Broadview®
www.broadview.com.cn

MyCAT

分布式数据库架构及企业实践

基于Mycat中间件

/ 周继锋 冯钻优 陈胜尊 左越宗 著 /

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

分布式数据库架构及企业实践

基于Mycat中间件

/ 周继锋 冯钻优 陈胜尊 左越宗 著 /

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书由资深 Mycat 专家及一线架构师、DBA 编写而成。全书总计 8 章，首先简单介绍了分布式系统和分布式数据库的需求，然后讲解了分布式数据库的实现原理，并对市场上存在的各种分布式数据库中间件进行了对比，再围绕着如何利用 Mycat 实现分布式数据库而展开。本书对 Mycat 从入门到进阶、从高级技术实践到架构剖析、从网络通信协议解析到系统工作原理的方方面面进行了详细讲解，并剖析了 Mycat 的 SQL 路由、跨库联合查询、分布式事务及原生 MySQL、PostgreSQL 协议等核心技术。通过本书不仅可以了解 Mycat 的基本概念，掌握 Mycat 配置等技术，还能感受到 Mycat 的架构设计之美，了解 Mycat 2.0 的未来规划。

无论是对于软件工程师、测试工程师、运维工程师、软件架构师、技术经理，还是对于资深 IT 人士来说，本书都极具参考价值。

图书在版编目（CIP）数据

分布式数据库架构及企业实践：基于 Mycat 中间件 / 周继锋等著. —北京：电子工业出版社，2016.11

ISBN 978-7-121-30287-9

Ⅰ. ①分… Ⅱ. ①周… Ⅲ. ①分布式数据库－数据库系统 Ⅳ. ①TP311.133.1

中国版本图书馆 CIP 数据核字（2016）第 268198 号

策划编辑：张国霞
责任编辑：徐津平
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×980　1/16　印张：20　　字数：450 千字
版　　次：2016 年 11 月第 1 版
印　　次：2016 年 11 月第 1 次印刷
印　　数：3000 册　　定价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819　faq@phei.com.cn。

推荐序 1

随着大数据时代的到来，海量数据存储、并行计算、异构数据互联等一系列新技术在市场上不断地涌现。相信数据库行业的很多从业者都对传统关系型数据库的单点故障及容量问题头疼不已，而“分库分表”也早已成为解决这类问题的基础，此时，Mycat 应运而生。

Mycat 是一款面向企业级应用的开源数据库中间件产品，它目前支持数据库集群、分布式事务与 ACID，被普遍视为基于 MySQL 技术的集群分布式数据库解决方案，在一些互联网、金融、运营商客户中用来替代昂贵的 Oracle。

Mycat 不仅可以轻松对接 MySQL、SQL Server 等传统关系型数据库，也融合了内存缓存、NoSQL、HDFS 等新兴大数据技术，是一款非常优秀的数据库中间件。

在如今的大数据时代，分布式架构已经成为企业级数据应用的标配，传统的关系型数据库产品已经面临一个真正的拐点：一方面，关系型数据库自身难以实现分布式，这大大限制了其数据存储能力及整体的性能表现；另一方面，商业化的传统数据库产品的成本和性价比在分布式架构崛起的状况下毫无优势可言。因此，无论是从底层全新实现分布式计算存储的 NoSQL、Hadoop，还是使用 Mycat 这样的分库分表工具，对关系型数据库大刀阔斧地进行“改装”都是大势所趋。

作为一名专注于数据库领域多年的从业者，我认为 Mycat 从中间件工具的角度成功地弥补了 MySQL 的诸多局限。

- 分布式存储：通过 Mycat，MySQL 可以实现集群化与分布式管理，使数据库容量与处理能力大大改善。
- 性能加速：通过分布式集群及 Mycat Booster 对 MySQL 数据库在集群环境下的加速，Mycat 大大提升了 MySQL 集群的性能。
- 异构数据互联互通：除了 MySQL，Mycat 同时支持如 SequoiaDB、MongoDB 这样的 NoSQL 数据库及 HDFS 分布式文件系统，实现了对非结构化数据、半结构化数据及结

