華中科技大學

数据中心技术课程实验报告

院　系　武汉光电国家研究中心

班　级　硕 2104

学　号　M202173488

姓　名　谢晨

2021 年 01 月 06 日

对象级存储入门实验

一. 系统搭建

实验环境

在本次实验中，我一开始选择的是在 VMware 中安装的 ubuntu14.04 版本中进行实验，在该版本中，下载和安装 minio 都可以完成，但登陆到 minio 时，页面显示空白，无法继续。因而选择在 VMware 中安装 ubuntu 20.04.3 版本，以进行实验

MINIO 下载

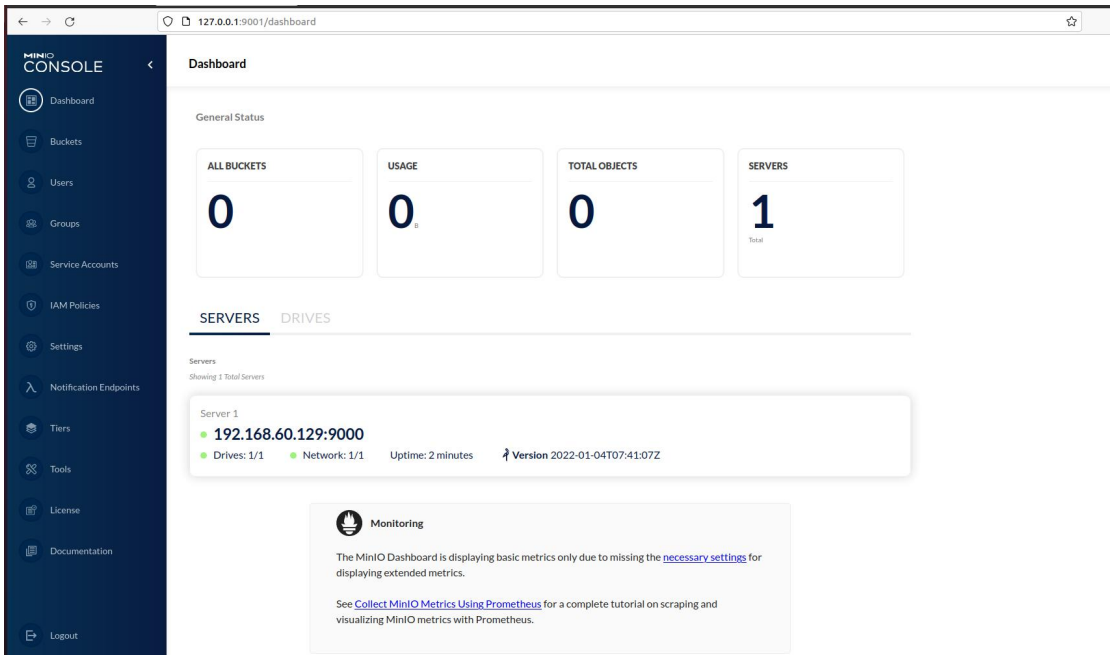
1.进入下载页面，可以看到 minio 官网已经对如何下载 minio 进行了说明

2.打开 ubuntu 终端，复制代码并执行

3.配置帐号为 xiec 密码为 xiec0401，使用管理员权限运行以下代码

```
sudo MINIO_ROOT_USER=admin MINIO_ROOT_PASSWORD=password ./minio server /mnt/data --console-address ":9001"
```

4.打开浏览器进入 127.0.0.1:9001,输入账号密码后登陆



5.至此，minio 安装完成

二、s3bench 基准测试

安装 s3bench 工具

1.安装 go 环境：

```
a@ubuntu:~$ sudo apt install golang-go
```

2.安装 s3bench

```
a@ubuntu:~$ go get -u github.com/igneous-systems/s3bench
```

使用 s3bench 工具

1.命令行中修改缺省参数以运行 s3bench

```
s3bench.exe \
    -accessKey=xiec\
    -accessSecret=xiec0401 \
    -bucket=loadgen \
    -endpoint=http://127.0.0.1:9000 \
    -numClients=10\
    -numSamples=100 \
    -objectNamePrefix=loadgen \
    -objectSize=1024
```

```
a@ubuntu:~/go/bin$ ./s3bench -accessKey=xiec -accessSecret=xiec0401 -endpoint=h
ttp://127.0.0.1:9000 -bucket=mybucket1 -objectNamePrefix=loadgen -numClients=10
 -numSamples=100 -objectSize=1024
```

