## 实验目的

1、掌握 RocketMQ 的安装，使用方法

2、掌握基于 SpringBoot 的 RocketMQ 使用方法

3、验证 RocketMQ 对于写数据库的效率提升

## 实验环境

1、服务器 A: Ubuntu 18.04 服务器 2 核 4G 内存虚拟机一台，图形界面，安装 JDK 11， Maven、git

2、服务器 B：Ubuntu 18.04 服务器 2 核 2G 内存虚拟机一台，命令行界面，安装 JDK 11，Maven、git，JMeter 5.4.1

3、服务器 C：Ubuntu 18.04 服务器 2 核 2G 内存虚拟机一台，命令行界面，安装 JDK 11，Maven、git，MySQL 8.0

4、服务器 C：Ubuntu 18.04 服务器 2 核 2G 内存虚拟机一台，命令行界面，安装 JDK 11，Maven、git，RocketMQ 4.9.1

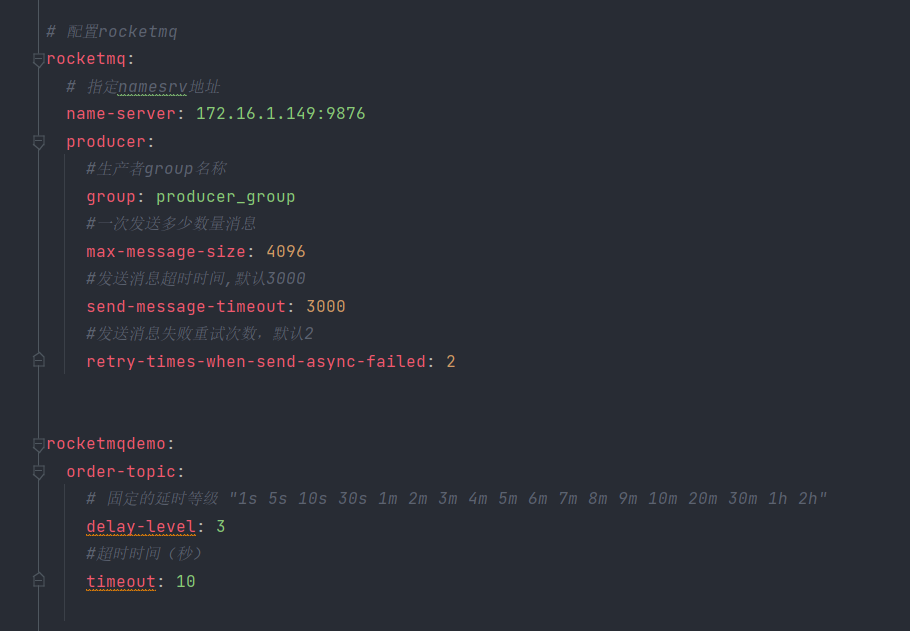
## 实验内容及要求

通过实验二中，我们已经知道数据库写的速度远远慢于数据库读的速度。为了验证 RocketMQ 对于数据写操作的提速，设计一个实验对比使用 Rocket 写数据库的效率。要求实 现以下 RESTful API,并与实验二的未使用 RocketMQ 的实验进行对比，给出实验报告。