构化数据的存储及互联。

- 多样化的数据库工具：Mycat 为用户提供了丰富的管理工具，可以帮助用户更好地管理数据库系统。

本书非常适合作为 Mycat 的入门及进阶参考读物，它非常全面地阐述了分库分表的基本原理、实现机制及实践经验。本书的作者有着丰富的行业经验及技术底蕴，能够把业界非常前沿的知识用深入浅出的语言传授给各位读者。

最后，作为 SequoiaDB 的联合创始人，我十分钦佩 Mycat 团队的技术及勇气。虽然基础软件的开发难度很大，但是我们都敢于去挑战一个个技术难点，并填补国内基础软件产品的巨大空白。因此，我在这里衷心地祝愿 Mycat 前程似锦！

巨杉数据库联合创始人　王涛

推荐序 2

随着分布式系统的发展，应用的分布式由于无状态的特性，可以利用消息机制相对简单地进行拆分，计算的分布式也可以通过 Map、Reduce 等相关算法来解决。但是随着业务压力和并发压力的增加，我们急需一种分布式数据库解决方案来支持数据库的水平扩展，通过简单地增加服务器及线性地提升数据库的并发访问能力，为闯过分布式系统的最后一道难关铺平道路。

从阿里巴巴的 Cobar 到开源社区的 Mycat，从 Cobar 的架构师贺贤懋、朱海清、邱硕到 Mycat 的核心人员南哥、冰风影，作为一名专注于 MySQL 数据库十多年的从业者，我见证了分布式数据库的从无到有到百花齐放，在收到本书的序言邀请时，我感到非常荣幸。

现在的分布式数据库产品越来越多。YouTube 公司提供的 Vitness 功能强大，在 YouTube 的生产环境下支撑了大量的业务访问；360 公司的 Altas 基于 MySQL Proxy 开发而成，最初主要在应用层进行透明的读写分离，于 2013 年引入了分库分表；陈菲在离开 360 公司后在 WPS 云平台用 Go 语言编写了 Kingshard；楼方鑫（黄忠）在离开支付宝后编写了 OneProxy；腾讯互娱的 DBA 团队基于 Spider 打造了自己的分布式数据库平台；淘宝在内部将 TDDL 的客户端工具作为了分库分表中间件；阿里巴巴的 B2B 开源了支撑其内部业务生产环境 3 年的 Cobar，为开源社区提供了一大助力；而基于 Cobar 开发的 Mycat 及其各种分支由于其易用性，将分布式数据库进一步推广到互联网和传统行业的各个业务领域。

Mycat 无疑是这些中间件中的佼佼者，支持百亿级别的数据分片和并行计算，支持高可用和 MySQL 的读写分离，并随着版本的更新进一步支持 Oracle、DB2、MongoDB 等后端数据库，随着周边产品的进一步成熟，在越来越多的分布式或者非分布式（仅用它的读写分离或者高可用）生产环境中得到部署，受到越来越多的企业的关注。本书恰逢其会，由 Mycat 核心开发人员撰写而成，详细讲述了 Mycat 的由来、架构特点、核心模块、实际使用案例和企业实践，是一本不可多得的好书。

沃趣科技 MySQL 负责人　李春

推荐序 3

作为国产开源数据库中间件——Mycat 的发起者，我不得不为本书作序。

这是一本由众多技术精英合著的数据库+中间件领域的专业书籍，这些人包括 Mycat Commiter、Mycat 志愿者及资深 DBA，大家在工作之余抽出大量时间来编写和完善此书，历经一年完成了本书的编写工作，实属不易。

数据库中间件是新兴的重要的互联网中间件，目前业界仍然缺乏一本系统性介绍相关领域的软件产品、常用技术、架构等的纸质书籍。本书围绕 Mycat 开源中间件，从基础入门到架构原理，从运行机制到源码实现，从系统运维到应用实践，讲解得详尽而又完善。本书内容丰富、图文并茂、由浅入深，对数据库中间件的基本原理阐述清晰，对程序源码分析透彻，对实践经验讲解深刻。