2.观察实验结果

```
Results Summary for Write Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  1.22 MB/s
Total Duration:    0.080 s
Number of Errors:  0
-----------------------------------
Write times Max:       0.015 s
Write times 99th %ile: 0.015 s
Write times 90th %ile: 0.011 s
Write times 75th %ile: 0.010 s
Write times 50th %ile: 0.008 s
Write times 25th %ile: 0.006 s
Write times Min:       0.003 s


Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  4.01 MB/s
Total Duration:    0.024 s
Number of Errors:  0
-----------------------------------
Read times Max:       0.009 s
Read times 99th %ile: 0.009 s
Read times 90th %ile: 0.004 s
Read times 75th %ile: 0.002 s
```

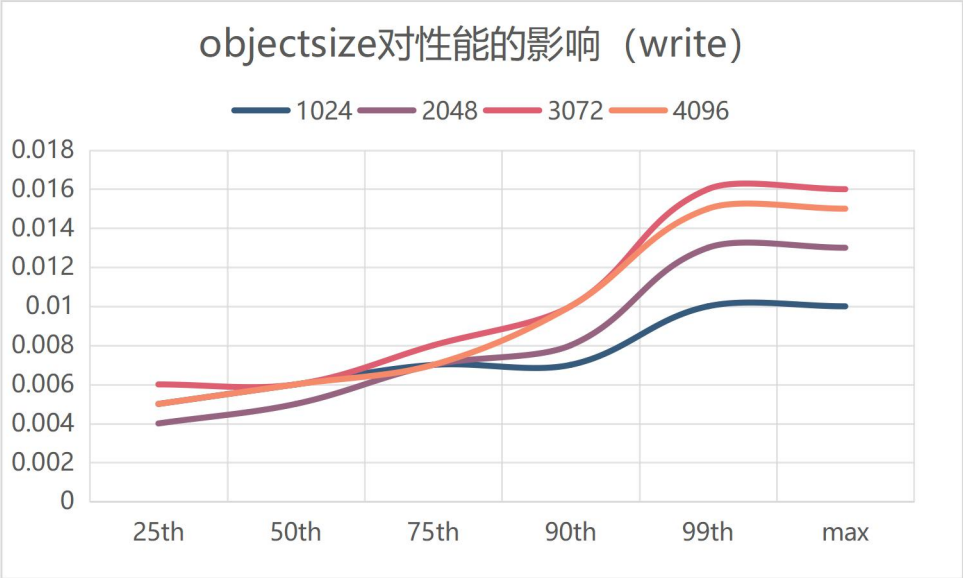
```
-----------------------------------
Read times Max:       0.009 s
Read times 99th %ile: 0.009 s
Read times 90th %ile: 0.004 s
Read times 75th %ile: 0.002 s
Read times 50th %ile: 0.002 s
Read times 25th %ile: 0.001 s
Read times Min:       0.001 s
```

