必须是 1.7 及以上版本。

5.3 基于 zk 的启动

https://www.cnblogs.com/leeSmall/p/9551038.html

参考：

https://blog.csdn.net/vbirdbest/article/details/83514361

https://blog.csdn.net/vbirdbest/article/details/83448757

https://blog.csdn.net/vbirdbest/article/details/83619960

https://blog.csdn.net/vbirdbest/article/details/83624665

https://blog.csdn.net/vbirdbest/article/details/83476292