# 实验内容

1. 熟悉性能指标：吞吐率、带宽、延迟
2. 分析不同负载下的指标、延迟的分布
3. 观测尾延迟现象
4. 尝试对冲请求方案

# 对象存储

在如今数据规模支持增长，结构日趋复杂的时代，如何搭建一个兼具高性能和高可靠的存储系统是一个困扰相关人员的难题，对象存储便由此应运而生。

对象存储（Object-based Storage）是一种新的网络存储架构，综合了NAS和SAN的优点，同时具有NAS的分布式数据共享和SAN的高速直通访问等优势，提供了具有高性能、高可靠性、跨平台、支持安全数据共享的存储体系结构。

# 实验环境

操作系统：Windows10家庭版

服务器：Minio

客户端：Minio Client

评测工具：s3bench

# 实验步骤及结果

1. 环境搭建

* Python环境

前往Python官网下载，使用官方安装器，安装，配置系统环境变量。

* Go环境

前往Go语言中文社区，下载Windows下的安装器，安装，配置系统环境变量。

* Minio及客户端

前往Minio官网下载，直接安装即可。然后配置运行Minio的脚本，如下图所示。

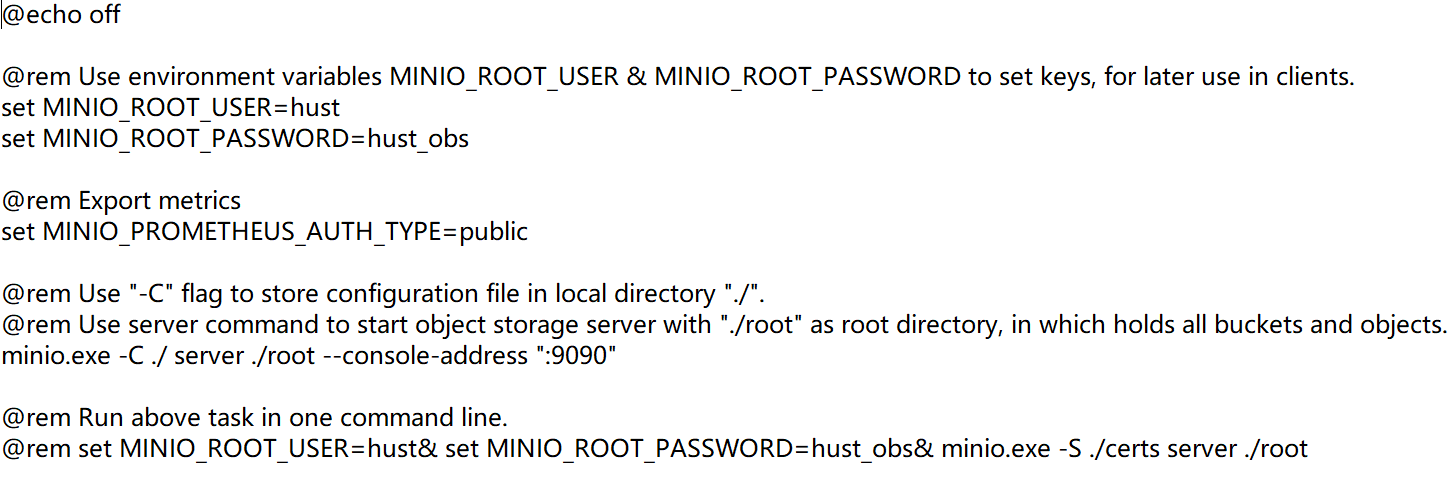


图 1 运行Minio脚本

配置完成后运行，使用浏览器访问<http://127.0.0.1:9090>，输入设置好的用户名和密码，进入后，如下图所示。

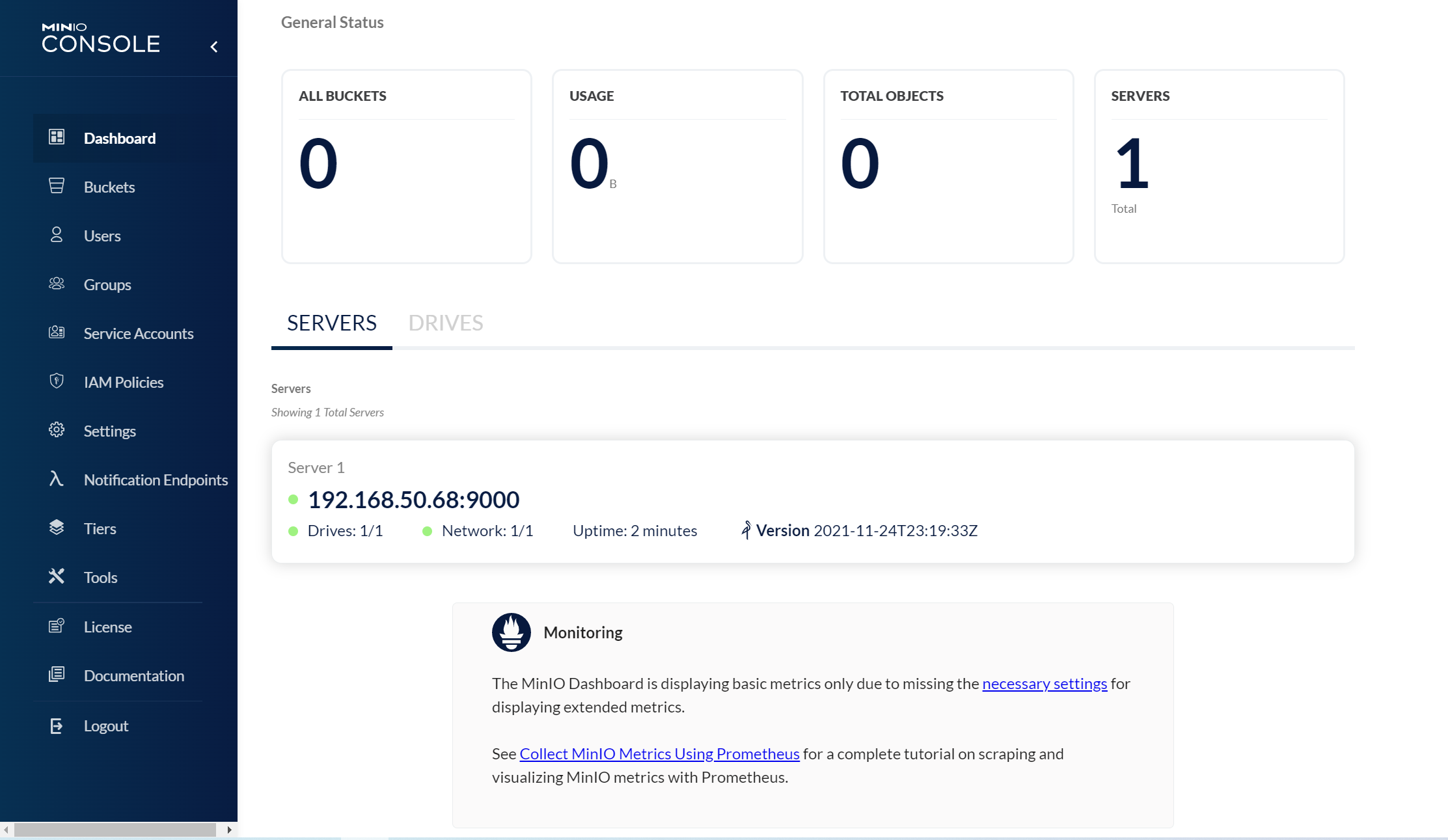


