Mycat和海量关系型数据读写(20:30)

2月26日脱产班，**3月12日**周末班，**3月19日**网络班

欢迎你的加入！

需要代码、PPT、视频资料请加下咨询老师QQ：

贾老师：1786418286

何老师：1926106490

詹老师：2805048645

讨论技术可以加入QQ群：172599077,156927834

Mycat是什么

- Mycat是什么
- Mycat的由来
- Mycat的架构
- Mycat使用方法
- Mycat解决海量关系型数据读写的最优方案

Mycat是什么

- 一个用于MySQL读写分离和与数据切分的高可用中间件
- 一个模拟为MySQL Server的超级数据库代理
- 一个能平滑扩展支持1000亿大表的分布式数据库系统
- 一个可管控多种关系数据库的数据库路由器

Mycat之前世今生

- 2013年阿里的Cobar在某大型项目中使用过程中发现存在一些比较严重的问题，于是第一代改良版——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，引发围观，更多的人参与进来，随后越来越多的项目采用了Mycat
- 2015年7月为止，Mycat项目总共有16个Committer，其中核心参与者的年薪总额超过200万
- 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。
- 截至2015年7月，超过100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。

Mycat之前世今生

- 截至2014年7月，Mycat官方QQ群（106088787）已经超过2700人，大多数为资深IT工程师、架构师、DBA、以及一些CXO和高端猎头，成为国内具有影响力的高端IT专业群
- Mycat社区首次提出BigSQL的概念，并逐步将大数据和实时计算等先进技术引入到Mycat里，从而吸引和聚集了一大批业内大数据和云计算方面的资深工程师，Mycat社区成为名副其实的国内大数据领域实力派成员。
- Mycat社区里不断有优秀工程师被创业公司挖走，为了能更好的支持创业公司并寻求更多的优秀工程师参与采用，Mycat社区目前已经开始开展在线高端IT培训，培养高端Java架构师、工程师。

系统切分及其解决方案

- **何为数据（系统）切分？**

  简单来说，就是指通过某种特定的条件，按照某个维度，将我们存放在同一个数据库中的数据分散存放到多个数据库（主机）上面以达到分散单库（主机）负载的效果。

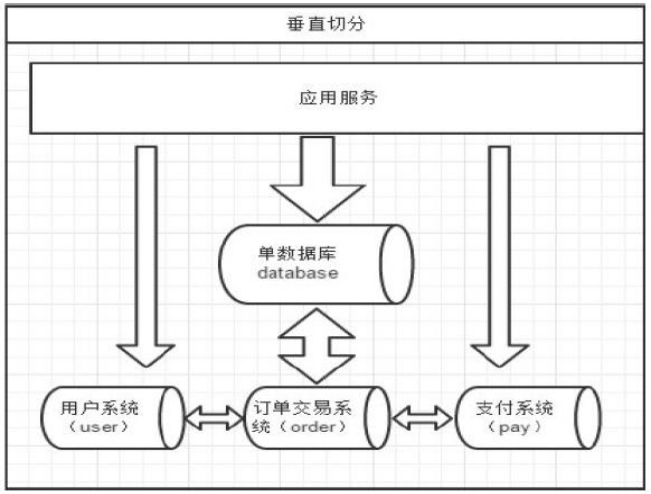
  切分模式：

  a.垂直（纵向）切分。

  B.水平切分。

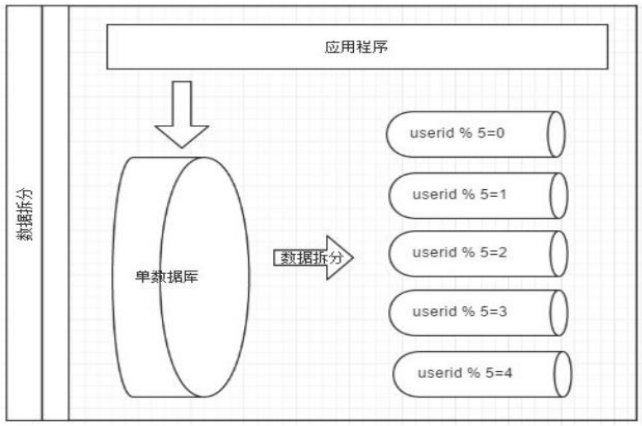
垂直拆分

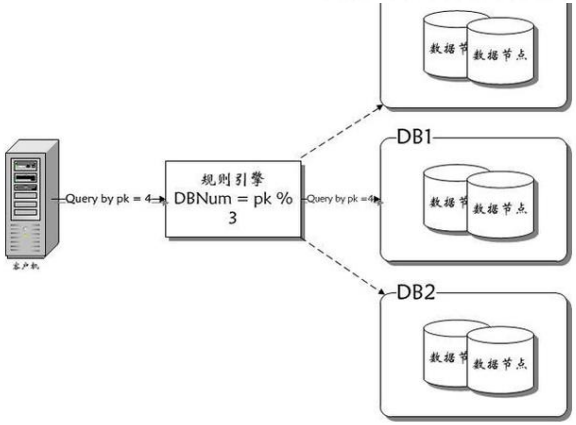
- 一个数据库由很多表的构成，每个表对应着不同的业务，垂直切分是指按照业务将表进行分类，分布到不同的数据库上面，这样也就将数据或者说压力分担到不同的库上面
  - 优点：
    --拆分后业务清晰，拆分规则明确。
    --系统之间整合或扩展容易。
    --数据维护简单。
  - 缺点：
    --部分业务表无法join，只能通过接口方式解决，提高了系统复杂度。
    --受每种业务不同的限制存在单库性能瓶颈，不易数据扩展跟性能提高
    --事务处理复杂。



水平拆分

- 相对于垂直拆分，水平拆分不是将表的数据做分类，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，主要有分表，分库两种模式
  - 优点：
    不存在单库大数据，高并发的性能瓶颈。
    对应用透明，应用端改造较少。
    按照合理拆分规则拆分，join操作基本避免跨库。
    提高了系统的稳定性跟负载能力。
  - 缺点：
    拆分规则难以抽象。
    分片事务一致性难以解决。
    数据多次扩展难度跟维护量极大
    跨库join性能较差