此时，程序已经能够正常运行并能够成功得到结果。

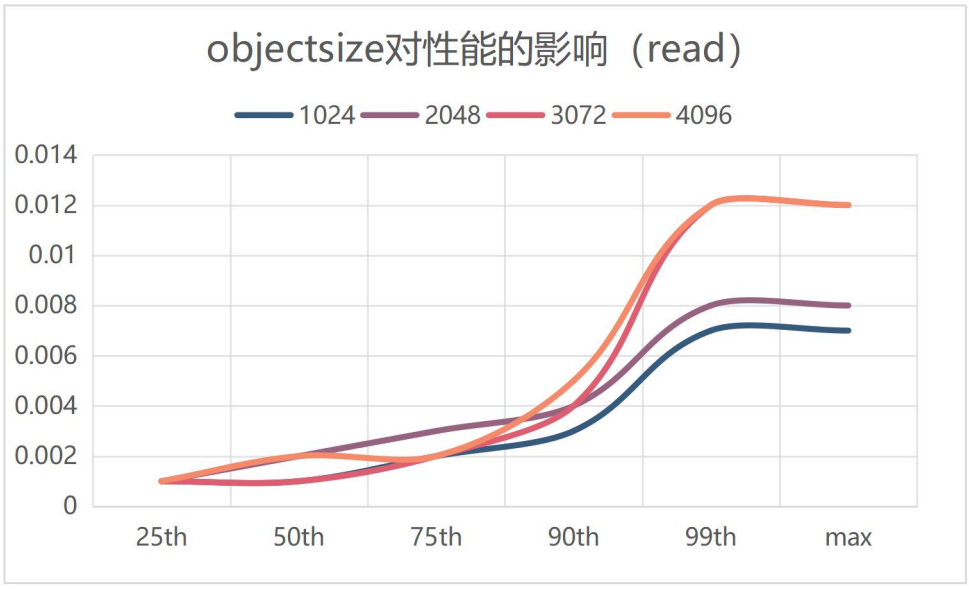
开始测试

1.对象尺寸对性能的影响

选取对象尺寸分别为：1024,2048,3072,4096，观察尾延迟。



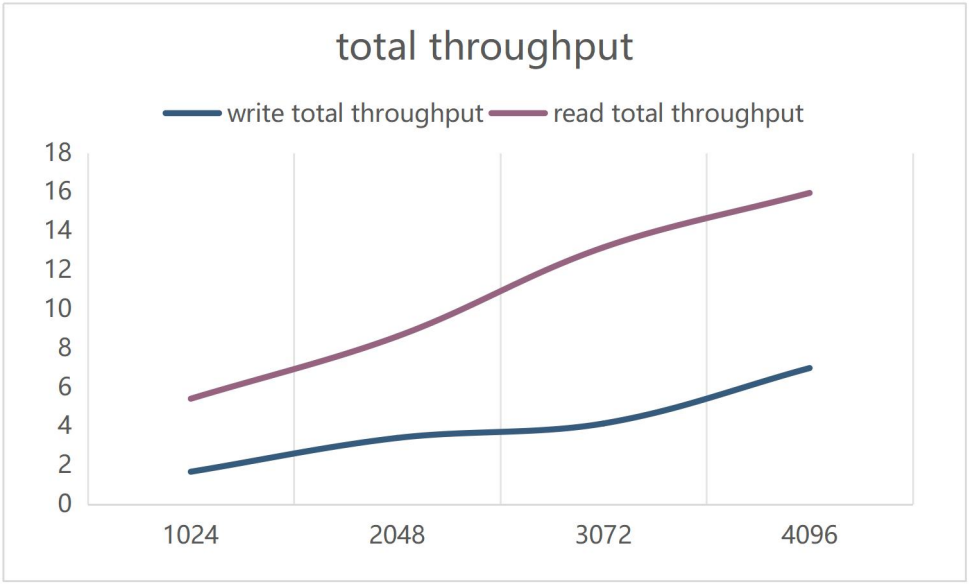
不难发现，当 objectsize 增大的时候,读操作的尾延迟和中位数延迟都增加了，当 objectsize 为 1024 时，延迟变化较为平稳，而 objectsize 上升到 4096 时，延迟变化会非常快。



当 objectsize 增大的时候,读操作的尾延迟增加了，当 objectsize 分别为 1024,2048,3072,4096 时，延迟变化趋势几乎相同。当 objectsize 为 4096 时，最大延迟约是 objectsize 为 1024 时的 1.5 倍。

所以可以得出结论，objectsize 的增加，会导致读写的尾延迟增加

选取对象尺寸分别为：1024,2048,3072,4096，观察吞吐量。



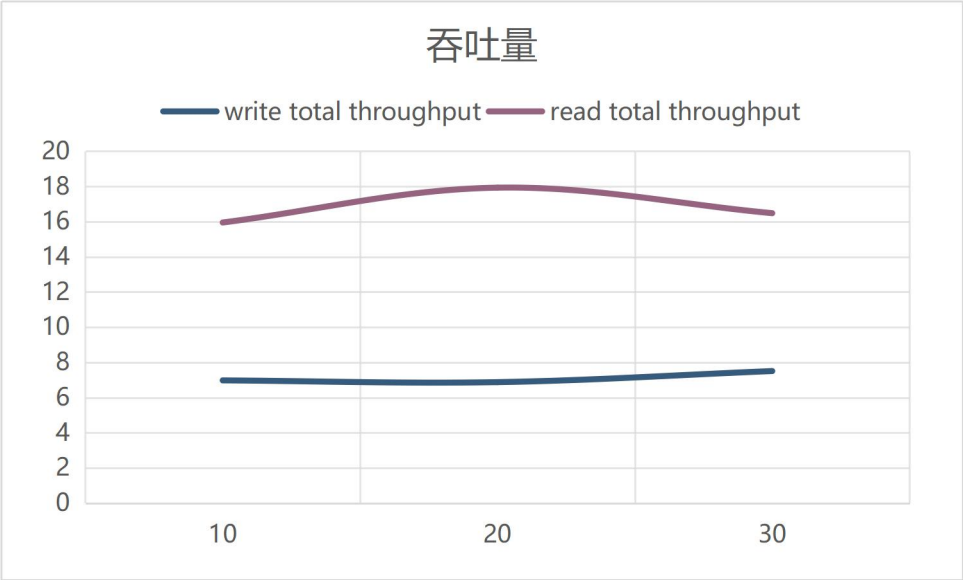
当 objectsize 增大的时候,读写操作的吞吐量增加了，当 objectsize 分别为 4096 时，其读写吞吐量比 1024 时搞了接近两倍。