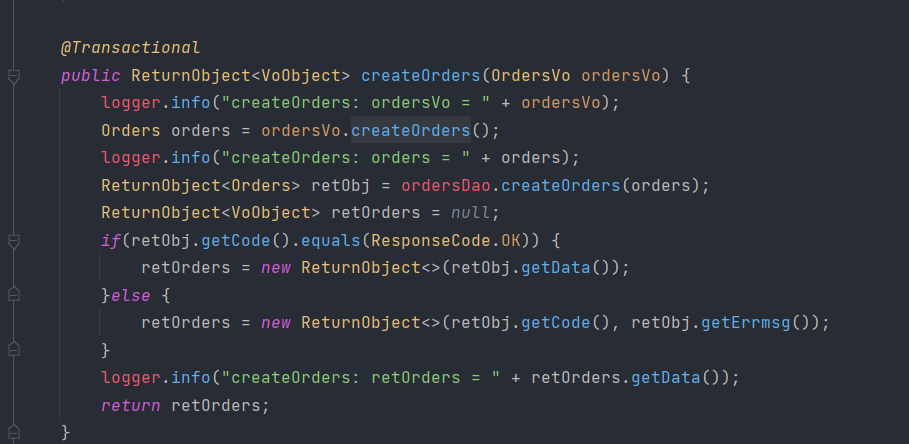


## 具体设计

RocketMQ配置



不使用RocketMQ的一般写入Service层对应代码：



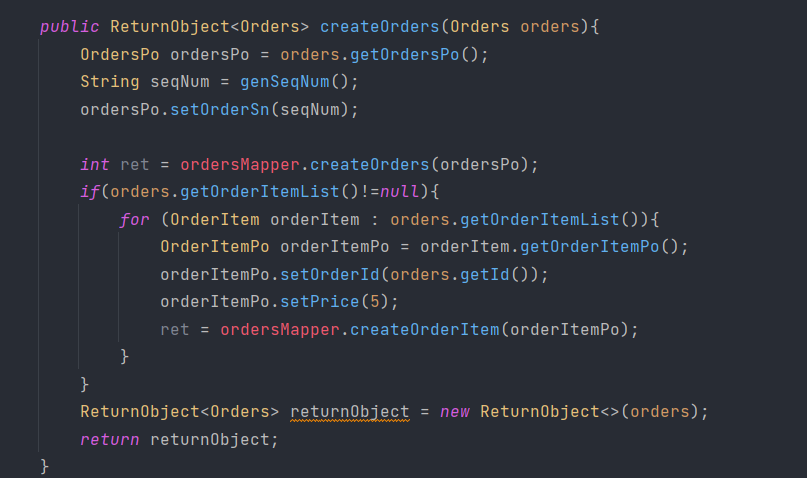
使用RocketMQ的Service层对应代码：

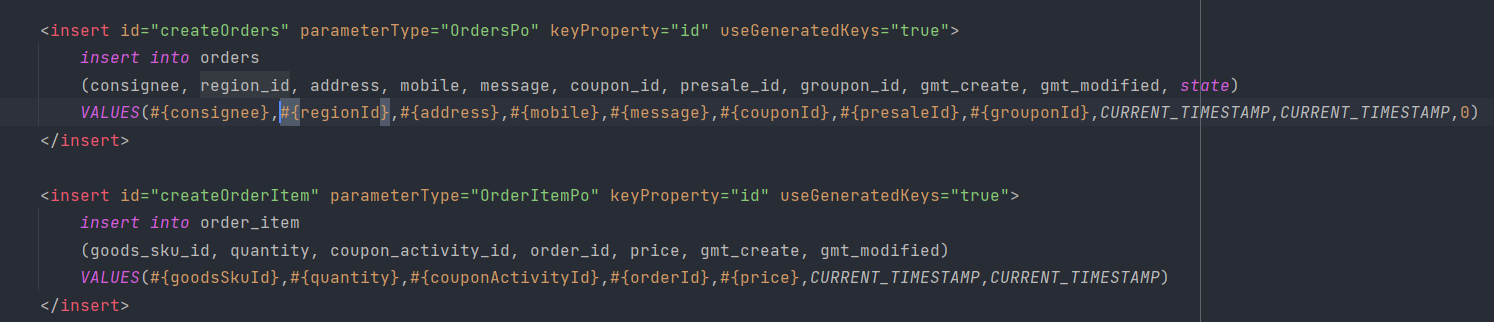


消费者：



二者使用相同Dao层代码与Mapper：





在实验之前，我们进行相关准备与规划：

1. 将JVM、RocketMQ、MySQL、jmeter运行在不同服务器上，防止相互干扰
2. 预热JVM：先运行1000个线程的测试计划，让jvm预热，防止干扰
3. 测试RocketMQ服务器的最大压力：我们一开始运行一个nameserver和一个broker，进行相关测试。之后我们试图增加nameserver与broker，但增加第二个broker时，会出现insufficient Memory的错误。因为非图形化界面服务器内存只有2G，除去系统应用后只剩1G，运行一个nameserver与broker之后只有900M，无法再增加nameserver与broker，故只用单个nameserver与broker进行试验
4. 测试jvm最大压力：

我们使用jmeter -GUI界面，从20线程/秒开始，逐步增加线程，寻找两种方式的阻塞边界。

一是为了找到一个不阻塞的值，以对比是否使用RocketMQ对于写速度的提升

二是为了找到两种方式的最大边界，以验证使用RocketMQ可以对高并发的写有多大提升

经过不断增加线程数，我们可以得到以下结论：

在不使用RocketMQ的情况下，阻塞边界大约位于400-500线程/秒

在使用RocketMQ的情况下，阻塞边界大约位于650-700线程/秒

（阻塞边界是指在增加线程的过程中，如果第n+1次的响应时间明显大于第n的响应时间，并且在多次重复后仍明显大于，那么我们就将第n次~第n+1次的线程数视为阻塞边界）