拆分处理的难点

- 垂直切分跟水平切分的不同跟优缺点，会发现每种切分方式都有缺点，但共同的特点缺点有：
  - 引入分布式事务的问题。
  - 跨节点Join 的问题。
  - 跨节点合并排序分页问题。
- 针对数据源管理，目前主要有两种思路：
  - A. 客户端模式，在每个应用程序模块中配置管理自己需要的一个（或者多个）数据源，直接访问各个
    数据库，在模块内完成数据的整合；
    优点：相对简单，无性能损耗。
    缺点：不够通用，数据库连接的处理复杂，对业务不够透明，处理复杂
  - B. 通过中间代理层来统一管理所有的数据源，后端数据库集群对前端应用程序透明；
    优点：通用，对应用透明，改造少。
    缺点：实现难度大，有二次转发性能损失。

拆分原则

- 切分原则：
  - 尽量不切分，架构是进化而来，不是一蹴而就。
  - 最大可能的找到最合适的切分维度。
  - 由于数据库中间件对数据Join 实现的优劣难以把握，而且实现高性能难度极大，业务读取
  - 尽量少使用多表Join
  - 尽量通过数据冗余，分组避免数据垮库多表join。
  - 尽量避免分布式事务。
  - 单表切分数据1000万以内。

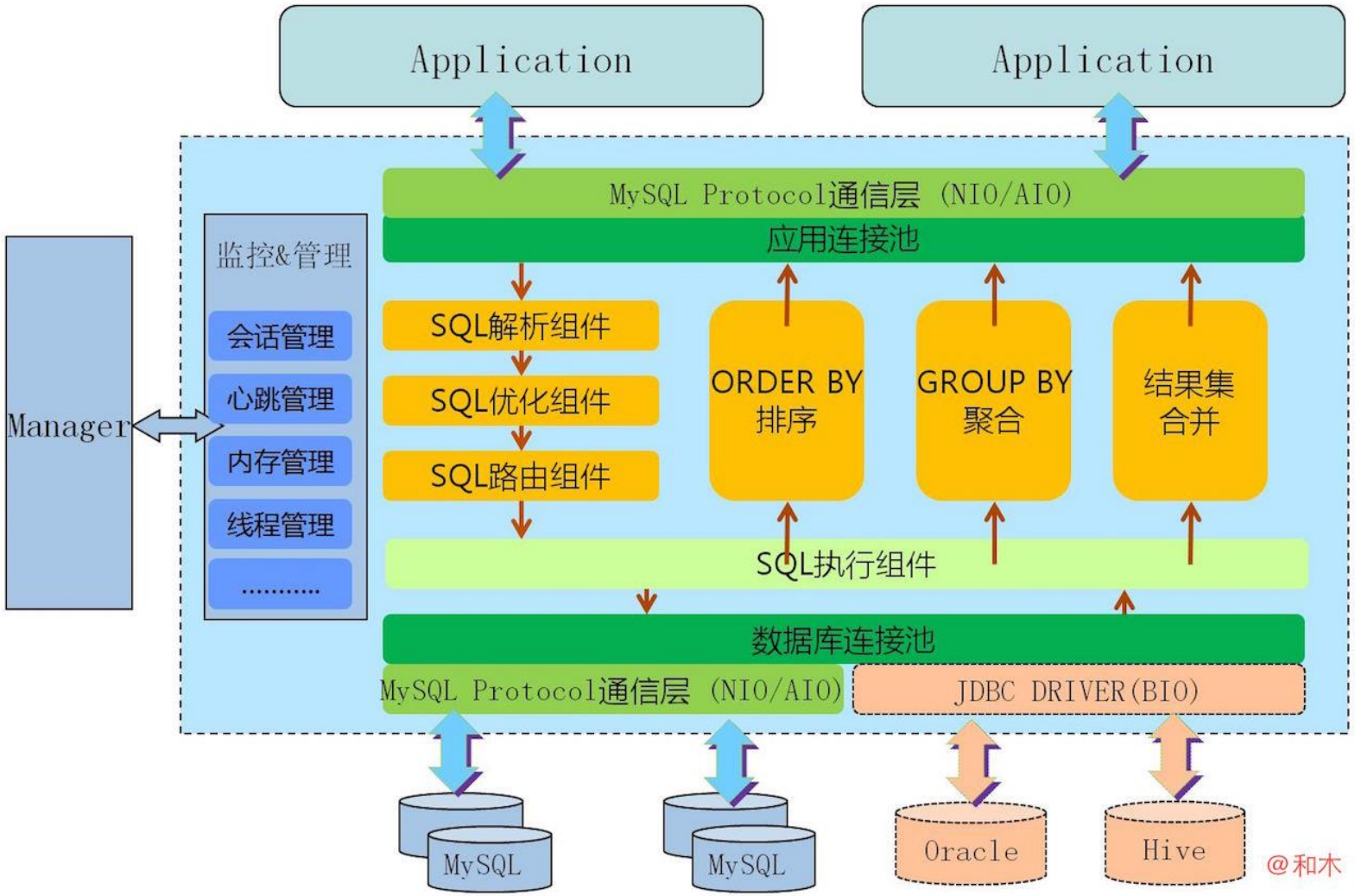
拆分方案

- 360 Atlas
  alibaba cobar
  Mycat
  tddl
  heisenberg
  Oceanus
  vitess
  OneProxy
  drds

Mycat

-遵守Mysql原生协议，跨语言，跨数据库的通用中间件代理。
-基于心跳的自动故障切换，支持读写分离，支持MySQL一双主多从，以及一主多从
-有效管理数据源连接，基于数据分库，而不是分表的模式。
-基于Nio实现，有效管理线程，高并发问题。
-支持数据的多片自动路由与聚合，支持sum,count,max等常用的聚合函数。
-支持2表join，甚至基于caltlet的多表join。
-支持通过全局表，ER关系的分片策略，实现了高效的多表join查询。
-支持多租户方案。
-支持分布式事务（弱xa）
-支持全局序列号，解决分布式下的主键生成问题。
-分片规则丰富，插件化开发，易于扩展。
-强大的web，命令行监控。
-支持前端作为mysq通用代理，后端JDBC方式支持Oracle、DB2、SQL Server 、mongodb 、巨杉。
-集群基于ZooKeeper管理，在线升级，扩容，智能优化，大数据处理（2.0开发版）。