从内容上讲，本书从一个使用者的角度去理解、分析和解决问题，通过大量的实例操作和源码解析，帮助读者深入理解 Mycat 的各种概念。读者对其中的案例只要稍做修改，再结合实际的业务需求，就可以在正在开发的项目中应用，达到事半功倍的效果。并且，通过学习书中应用实战方面的内容，不仅可以直接提高开发技能，还可以解决在实践过程中经常遇到的各种关键问题。另外，本书中的所有观点和经验均是作者们在多年建设、维护大型应用系统的过程中积累形成的，非常值得借鉴和推广。

希望 Mycat 在大家的支持下走得更远，成为中国软件的骄傲。

Leader-us

前　言

随着移动互联网的兴起和大数据的蓬勃发展，系统的数据量正呈几何倍数增长，系统的压力也越来越大，这时最容易出现的问题就是服务器繁忙，我们可以通过增加服务器及改造系统来缓解压力，然后采用负载均衡、动静分离、缓存系统来提高系统的吞吐量。然而，当数据量的增长达到一定程度的时候，增加应用服务器并不能明显地提高系统的效率，因为所有压力都会传导到数据库层面，而大多数系统都是用一个数据库来存储和管理系统数据的。这时，Mycat 应运而生。

谈到 Mycat 就不得不谈谈 Cobar，Cobar 是阿里巴巴开源的数据库中间件，由于其存在使用限制及一些比较严重的问题，Leader-us 在其基础上于 2013 年年底实现了 Mycat 1.0 版本，Mycat 一经发布便引起了很多人的关注。之后 Mycat 社区对 Cobar 的代码进行了彻底重构，使用 NIO 重构了网络模块，并且优化了 Buffer 内核，增强了聚合、Join 等基本特性，同时兼容了绝大多数数据库，使之成为通用的数据库中间件。Mycat 在 1.4 版本以后完全脱离了 Cobar 内核，同时采用了 Mycat 集群管理、自动扩容及智能优化，成为了高性能的数据库中间件。Mycat 从诞生至今已有三年多了，一直在坚持做最好的开源数据库中间件产品。

本书总计 8 章，涵盖了 Mycat 入门、进阶、高级技术实战、企业运维、架构剖析、核心技术分析、多数据库支持原理与实现等内容，内容详尽、图文并茂，几乎囊括了 Mycat 所涉及的方方面面，无论是对于软件工程师、测试工程师、运维工程师、软件架构师、技术经理，还是对于资深 IT 人士来说，本书都极具参考价值。

第 1 章：介绍了分布式系统和分布式数据库系统的原理，介绍 Mycat 的起源和发展状况，并对各种数据库中间件做了简要介绍和对比。

第 2 章：讲解了 Mycat 的入门知识，介绍了 Mycat 的安装环境、核心概念和分库分表的原理，以及 Mycat 源码开发调试的过程。

第 3 章：讲解了 Mycat 的进阶知识，主要介绍 Mycat 的各种配置和分片算法。

第 4 章：讲解了 Mycat 和 MySQL 实战案例，由拥有丰富的 Mycat 线上实战经验的专家和 DBA 共同编写而成，有很高的参考价值。

第 5 章：简要介绍了用于 Mycat 性能监控的工具——Mycat-web，详细讲解了 Mycat 和 MySQL 的优化技术，是 DBA 的亲身总结和经验之谈。

第 6 章：重点阐述了 Mycat 的架构，包括网络、线程、连接池、内存管理及缓存实现等，是了解 Mycat 框架的基础。

第 7 章：介绍了 Mycat 的核心技术，包括分布式事务的实现、跨库 Join 的三种实现方式等，介绍了多节点数据汇聚和排序的原理，并详细阐述了在 Mycat 1.6 版本中实现的一致性分布式事务的功能。

第 8 章：介绍了 MySQL 和 PostgreSQL 的通信协议及 Mycat 对这些通信协议的实现，然后介绍了 Mycat 对 JDBC 及多种数据库的支持，例如 Oracle、SQL Server、MongoDB 等。