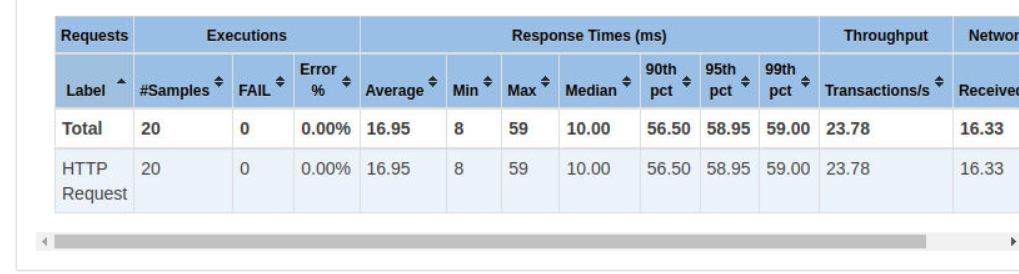
所以，为了对比RocketMQ对于写效率的提升，我们选择20个线程（轻量级写）和100个线程（中等写）来进行对比试验

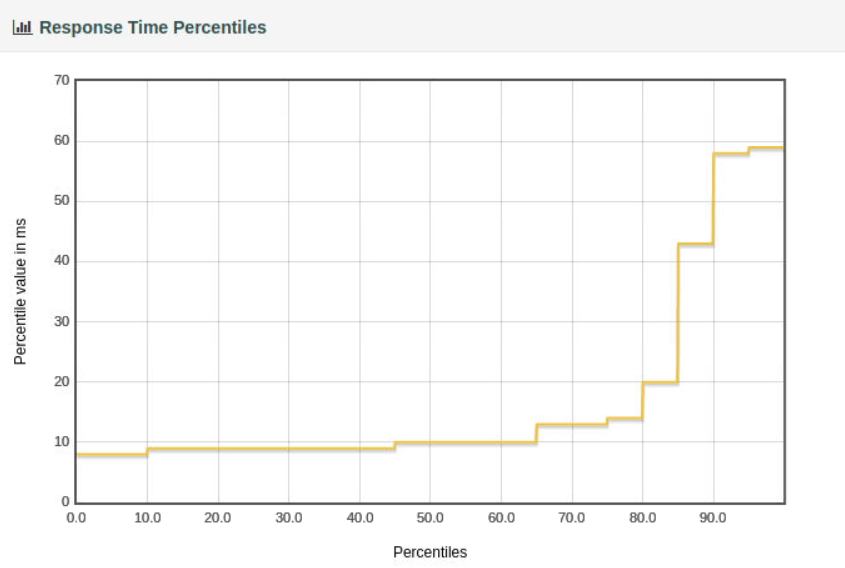
使用RocketMQ的url：/orders/rocketmq

不使用RocketMQ的url：/orders