Mycat架构图


Application
Application
MySQL Protocol通信层 (NIO/AIO)
应用连接池
监控&管理
会话管理
心跳管理
内存管理
线程管理
.....
Manager
SQL解析组件
SQL优化组件
SQL路由组件
ORDER BY
排序
GROUP BY
聚合
结果集
合并
SQL执行组件
数据库连接池
MySQL Protocol通信层 (NIO/AIO)
JDBC DRIVER(BIO)
MySQL
MySQL
Oracle
Hive
@和木

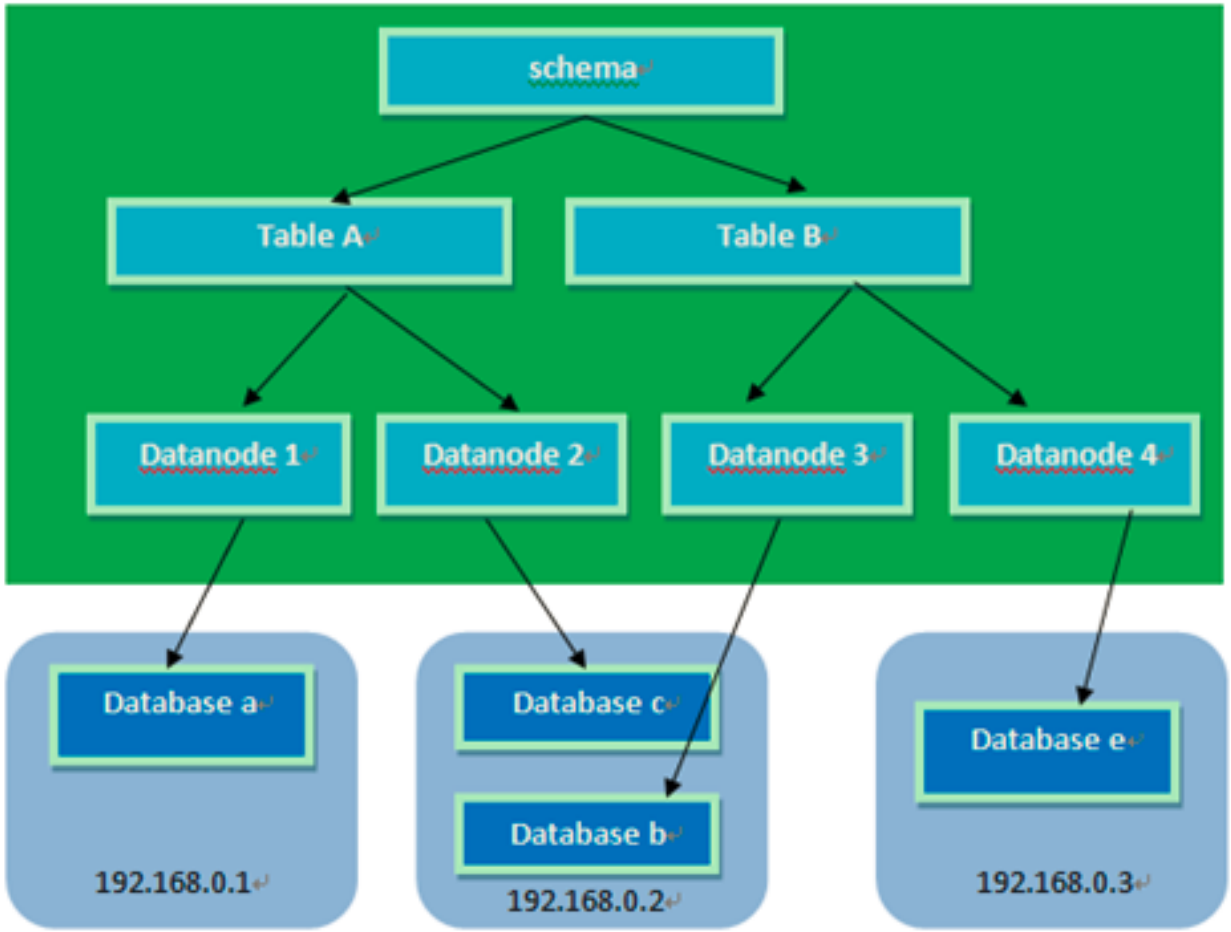
Mycat

逻辑库(schema)
- -逻辑表(table)
  - -- **ER 表**
  - -- 非分片表
  - -- 分片表
  - -- 全局表
- - 分片节点(datanode)
- -节点主机(datahost)
- -分片规则(rule)
- -全局序列号(sequence)
- -多租户

逻辑库与逻辑表

- Mycat抽象多个myql实例，对外提供唯一的访问数据源
- Server.xml
  - <?xml version="1.0" encoding="UTF-8"?>
  - <!DOCTYPE mycat:server SYSTEM "server.dtd">
  - <mycat:server xmlns:mycat="http://io.mycat/">
  - <user name="mycat">
  - <property name="password">mycat</property>
  - <property name="schemas">orderdb</property>
  - </user>
  - </mycat:server>

逻辑库与逻辑表

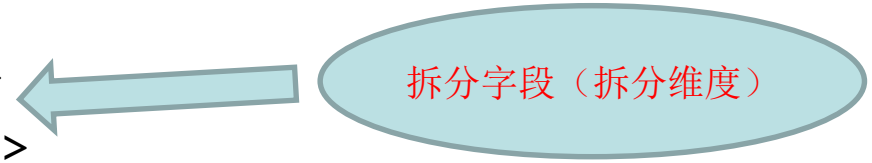

schema
Table A
Table B
Datanode 1
Datanode 2
Datanode 3
Datanode 4
Database a
192.168.0.1
Database c
Database b
192.168.0.2
Database e
192.168.0.3