所以可以得出结论，objectsize 的增加，会导致系统吞吐量的增加。

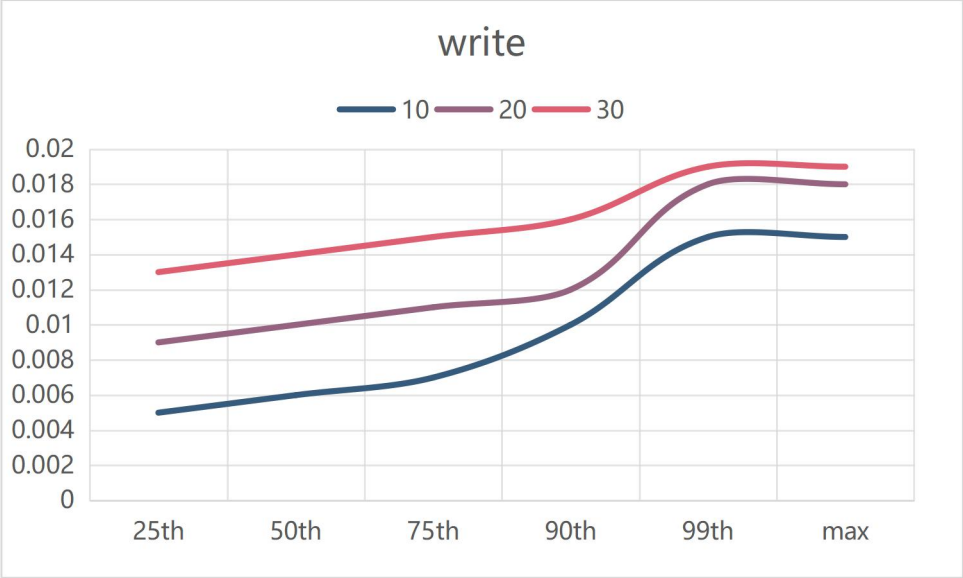
2.numClients 对性能的影响

设置 object size = 4096，分别测试 numClients 为 10,20,30 时的性能。

吞吐量：

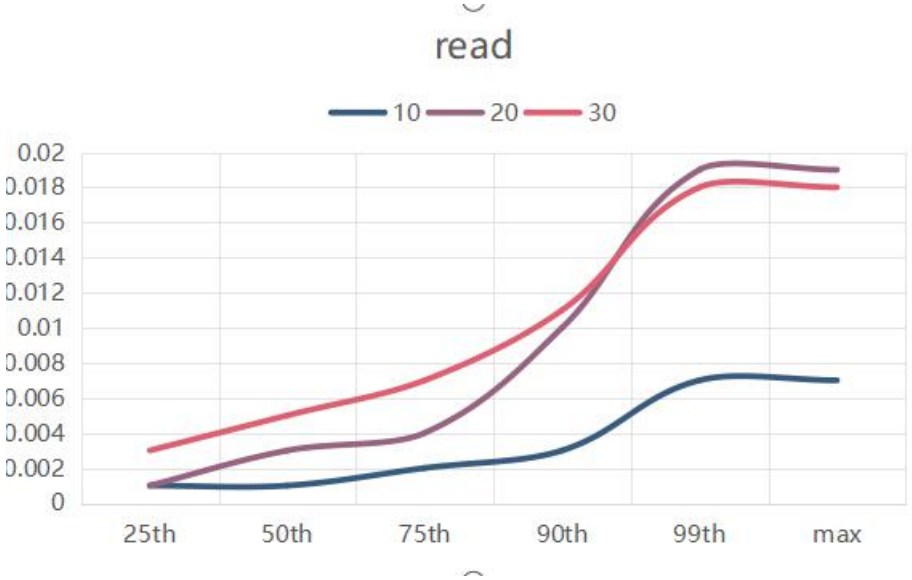


当 numClients 为 30 的时候，其吞吐量相比 10 的时候有了些许的增加



当 numClients 增大的时候,写的尾延迟增加了，当 numClients 为 30 时，其延迟总体变化不大，但在任意情况下均大于 numClients 为 10 的延迟。