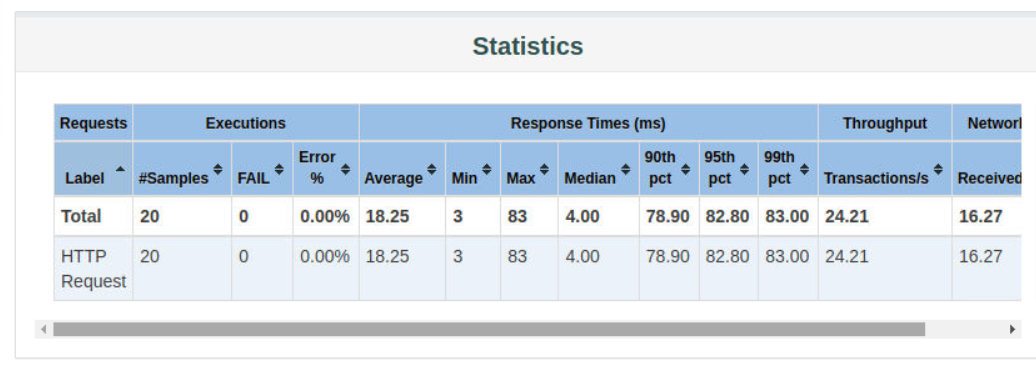
## 具体测试与结果分析

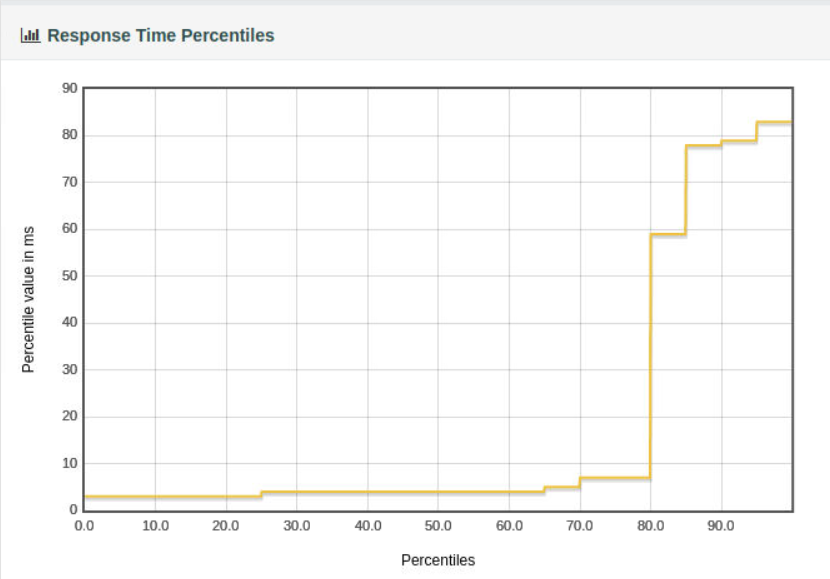
20个线程，不使用RocketMQ：





20个线程，使用RocketMQ：

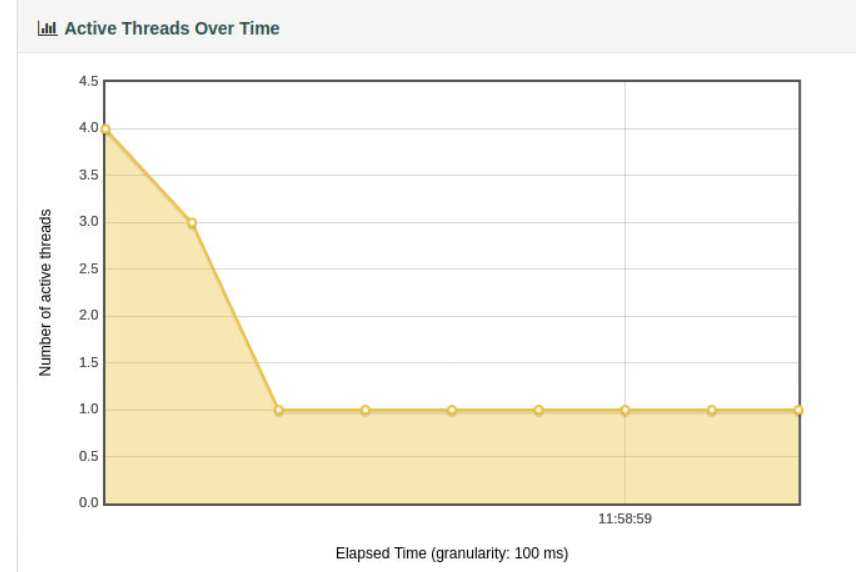


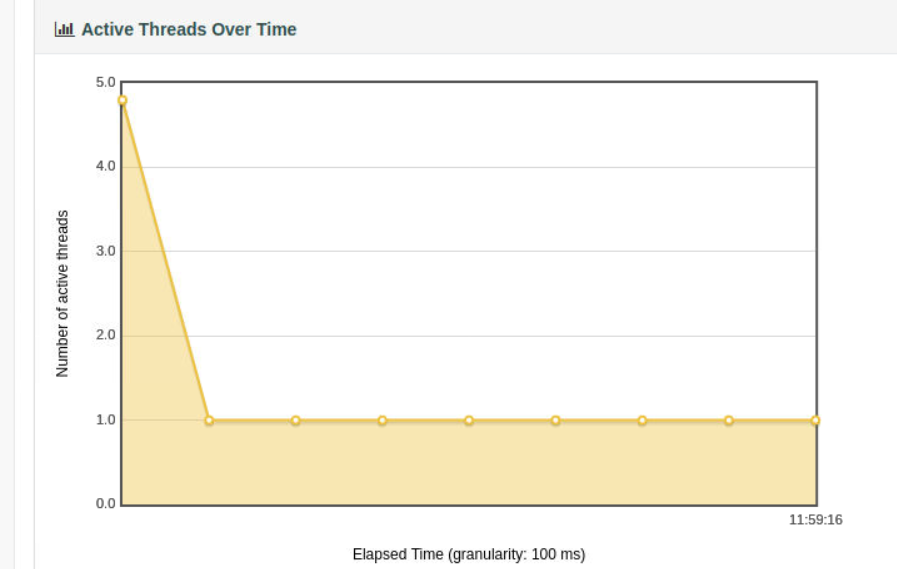