Schema.xml

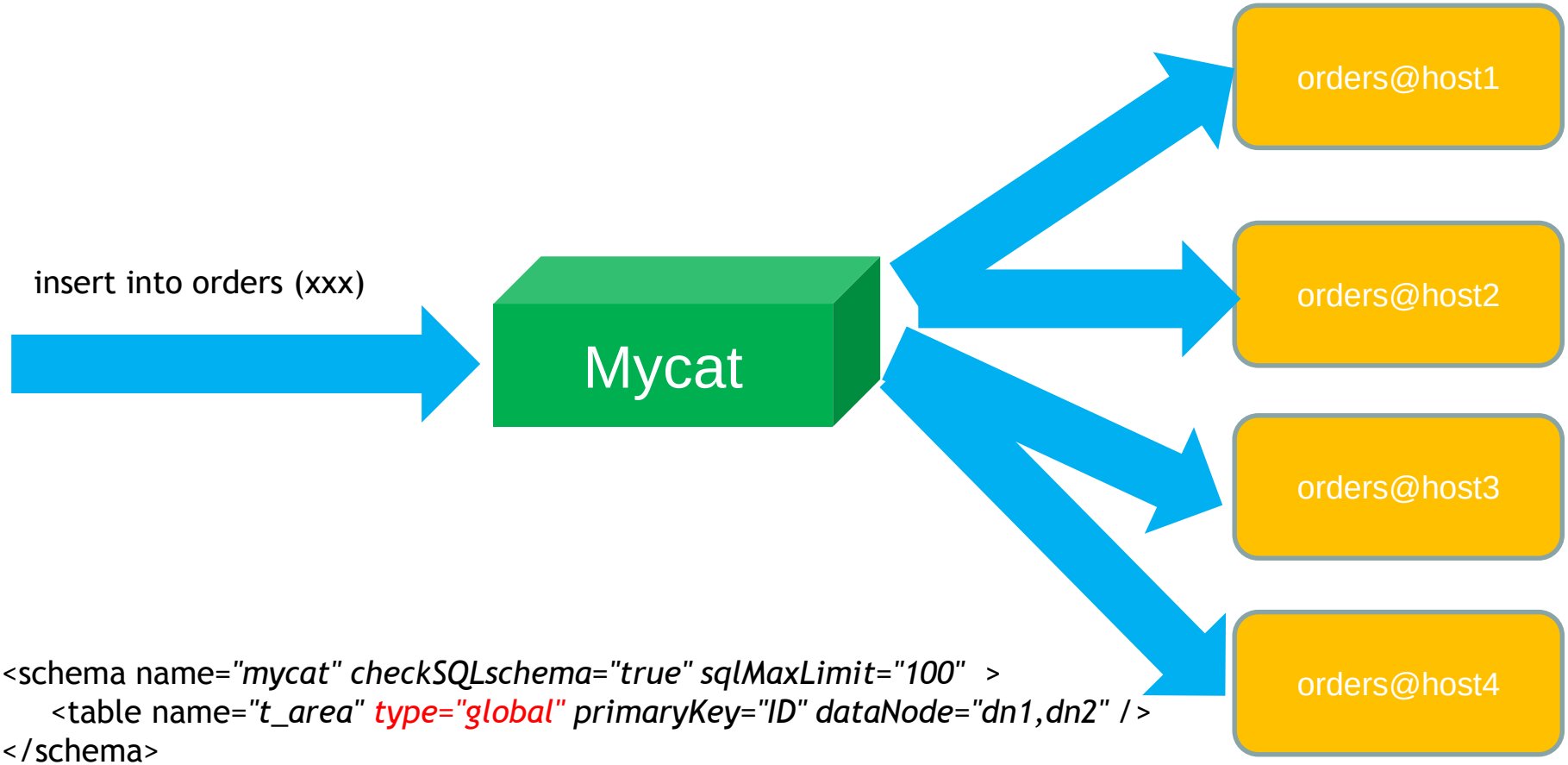
```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
  <schema name="orderdb" checkSQLschema="true" sqlMaxLimit="100" >
    <table name="t_user" dataNode="dn1,dn2" rule="rule1"> </table>
  </schema>
  <dataNode name="dn1" dataHost="192.168.1.101_order_host1" database="order" />
  <dataNode name="dn2" dataHost="192.168.1.101_order_hos2" database="order" />
  <dataHost name="jdbchost" maxCon="2" minCon="1" balance="0"
    writeType="0" dbType="mysql" dbDriver="native">
    <heartbeat>select 1</heartbeat>
    <writeHost host="master" url="121.40.121.133:3306" user="root"
password="zaq1xsw2cde3"></writeHost>
  </dataHost>
  <dataHost name="jdbchost2" maxCon="2" minCon="1" balance="0"
    writeType="0" dbType="mysql" dbDriver="native" >
<heartbeat>select 1</heartbeat>
    <writeHost host="master" url="121.40.121.133:3306" user="root"
password="zaq1xsw2cde3"></writeHost>
  </dataHost>
</mycat:schema>
```

Rule.xml


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mycat:rule SYSTEM "rule.dtd">
<mycat:rule xmlns:mycat="http://io.mycat/">
<tableRule name="rule1">
    <rule>
      <columns>user_id</columns>
      <algorithm>func1</algorithm>
    </rule>
  </tableRule>
   <function name="func1"
class="io.mycat.route.function.PartitionByLong">
    <property name="partitionCount">1</property>
    <property name="partitionLength">1024</property>
  </function>
</mycat:rule>
```



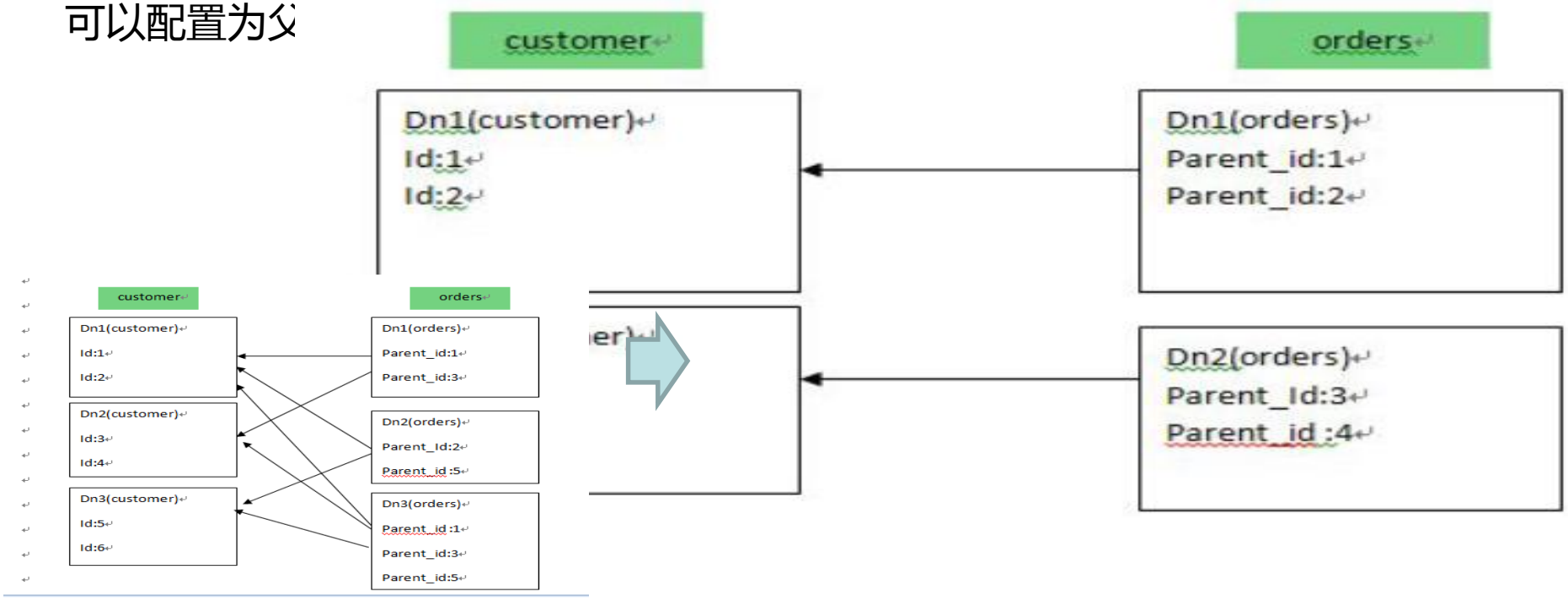
Mycat全局表