本书的编写和校对历经一年，参与编写的作者都是 Mycat 开源项目中参与度比较高、提交过不少代码或有丰富的实战经验的资深人士。非常感谢参与本书编写、指导或校对的专家：Leader-us、南哥（曹宗南）、从零开始（宋伟）、小张哥（张超）、yuanfang（杨鹏飞）、顽石神（张治春）、冰麒麟（杨峰）、望舒（胡雅辉）、明明 Ben（朱阿明）、零（章爱国）、little-pan（潘自朋）、CrazyPig（陈建欣）、毛茸茸的逻辑（王成瑞）、海王星（林志强）、石头狮子（林晁）、HanSenJ（姬文刚）、武（王灯武）、战狼（刘胡波）、KK（刘军）、董海雄（易班网）、arx（李秋伟）、正能量（王金剑）、吉光（李伟）。

由于作者的写作水平有限，书中难免会有不妥或者疏漏之处，欢迎读者批评指正。

冰风影

Mycat 社区负责人

2016 年 11 月 6 日于广州番禺

目　录

第 5 章
Mycat 企业运维

数据库性能优化的重要性已经不仅仅是 DBA、运维人员所关心的了，更是广大开发人员所关注的。在数据量少且并发量不高的情况下，很多问题体现不出来，隐藏的问题也难以被发现，通过监控能为数据库性能优化提供许多参考依据。

5.1 Mycat 性能监控——Mycat-web 详解

5.1.1 Mycat-web 简介

Mycat-web 是 Mycat-server 可视化运维的管理和监控平台，弥补了 Mycat-server 在监控上的空白，是由 rainbow 和冰风影等人主导的一个开源项目，主要目的是为 Mycat-server 分担统计任务和配置管理任务等，使得 Mycat-server 更专注于数据服务。Mycat-web 引入了 ZooKeeper 作为配置中心，可以管理多个节点。Mycat-web 主要管理和监控 Mycat 的流量、连接、活动线程和内存等，具备 IP 白名单、邮件告警等模块，还可以统计 SQL 并分析慢 SQL 和高频 SQL 等，为 SQL 优化提供了重要的参考依据。

Mycat-web 支持 Windows 和 Linux 版本，用户可以通过 GitHub 的 Mycat-Download 下载 Mycat-web 安装文件，也可以下载源码自己编译。Mycat-web 目前已更新至较为成熟的 1.0 版本，下面以其 1.0 版本为基础进行介绍。

在安装 Mycat-web 前需要先安装 ZooKeeper，Mycat-web 的安装步骤如下。

1. 安装 ZooKeeper

（1）下载并解压 ZooKeeper 安装包。

（2）将 zookeeper conf 目录下的 zoo_sample.cfg 重命名为 zoo.cfg。

（3）启动 ZooKeeper。在 Windows 环境下的命令为 bin\zkServer.bat，在 Linux 环境下的命令为 bin\zkServer.sh start。

2. 安装 Mycat-web

（1）下载并安装 Mycat-web。

（2）启动 Mycat-web，之前需要确保 ZooKeeper 已经启动，启动命令为 sh start.sh。启动完成后即可访问 Mycat-web，默认地址是 http://localhost:8082/mycat/。

5.1.2 Mycat-web 的配置和使用

1. 连接 ZooKeeper

初次使用 Mycat-web 时需要配置 ZooKeeper 的地址，有以下两种方法。

（1）访问 http://localhost:8082/mycat/，在注册中心填写 ZooKeeper 的 IP 地址和端口即可，第一次访问时才需要进行配置。

（2）修改 mycat.properties 文件，填写格式为 zookeeper=IP:2181

2. 连接 Mycat

访问管理界面 http://localhost:8082/mycat/，选择 Mycat 配置→Mycat 服务管理菜单，单击“新增”按钮，依次填写 Mycat 的连接信息，如图 5-1 所示。

Mycat 的名称可以自由定义，管理端口默认为 9066，服务端口默认为 8066，数据库的名称为 Mycat 的 schema 的名称。

3. Mycat-VM 管理

Mycat-VM 管理是一个基于 JMX 的图形监控工具，用于连接正在运行的 JVM，以图表化的形式显示各种数据，并可通过远程连接监视远程服务器的 VM 情况，可以较直观地观察各种变化，如图 5-2 所示。

