**MyCat：开源分布式数据库中间件**

参考：<https://www.csdn.net/article/2015-07-16/2825228>

**摘要：**MyCat截至到2015年4月，保守估计已经有超过60个项目在使用，主要应用在电信领域、互联网项目，大部分是交易和管理系统，少量是信息系统。比较大的系统中，数据规模单表单月30亿。本文带你全面了解MyCat。

**为什么需要MyCat？**

虽然云计算时代，传统数据库存在着先天性的弊端，但是NoSQL数据库又无法将其替代。如果传统数据易于扩展，可切分，就可以避免单机（单库）的性能缺陷。

MyCat的目标就是：低成本地将现有的单机数据库和应用平滑迁移到“云”端，解决数据存储和业务规模迅速增长情况下的数据瓶颈问题。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。

MyCat截至到2015年4月，保守估计已经有超过60个项目在使用，主要应用在电信领域、互联网项目，大部分是交易和管理系统，少量是信息系统。比较大的系统中，数据规模单表单月30亿。

**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协议的服务器，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。

MyCat发展到目前的版本，已经不是一个单纯的MySQL代理了，它的后端可以支持MySQL、SQL Server、Oracle、DB2、PostgreSQL等主流数据库，也支持MongoDB这种新型NoSQL方式的存储，未来还会支持更多类型的存储。而在最终用户看来，无论是那种存储方式，在MyCat里，都是一个传统的数据库表，支持标准的SQL语句进行数据的操作，这样一来，对前端业务系统来说，可以大幅降低开发难度，提升开发速度

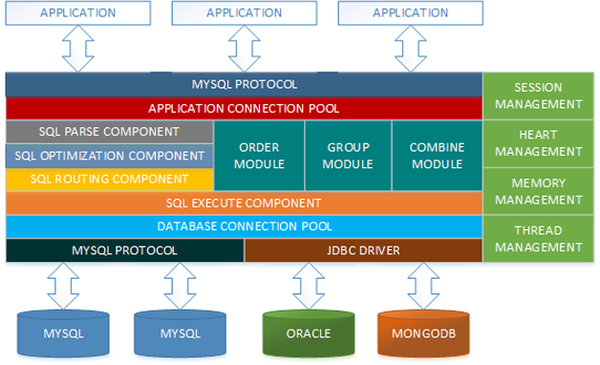


图1 MyCat架构设计图

**MyCat解决了哪些问题**

**1. 连接过多问题，可以通过MyCat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。**使用MyCat之前系统结构如图2。

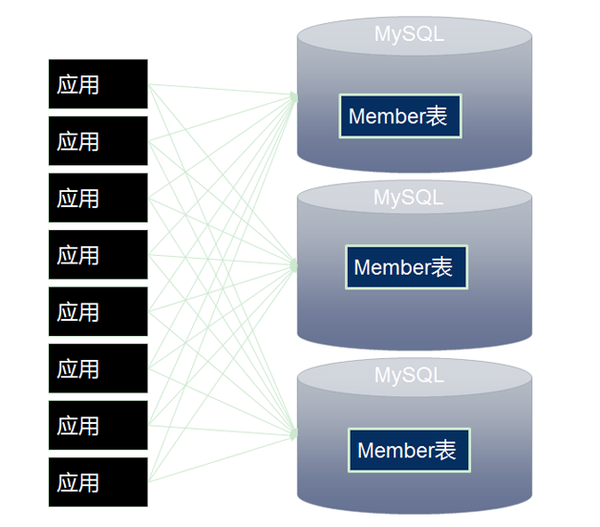


图2  MyCat早前系统架构

MyCat引入连接复用解决多应用竞争问题，通过MyCat改造后，如图3所示。

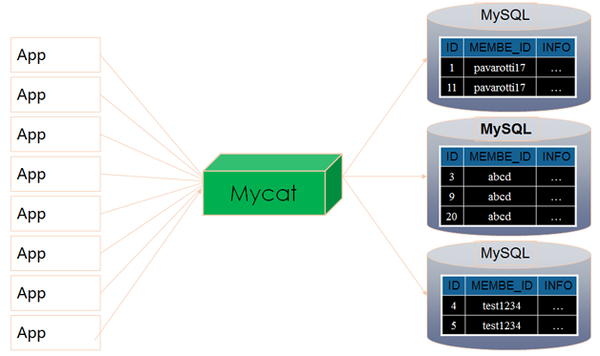


图3  改造后的MyCat

**2. 独创的ER关系分片，解决E-R分片难处理问题**，存在关联关系的父子表在数据插入的过程中，子表会被MyCat路由到其相关父表记录的节点上，从而父子表的Join查询可以下推到各个数据库节点上完成，这是最高效的跨节点Join处理技术（me:其实并不是真实的跨节点，而是在每个节点上进行连接查询然后在汇总结果），也是MyCat首创。