所以可以得出结论，numClients 的增加，会导致读写的尾延迟增加



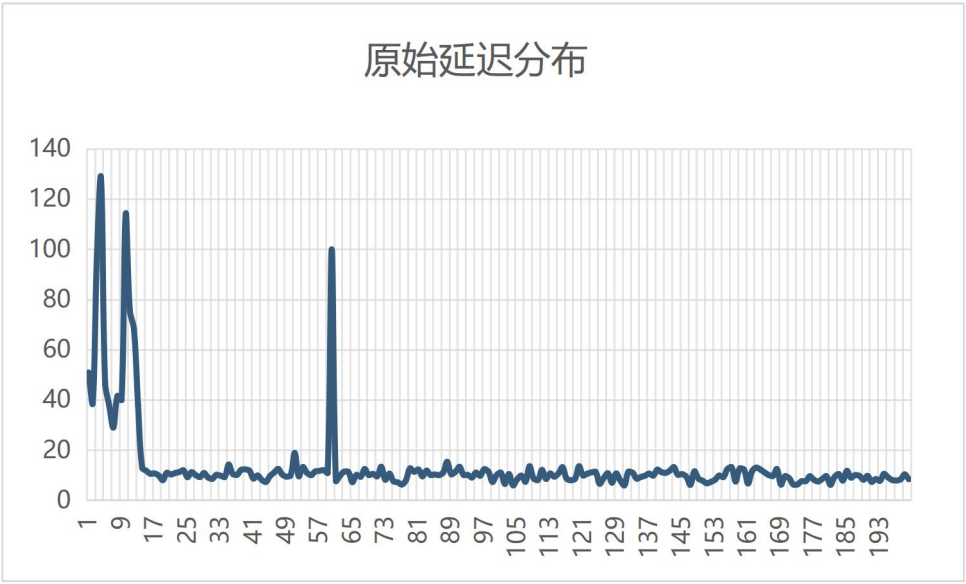
当 numClients 增大的时候,读的尾延迟增加了，当 numClients 为 30 是，尾延迟几乎为 numClients 为 10 的三倍。

所以可以得出结论，numClients 的增加，会导致读写的尾延迟增加

三、尾延迟测试

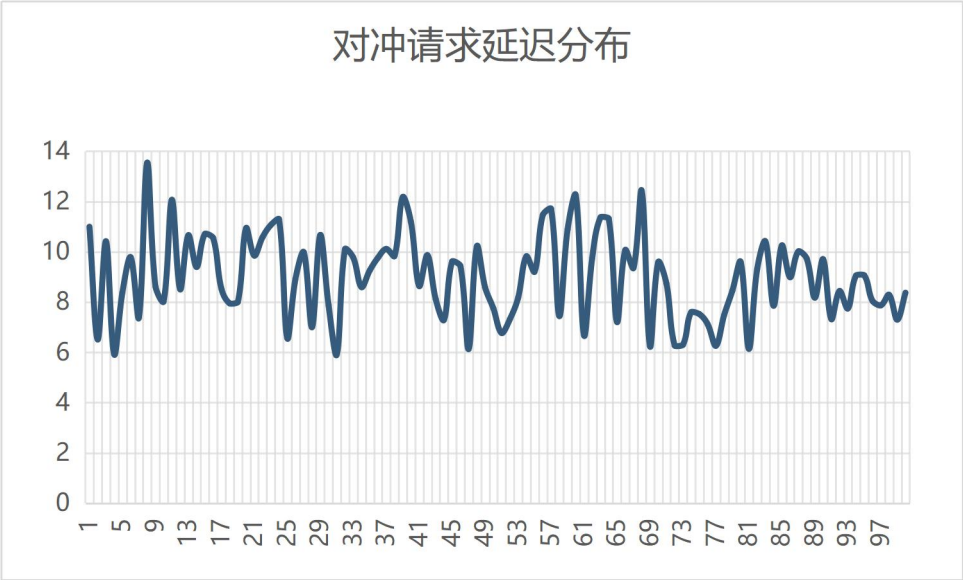
设置 objectsize 为 100，不对请求进行限速，分别进行原始请求，对冲请求以及相关请求的测试，得出其延迟分布图如下所示。