图 5-1

图 5-2

5.1.3　Mycat 性能监控指标

Mycat 性能监控涉及对 Mycat 的流量、连接、活动线程、缓冲队列、MycatTPS 和内存的分析和监控等，如图 5-3 所示。

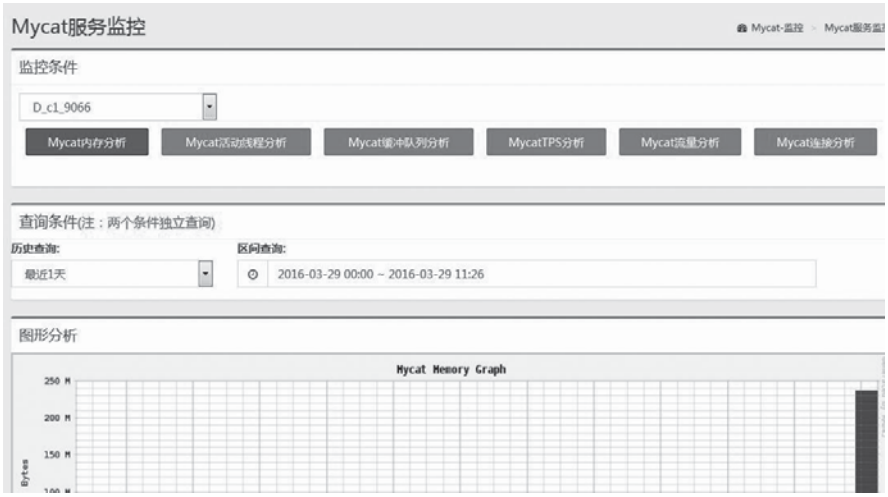

图 5-3

（1）Mycat 内存分析反映了当前的内存使用情况及历史时间段的峰值、平均值。

（2）Mycat 活动线程分析反映了 Mycat 线程的活动情况。

（3）Mycat 流量分析统计了历史时间段的流量峰值、当前值、平均值，是 Mycat 数据传输的重要指标。

（4）Mycat 连接分析反映了 Mycat 的连接数。

（5）MycatTPS 是 LoadRunner 中重要的性能参数指标，为系统在每秒内能够处理的交易或事务的数量。MycatTPS 分析统计了 Mycat 在单位时间内处理的交易或事务的数量，是衡量 Mycat 处理能力的重要指标。

Mycat-web 的 SQL 监控以用户为单位统计读写次数、读占比、最大并发数、分段请求数和耗时请求数，以时间为维度分析 SQL 在不同时间段内的执行、响应时间分布，最后统计高频 SQL、慢 SQL、表读写占比，是 SQL 优化不可或缺的采集数据。

SQL 统计界面按用户统计 SQL 的执行时间分布、响应时间分布和 SQL 读写比例，也可以通过 show @@sql.sum 命令获取该数据。

SQL 表统计以数据表为维度统计了表的读写次数、读占比、关联表、关联和次数，运维人员可以直观地看到数据表被访问的情况，也可以通过 show @@sql.sum.table 命令获取该数据。

SQL 语句监控业务系统执行的 SQL 语句耗时多少毫秒，也可以通过 show @@sql 命令获取该数据。

高频 SQL 统计执行频率高的 SQL 语句，记录高频语句被执行的最大耗时、最小耗时，也

可以通过 show @@sql.high 命令获取该数据。

慢 SQL 统计通过设置阈值来定义慢 SQL，默认查询耗时 1000 毫秒的 SQL 语句，记录了慢 SQL 的执行耗时、执行时间和执行节点，也可以通过 show @@sql.slow 命令获取该数据。