图 2 运行Minio

* S3Bench

直接使用提供的预编译Windows执行程序，并配置相应的运行脚本。运行相应脚本，结果如下图所示，说明S3Bench正确配置。

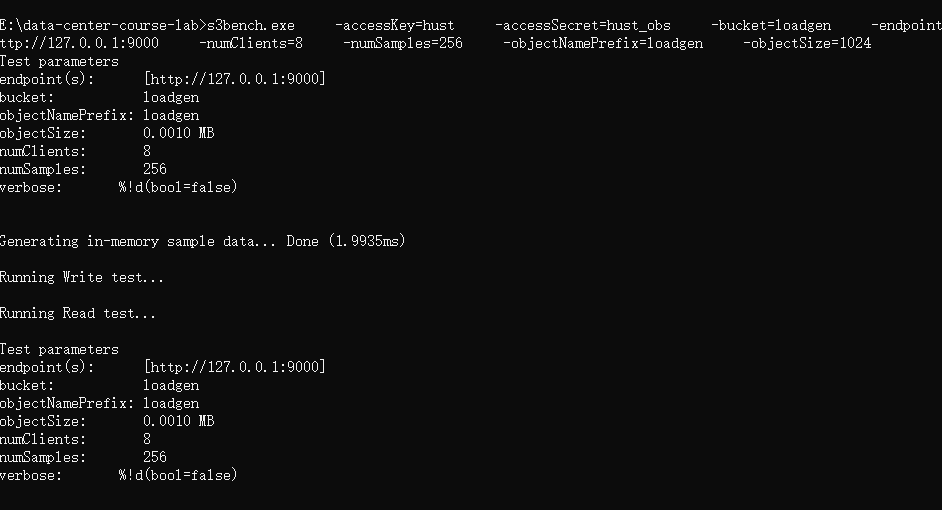


图 3 运行S3Bench

1. 标准测试

* 对象尺寸对尾延迟的影响

选取了4KB，32KB，256KB，3种不同的对象尺寸进行了实验，实验结果如下图所示。

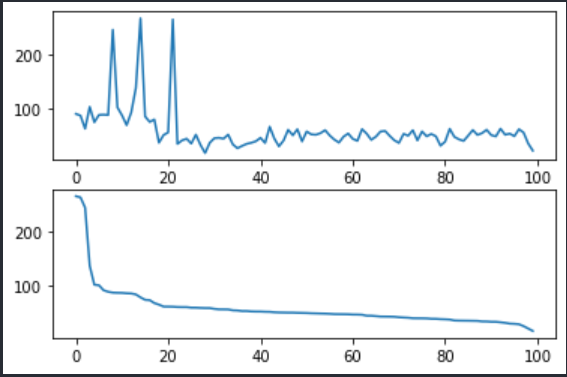
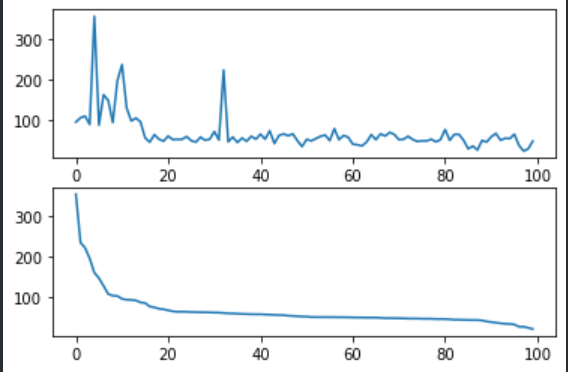
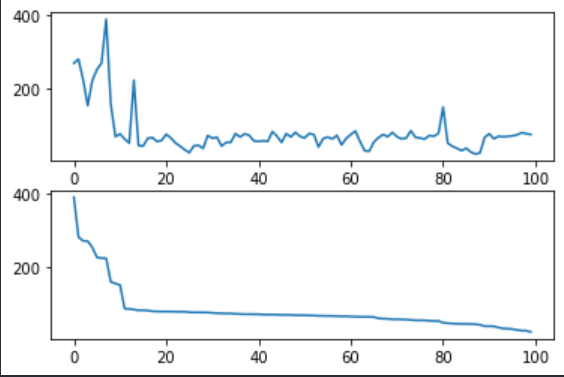
  