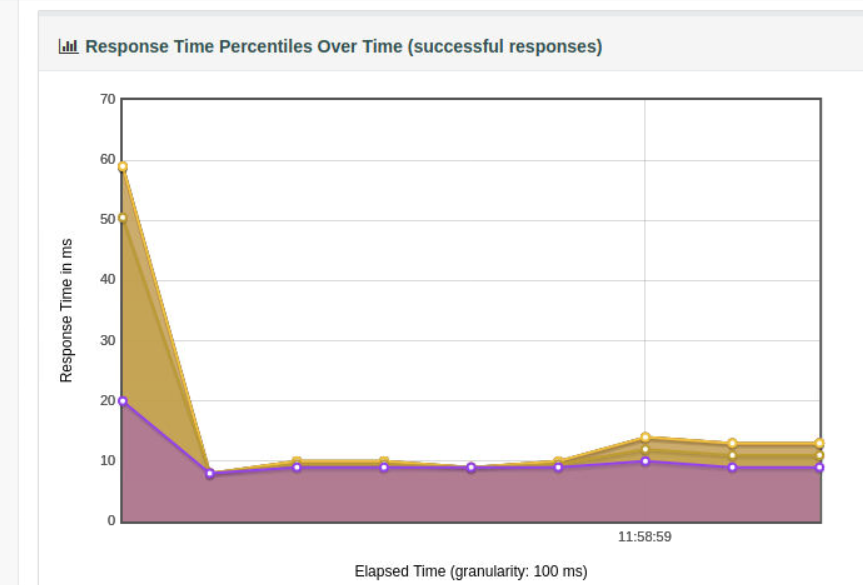
可以看到，在不阻塞的情况下，RocketMQ的写速度只是稍快于不使用RocketMQ的

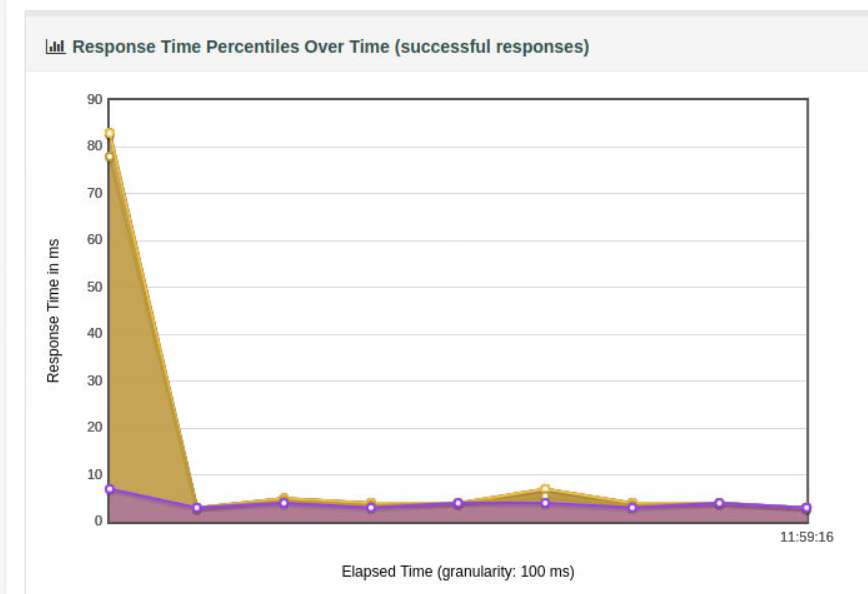
虽然使用RocketMQ的min值明显低于不使用RocketMQ，但使用RocketMQ的情况下，在80%左右会有明显的阻塞（响应时间急速增加），不使用RocketMQ也是大约位于80%左右