5.2　Mycat 性能优化

Mycat 性能优化的第 1 步是 JVM、操作系统、MySQL 和 Mycat 本身的调优。

1. JVM 调优

内存占用分为两部分：Java 堆内存和直接内存映射（DirectBuffer 占用），建议堆内存大小适度，直接映射的内存尽可能大，总计占用操作系统 50%～67%的内存。下面以 16GB 内存的服务器为例，Mycat 堆内存为 4GB，直接内存映射为 6GB，JVM 参数如下：

```
-server -Xms4G -Xmx4G XX:MaxPermSize=64M -XX:MaxDirectMemorySize=6G
```

Mycat 中 JVM 参数的配置在 conf\wrapper.con 配置文件中，下面是一段实例：

```
# Java Additional Parameters
wrapper.java.additional.5=-XX:MaxDirectMemorySize=2G
wrapper.java.additional.6=-Dcom.sun.management.jmxremote
# Initial Java Heap Size (in MB)
wrapper.java.initmemory=2048
# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=2048
```

2. 操作系统调优

分别把 Mycat Server 和 MySQL 数据库机器的最大文件句柄数量设置为 5000～10000。Linux 操作系统对每个进程打开的文件句柄数量是有限制的（包含打开的 Socket 数量，影响 MySQL 的并发连接数量）。这个值可通过 ulimit 命令来修改，但 ulimit 命令的修改只对当前登录的用户有效，系统重启或者用户退出后就会失效。

3. Mycat 调优

conf/schema.xml 调优如下：

```
<system>
    <!-- CPU 核心数越多，可以越大，当发现系统 CPU 压力很小的情况下，可以适当调大此参数，如
4 核心的 4CPU，可以设置为 16，24 核心的可以最大设置为 128-->
    <property name="processors">1</property>
```

```
下面这个参数为每个 processor 的线程池大小，建议可以是 16-64，根据系统能力来测试和确定。
<property name="processorExecutor">16</property>
</system>
```

- processorBufferPool：每个 processor 分配的 Socket Direct Buffer，用于网络通信。
- Processor：每个 processor 上管理的所有连接共享。
- processorBufferChunk：为 Pool 的最小分配单元，每个 Pool 的容量为 processorBufferPool / processorBufferChunk，processorBufferPool 的默认值为 16MB，processorBufferChunk 的默认值为 4096 字节。对 processorBufferPool 参数的调整需要通过观察 show@@processor 的结果来确定。
- BU_PERCENT：已使用的百分比。
- BU_WARNS：当 Socket Buffer Pool 不够时，临时创建新 Buffer 的百分比如果经常超过 90%并且 BU_WARNS>0，则表明 Buffer 不够，需要增大 processorBufferPool。基本上连接数越多，并发越高，需要的 Pool 越大，建议 BU_PERCENT 最大为 40%～80%。

conf/schema.xml 调优如下：

```
<schema name="TESTDB" checkSQLschema="true">
```

checkSQLschema 属性建议设置为 false，不能在 SQL 中添加数据库的名称，这样可以优化 SQL 解析。

```
<dataHost name="localhost1" maxCon="500" minCon="10" balance="0"
       dbType="MySQL" dbDriver="native" banlance="0">
```

最大连接池 maxCon 的值可以改为 1000～2000，同一个 MySQL 实例上的所有 dataNode 节点共享本 dataHost 上的所有物理连接。

性能测试时，建议 minCon、maxCon、MySQL max_connections 的值相等，设为 2000 左右。另外，读写分离是否开启根据环境的配置来决定。

4. 缓存优化调整

show @@cache 命令展示了缓存的使用情况，需要经常观察其结果，并在需要时进行调整。若 CUR 接近 MAX，而 PUT 比 MAX 大很多，则表明 MAX 需要增大。HIT/ACCESS 为缓存命中率，这个值越高越好。在重新调整缓存的最大值以后，观测指标都会跟着发生变化，如果想知道调整是否有效，则需要观察缓存命中率是否在提升，PUT 是否在下降。

目前缓存服务的配置文件为 cacheservice.properties，主要使用的缓存为 enhache，在 enhache.xml 里设定了 enhance 缓存的全局属性。下面定义了几个缓存：

```
#used for mycat cache service conf
```

```
factory.encache=org.opencloudb.cache.impl.EnchachePooFactory
#key is pool name ,value is type,max size, expire seconds
pool.SQLRouteCache=encache,10000,1800
pool.ER_SQL2PARENTID=encache,1000,1800
layedpool.TableID2DataNodeCache=encache,10000,18000
layedpool.TableID2DataNodeCache.TESTDB_ORDERS=50000,18000
```

