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

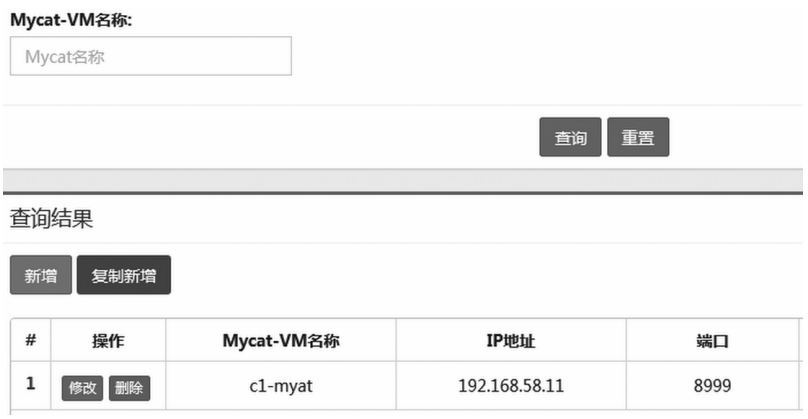

图 5-2

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。

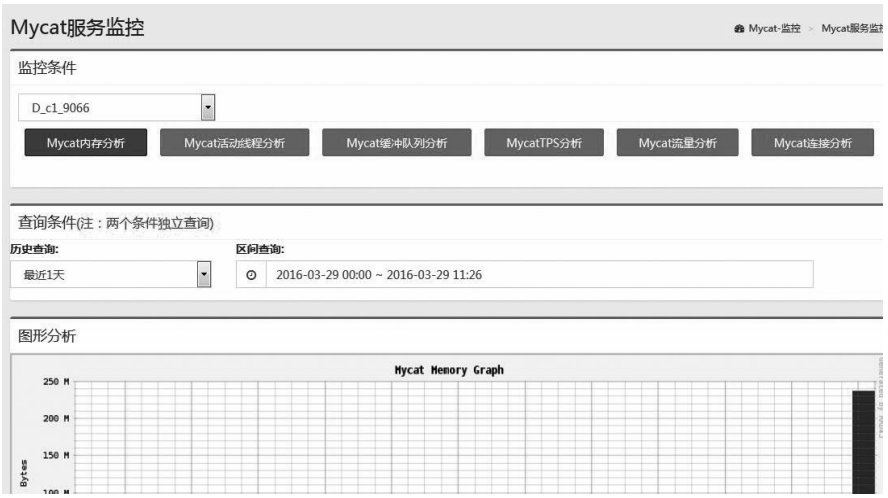

图 5-3

（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
```