```
<schema name="mycat" checkSQLschema="true" sqlMaxLimit="100"  >
    <table name="t_area" type="global" primaryKey="ID" dataNode="dn1,dn2" />
</schema>
```

ER表及ER关系模型

- ER模型就是将表按照相同拆分规则，相同拆分值，切分到同一个分片节点上，避免出现跨分片的join的一种切分模型
可以配置为父



ER表及ER关系模型


```
<table name="t_user" dataNode="dn1,dn2" rule="rule1">
     <childTable name="t_user_class_rel" primaryKey="id" joinKey="user_id"
parentKey="user_id" >
          <childTable name="t_user_class_rel2" primaryKey="id" joinKey="pid" parentKey="id" >
   </childTable>
       </childTable>
 </table> -->
```

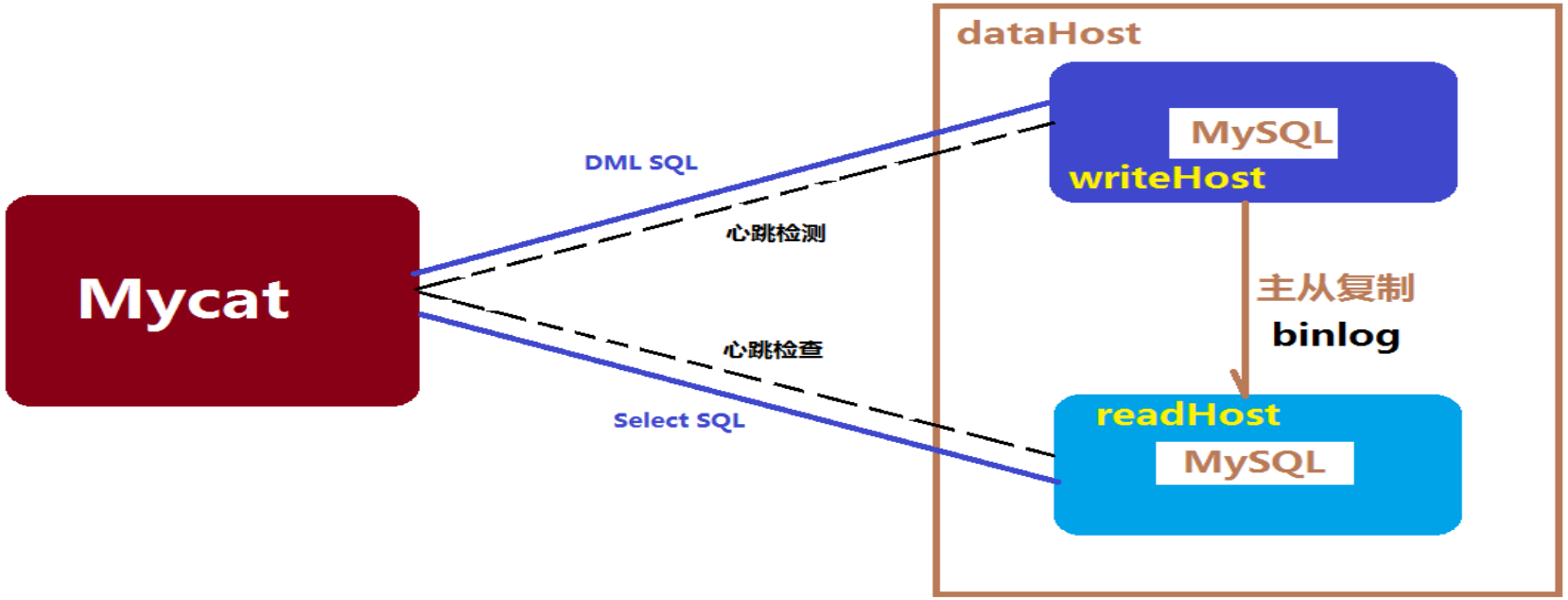
```
<table name="tuser"    primaryKey="vid"  joinKey="vid "      dataNode="dn1,dn2"  rule="rule1" />
<table name="torder"  primaryKey="vid"  joinKey="userid"  dataNode="dn1,dn2" rule="rule1" />
```

Mycat数据join解决方案

- 全局表技术
- 独创的ER关系分片
- 基于Catlet的sharejoin两表自动Join模块
- 复杂SQL可通过用户自定义的Catlet进行处理
- Catlet是Java编写的一段程序，类似数据库中的存储过程，可以实现任意复杂SQL的Join、Group、Order等功能

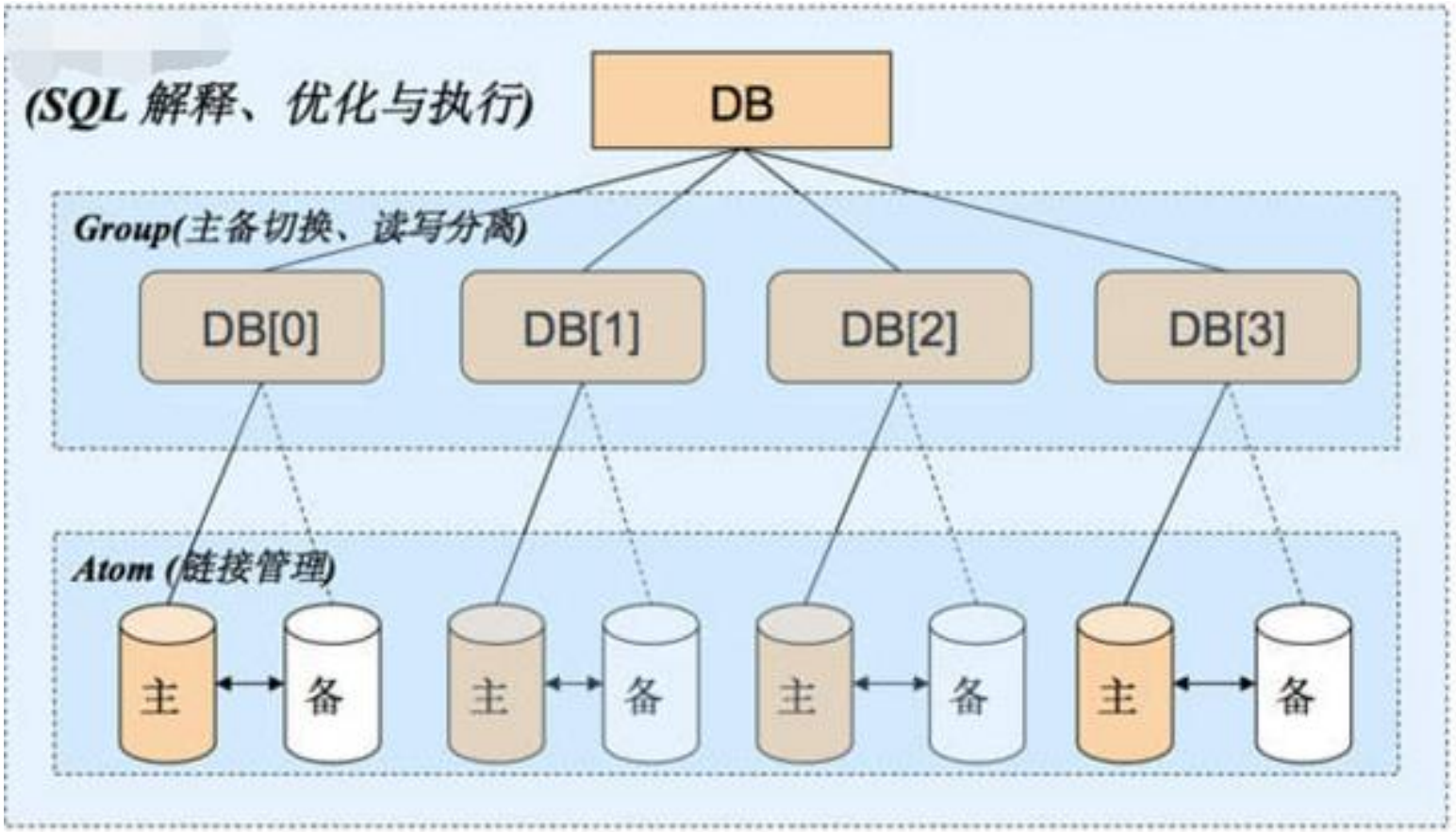
Mycat高可用

- 读写分离和自动切换机制
  - 基于心跳的自动切换
  - Mycat 支持基于MySQL主从复制状态的高级读写分离与主从切换控制机制。
  - 强制读写节点解决读写分离时延问题