- SQLRouteCache：SQL 解析和路由选择的缓存，其大小基本上相对固定，就是所有 Select 语句的数量。
- ER_SQL2PARENTID：在 ER 分片时根据关联 SQL 查询父表的节点时用到，如果没有使用到 ER 分片，则用不到这个缓存。
- TableID2DataNodeCache：当某个表的分片字段不是主键时，缓存主键到分片 ID 的关系，这里命名为 schema_tableName（tablename 要大写）如“TEST_ORDERS”；当根据主键查询比较多时，这个缓存往往需要设置得比较大才能更好地提升性能。

5. Mycat 大数据量查询调优

如果返回的结果比较多，则建议调整 frontWriteQueueSize 的大小，即将默认值乘以 3，原因是返回数据太多。这里做了一个改进，就是超过 Pool 以后，仍然创建临时的 Buffer 以供使用，但对这些缓冲区不进行回收。在这样的情况下，需要增大 Buffer 参数 processorBufferPool。

6. Buffer Pool 调优

所有 NIOProcessor 共享一个 Buffer Pool。Buffer Pool 的总长度为 bufferPool 与 bufferChunk 的比值。

我们可以连接到 Mycat 管理端口，使用 show @@processor 命令列出所有 processor 的状态。

查看列 FREE_BUFFER、TOTAL_BUFFER、BU_PERCENT，如果 FREE_BUFFER 的数值过小，则说明配置的 Buffer Pool 的大小可能不够，这时就要根据公式手动配置这个属性了，bufferPool 的大小最好是 bufferChunk 的整数倍。例如配置 Buffer Pool 的大小为 5000，在 server.xml 文件中定义：

```
<property name="processorBufferPool">20480000</property>
```

另一个 Buffer Pool 是线程内的 Buffer Pool，这个值可以根据 processors 的数值计算出来。具体看 server.xml 配置详解。

7. Mycat I/O 调优

NIOProcessor 类持有所有的前后端连接，定期进行空闲检查和写队列检查。Mycat 是通过

遍历 NIOProcessor 持有的所有连接来完成这个动作的。可以适当地根据系统性能调整 NIOProcessor 的数量，使得前、后端连接可以均匀地分布在每个 NIOProcessor 上，这样就可以加快每次的空闲检查和写队列检查，快速地将空闲的连接关闭，减少服务器的内存使用量。

NIOReactor 是 NIO 中具体执行 selector 的类，当该类事件发生时，就通知上层逻辑进行具体处理。NIOReactor 的数量与具体事件处理器的数量相等，如果系统配置允许，则应该尽可能地增加 NIOReactor 的数量，默认值是 CPU 的核心数。

AsynchronousChannelGroup 是 AIO 中必须提供的一个组成部分，根据 processors 的数值确定实例数和 channelGroup 组内线程池的大小。后端 AIO 连接循环读取 AsynchronousChannelGroup 数组中的实例。如果在 AIO 模式下使用 Mycat，则调整这个参数也是有必要的，默认值是 CPU 的核心数。

5.3 MySQL 优化技术

5.3.1 数据库建表设计规范

1. MySQL 字符集

MySQL 的字符集支持（Character Set Support）涉及两个方面：字符集（Character set）和排序方式（Collation）。

对于字符集的支持可细化到四个层次：服务器（server）、数据库（database）、数据表（table）、连接（connection）。

连接 MySQL 服务并通过如下命令查看字符集的详情：

```
SHOW VARIABLES LIKE 'character_set%';show variables like '%collation_%';
```

客户端字符集的设置如下：

```
character_set_client= utf8 ;
```

连接层字符集的设置如下：

```
character_set_connection= utf8
```

数据库端字符集的默认设置如下：

```
character_set_database= utf8
```

服务端字符集的设置如下：

```
character_set_server= utf8
```