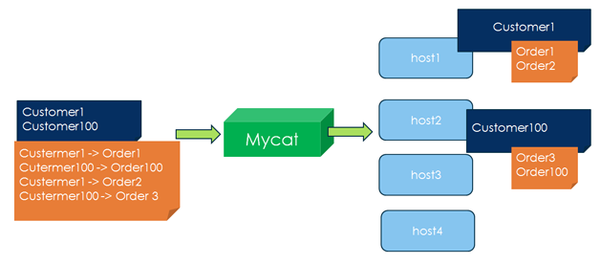


图4  独创的ER关系分片，是MyCat首创

**3. 采用全局分片技术，每个节点同时并发插入和更新数据，每个节点都可以读取数据(me:如果上线先数据库实例就会出现问题)**，提升读性能的同时，也解决跨节点Join的效率。

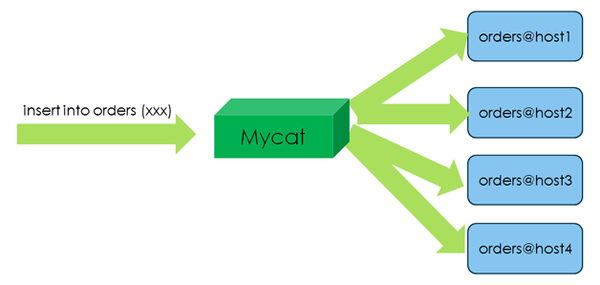


图5 采用全局分片技术

**4. 通过人工智能的catlet支持跨分片复杂SQL实现以及存储过程支持等。**使用方式主要通过MyCat注释的方式来执行，如下：

（1）跨分片联合查询注解支持：

/\*!MyCat:catlet=demo.catlets.ShareJoin / select bu. ,sg.\* from base\_user bu,sam\_glucose sg where bu.id\_=sg.user\_id；

注：sam\_glucose是跨分片表。

（2）存储过程注解支持：

/\*!MyCat: sql=select \* from base\_user where id\_=1;\*/ CALL proc\_test();

注：目前执行存储过程通过MyCat注解的方式执行，注意需要把存储过程中的sql写到注解中。

（3）批量插入与ID自增长结合的支持：

/\*!MyCat:catlet=demo.catlets.BatchInsertSequence \*/ insert into sam\_test(name\_) values(‘t1’),(‘t2’);

注：此方式不需要在sql语句中显示的设置主键字段，程序在后台根据primaryKey配置的主键列，自动生成主键的sequence值并替换原sql中相关的列和值；

（4）获取批量sequence值的支持：

/\*!MyCat:catlet=demo.catlets.BatchGetSequence \*/SELECT MyCat\_get\_seq(‘MyCat\_TEST’,100);

注：此方法表示获取MyCat\_TEST表的100个sequence值，例如当前MyCat\_TEST表的最大sequence值为5000，则通过此方式返回的是5001，同时更新数据库中的MyCat\_TEST表的最大sequence值为5100。

（5）更好地支持数据库读写分离与高可用性，MyCat支持基于MySQL主从复制状态的高级读写分离控制机制（比如Slave\_behind\_master <100则开启），而一旦检测到主从同步出错或者延时超过发展，则自动排除readHost，防止程序读到很久的旧数据。

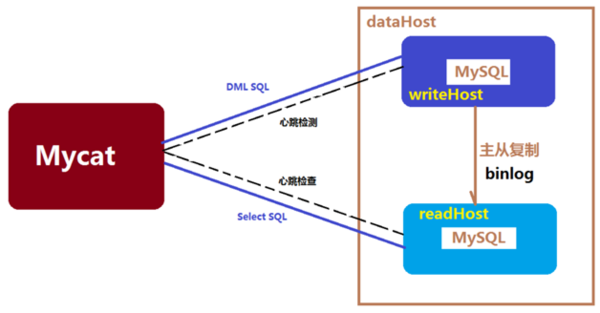


图6  Mycat支持基于MySQL主从复制状态的高级读写分离控制机制

**MyCat技术原理**

MyCat技术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（me:即作为后端各个服务库的代理，使得前端应用对多实例数据库透明）。

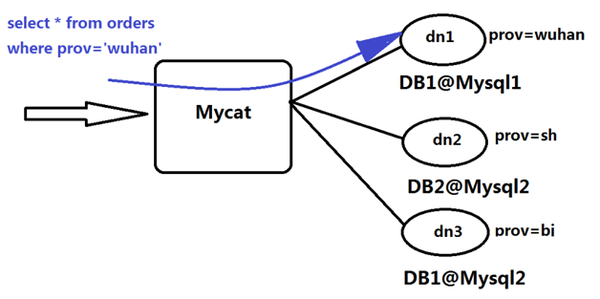


图7  Orders被分为三个分片datanode

如图7所示，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。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、HBT（Human Brain Tech)人工智能的Catlet等。

**MyCat下一步规划**

强化分布式数据库中间件的面的功能，使之具备丰富的插件、强大的数据库智能优化功能、全面的系统监控能力、以及方便的数据运维工具，实现在线数据扩容、迁移等高级功能。

进一步挺进大数据计算领域，深度结合Spark Stream和Storm等分布式实时流引擎，能够完成快速的巨表关联、排序、分组聚合等OLAP方向的能力，并集成一些热门常用的实时分析算法，让工程师以及DBA们更容易用MyCat实现一些高级数据分析处理功能。