Mycat高可用

- 完整架构



Mycat读写分片

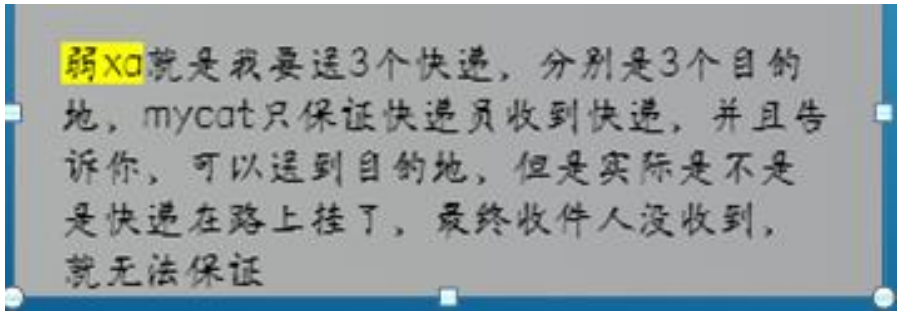
- MyCAT的读写分离机制如下：
  - a.事务内的SQL，默认走写节点，以注释/*balance*/开头，则会根据balance= “1” 或“2”或“3” 去获取 .
  - b.自动提交的select语句会走读节点，并在所有可用读节点中间随机负载均衡,默认根据balance= “1” 或“2”或“3” 去获取，以注释/*balance*/开头则会走写,解决部分已经开启读写分离，但是需要强一致性数据实时获取数据的场景走写
  - c. 当某个主节点宕机，则其全部读节点都不再被使用，因为此时，同步失败，数据已经不是最新的，MYCAT会采用另外一个主节点所对应的全部读节点来实现select负载均衡。
  - d.当所有主节点都失败，则为了系统高可用性，自动提交的所有select语句仍将提交到全部存活的读节点上执行，此时系统的很多页面还是能出来数据，只是用户修改或提交会失败。
- dataHost的balance属性设置为：
  - 0，不开启读写分离机制
  - 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，所有的readHost与writeHost都参与select语句的负载均衡，也就是说，当系统的写操作压力不大的情况下，所有主机都可以承担负载均衡。
  - 3. 全部的读节点参数读，写节点不参与，如果配置了多个writerhost，则多个writerhost下面的readhost参数读负载。

Mycat读写分片

- MyCAT的读写分离的配置如下：
  - <dataHost name="testhost" maxCon="1000" minCon="10" **balance="1"** writeType="0" dbType="mysql" dbDriver="native">
  - <heartbeat>select user()</heartbeat> <!-- can have multi write hosts -->
  - <writeHost host="hostM1" url="localhost:3306" user="root" password="">
  - <readHost host="hostM2" url="10.18.96.133:3306" user="test" password="test" />