因此大约是由4个线程处于阻塞状态









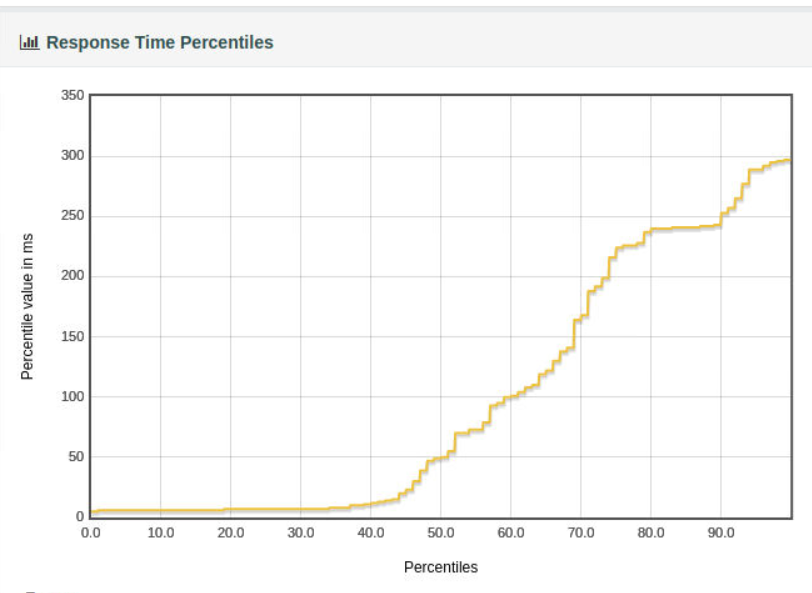
查看线程时间分布图，可以看到两种方式在测试开始时都是有4个线程的

结合响应时间的时间分布图，也可以看到一开始的四个线程是最慢的

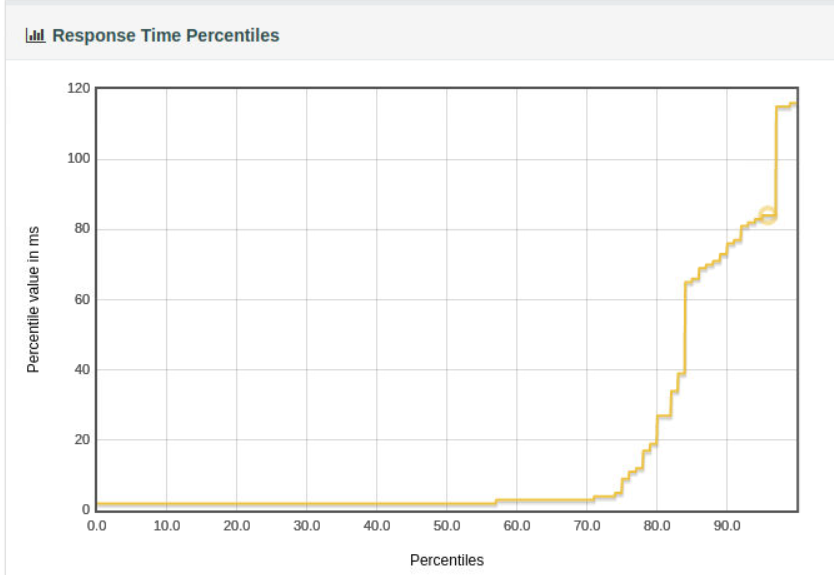
因此，如果忽略这四个阻塞状态的线程，我们还是可以得出使用RocketMQ快于不使用RocketMQ的结论

下面我们进行100写操作/秒的测试，来验证RocketMQ在较高负载情况下对响应时间的影响

不使用RocketMQ，100写操作/秒



使用RocketMQ，100写操作/秒



对比两张图，可以明显看到，在高并发的情况下，使用RocketMQ拥有更良好的性能

包括更靠后的阻塞边界、更低的总体响应时间

综上所述：

从写效率来看，使用RocketMQ可以拥有更快的写效率

从高负载来看，使用RocketMQ可以在高负载情况下带来更良好的体验

## git地址：

gitee：<https://gitee.com/yjz6666774/heavy-team/tree/master/Experiment/Exp_5>

github：https://github.com/529106896/HeavyTeam/tree/main/Experiment/Exp\_5