原始延迟分布



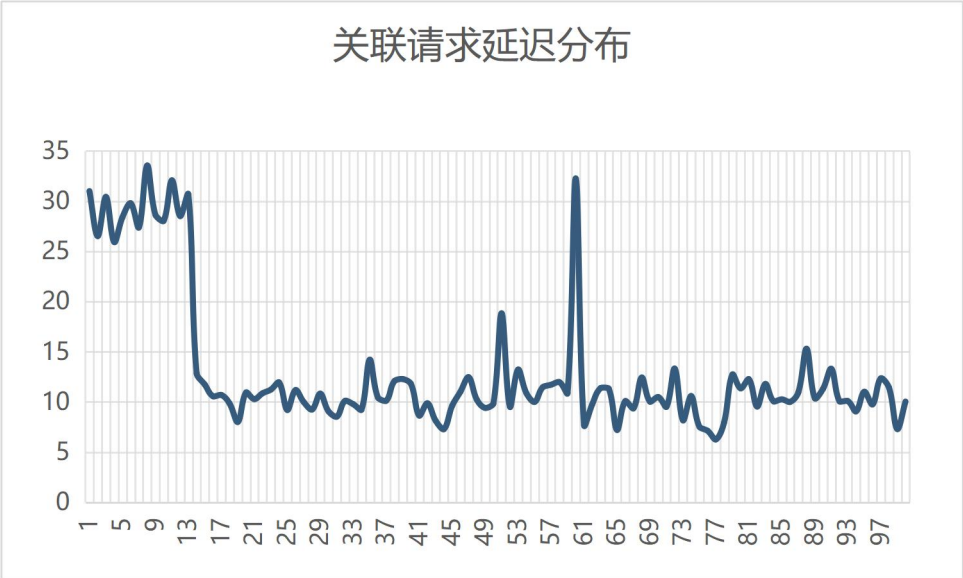
可以发现开始的延迟非常高，超过了 120，后续逐渐平稳到 10 左右。

对冲请求延迟分布



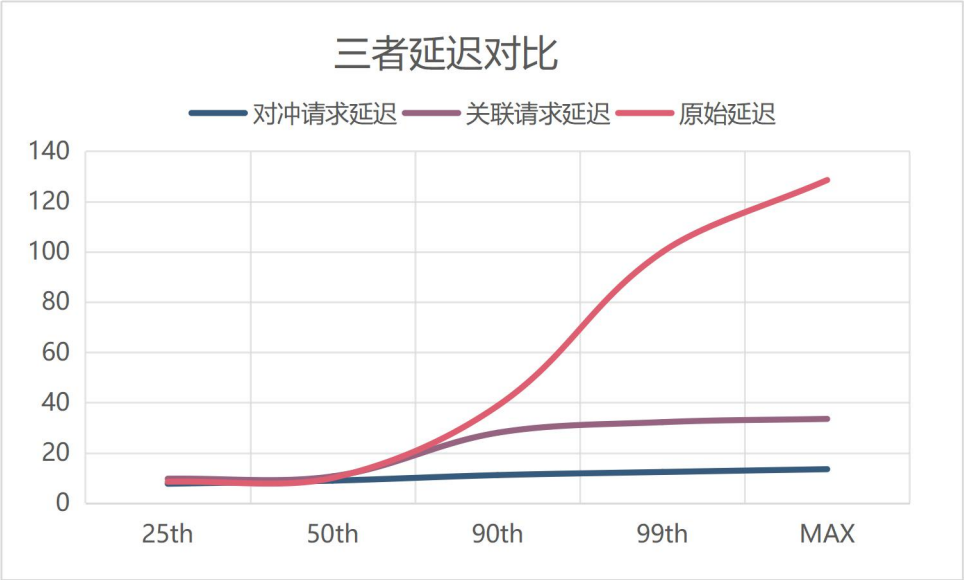
可以发现对冲请求的延迟分布在 10 上下振动，相比普通请求，对冲请求在延迟优化上有了质的飞跃

关联请求延迟分布



可以发现关联请求的延迟分布开始较高，在 30 左右，但逐渐平稳到 10 左右，相比普通请求，关联请求在延迟上有了优化，但并不如对冲请求。

三者延迟对比



可以发现，关联请求和对冲请求，都可以有效地改善系统尾延迟，并且在效果上，对冲请求对尾延迟的改善能力比关联请求更强。