  - </writeHost>
  - </dataHost>

Mycat事务

- Mycat对事务的支持为弱xa，如果业务需要严格的强一致性，需要采用事务补偿方案，或者尽量避免跨库操作。
- mycat的事务处理是，依次轮询需要处理的分片执行对应的Sql，当应用commit时在依次轮询commit。
- 如果中间一个分片执行sql出错全部回滚。
- 但是如果应用发起commit命令，Mycat发起commit到db成功，但是实际db未成功时，mycat
- 无法保证各个节点回滚，这就是弱xa。



Mycat事务

- -异步化处理
- -最终一致性

而我们处理事务的方法却和传统应用处理事务的方案不大一样，我们非常强调事务的最终一致性和异步化。利用这种方式，能够极大地降低分布式系统中锁持有的时间，从而极大地提升系统性能。

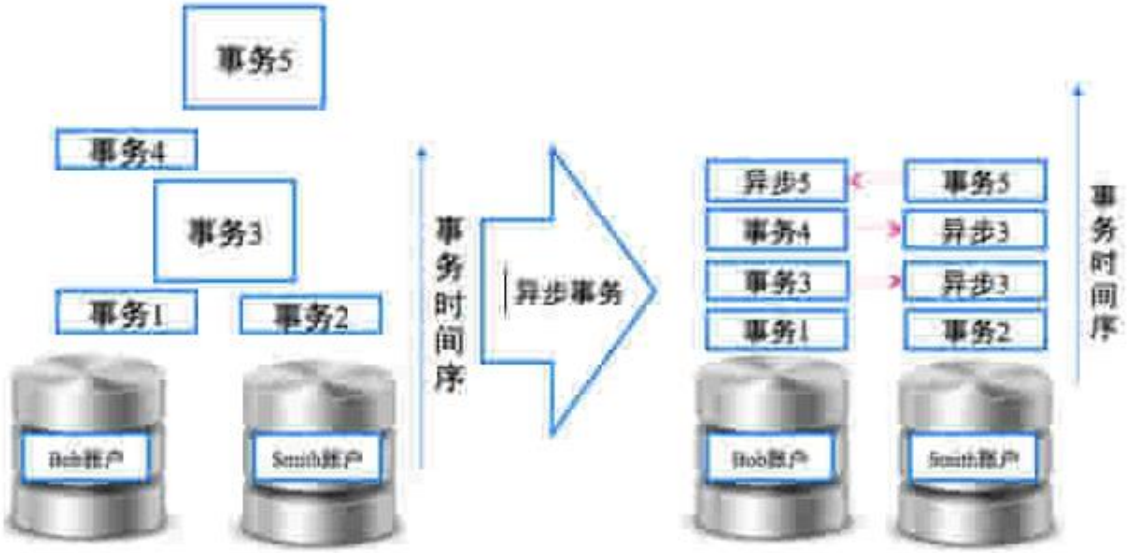
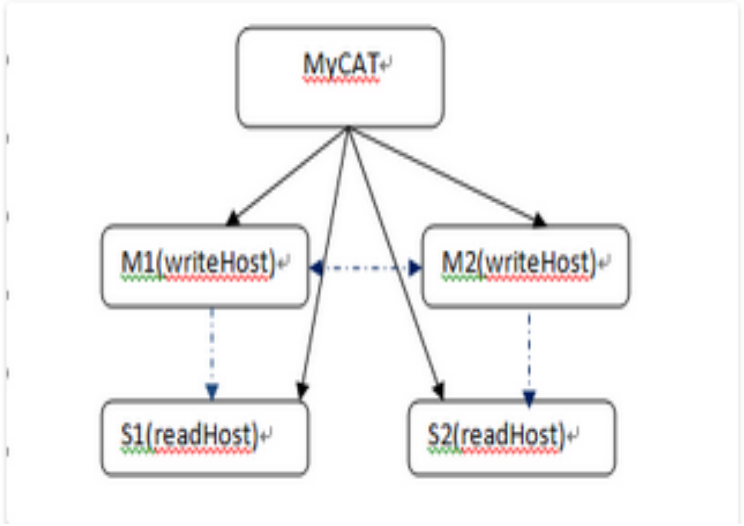


图7 DRDS分布式事务解决套件

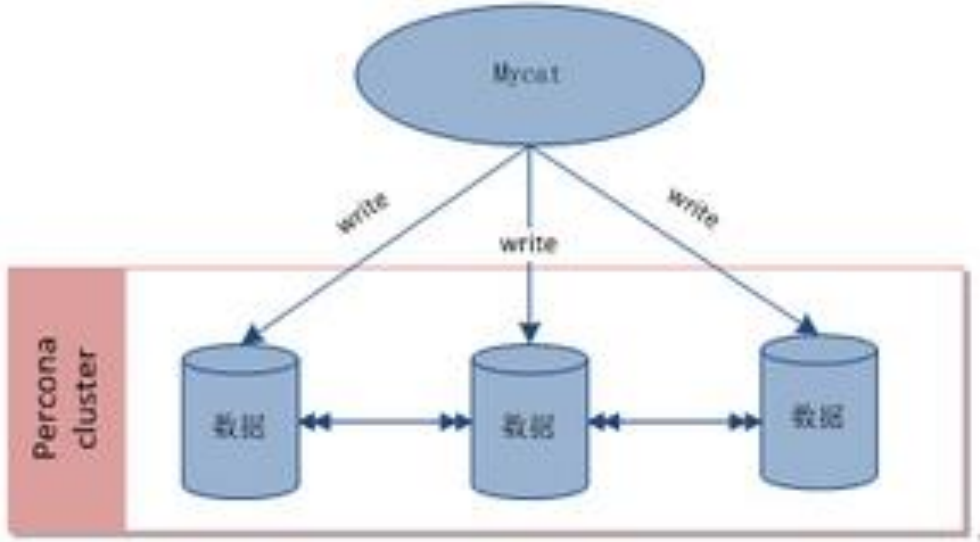
Mycat主从及多主配置

- dataHost的writeType属性设置为：
- writeType=0 默认配置。
- writeType=1 代表配置多主，mycat会往所有写节点，随机写数据，但是每次只会写入一个节点，此模式下无读节点，节点之间开启数据库级别同步。
- 此配置为mysql高级级别使用，因为 多主会带来数据库同步问题。

下面是典型的双主双从的Mysql集群配置：



架构：



Mycat主从及多主配置

- <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0" writeType= “0" dbType="mysql" dbDriver="native">
- <heartbeat>select user()</heartbeat>
- <writeHost host="hostM1" url="localhost:3306" user="root" password="123456" />
- <writeHost host="hostM2" url="localhost:3317" user="root" password="123456" />
- <writeHost host="hostM3" url="localhost:3319" user="root" password="123456" />
- </dataHost>

Mycat全局序列号

- 全局序列号是MyCAT提供的一个新功能，为了实现分库分表情况下，表的主键是全局唯一，而默认的MySQL的自增长主键无法满足这个要求。全局序列号的语法符合标准SQL规范，其格式为：
- next value for MYCATSEQ_GLOBAL
- 其中MYCATSEQ_GLOBAL是序列号的名字，MyCAT自动创建新的序列号，免去了开发的复杂度，另外，MyCAT也提供了一个全局的序列号，名称为：MYCATSEQ_GLOBAL。
- **注意，MYCATSEQ_必须大写才能正确识别。**

- <table name= *"t_node" primaryKey="vid" autoIncrement="true" dataNode="dn1" rule="rule1" />*

- 参见： http://songwie.com/articlelist/68

Mycat多数据库支持

More….

Mycat引擎

- MySql的引擎 InnoDB与MyISAM
- MyISAM 是MySQL中默认的存储引擎，一般来说不是有太多人关心这个东西。但是面对实际的业务问题，这里需要思考一番

Mycat引擎

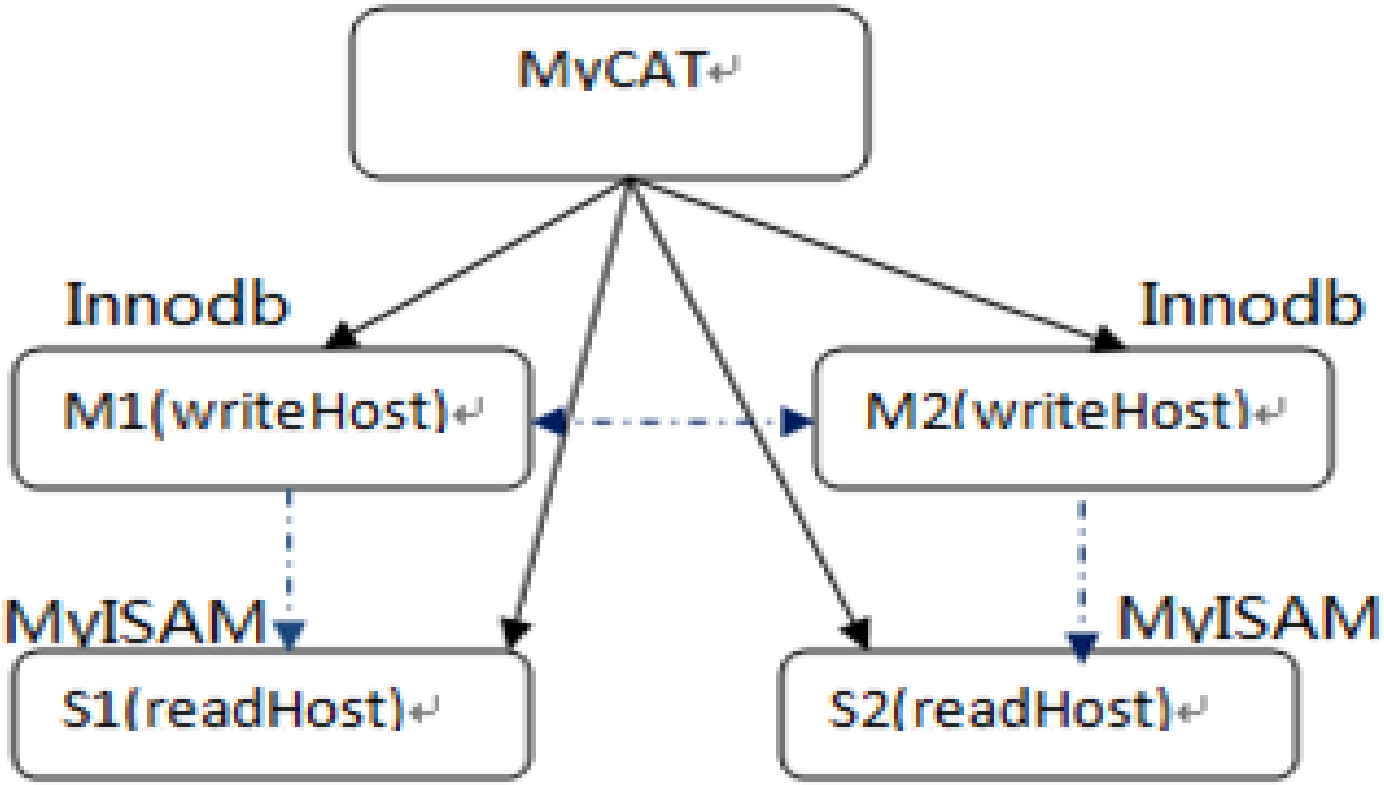
- 你需要事务支持吗？
- 你需要全文索引吗？
- 你经常使用什么样的查询模式?

Mycat引擎

- MyISAM管理非事务表。它提供高速存储和检索，以及全文搜索能力。如果应用中需要执行大量的SELECT查询，那么MyISAM是更好的选择。
- InnoDB用于事务处理应用程序，具有众多特性，包括ACID事务支持。如果应用中需要执行大量的INSERT或UPDATE操作，则应该使用InnoDB，这样可以提高多用户并发操作的性能。

Mycat引擎

- 核心思想：利用MyCat的读写分离模式，写节点使用MySql的InnoDB引擎，读节点使用的是MySql的MyISAM引擎。



Mycat和海量关系型数据读写(20:30)

2月26日脱产班，**3月12日**周末班，**3月19日**网络班

欢迎你的加入！

需要代码、PPT、视频资料请加下咨询老师QQ：

贾老师：1786418286

何老师：1926106490

詹老师：2805048645

讨论技术可以加入QQ群：172599077,156927834