图 4 4KB，32KB，256时的尾延迟分布

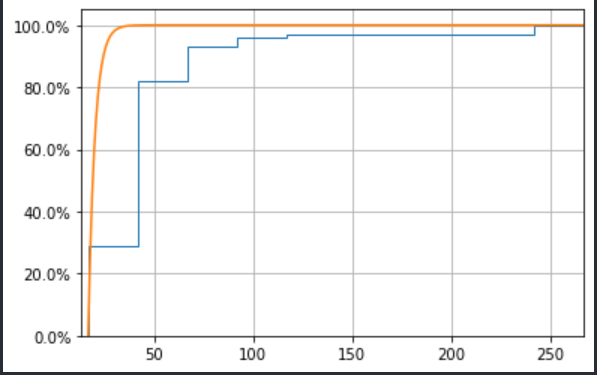
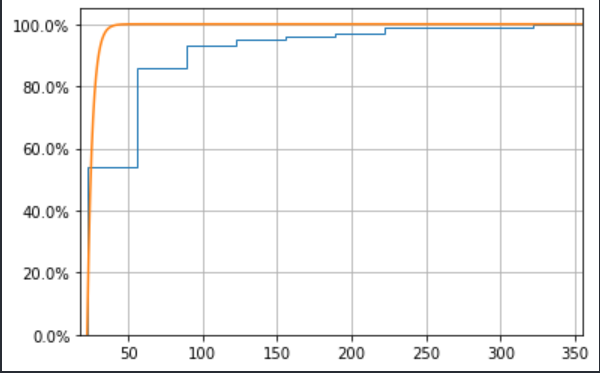
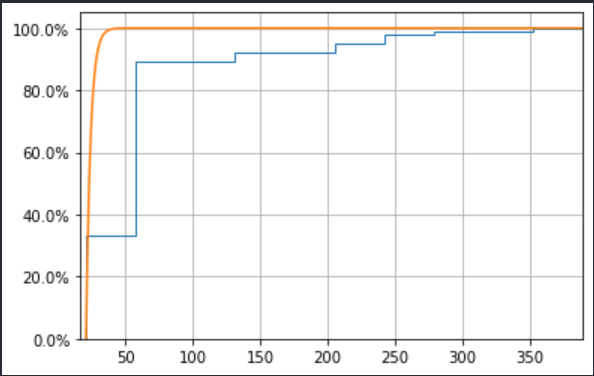
  

图 5 4KB，32，256时的尾延迟分布

可以看出，对象尺寸越大，在向存储系统写入文件时尾延迟也越大。

* 对象尺寸对吞吐量的影响

选取了512KB，1024KB，2048KB，3种不同的对象尺寸进行了实验，实验结果如下图所示。

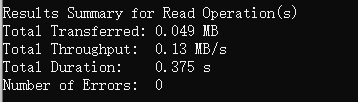
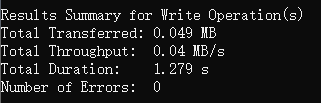


图 6 对象尺寸为512KB时的读写吞吐量

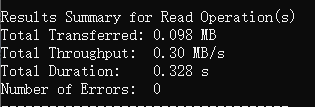
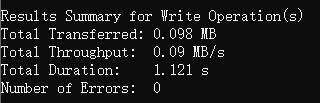


图 7 对象尺寸为1024KB时的读写吞吐量

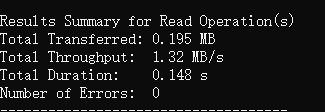
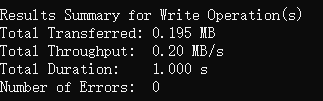


图 8对象尺寸为2048KB时的读写吞吐量

可以看出，对象尺寸越大，读写吞吐量越大。

* 并发性对尾延迟的影响

测试并发线程为1，2，4，对象尺寸为256KB，系统尾延迟的分布。结果如下图所示。

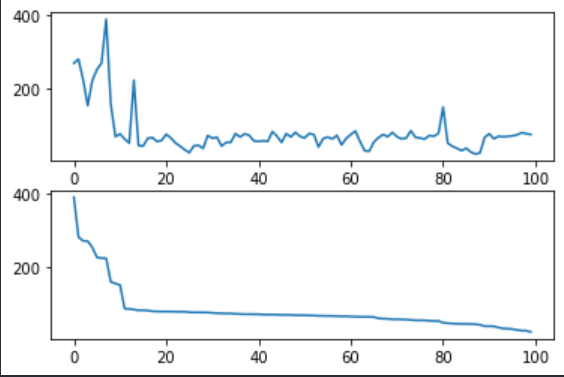
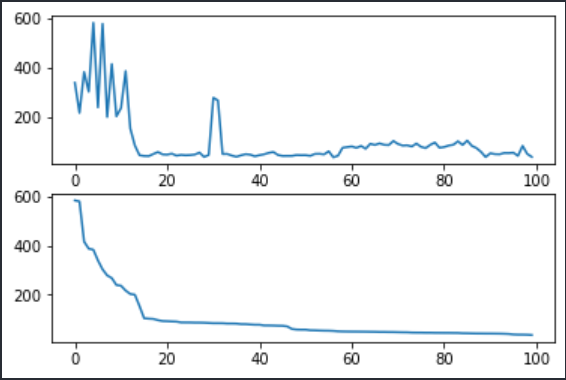
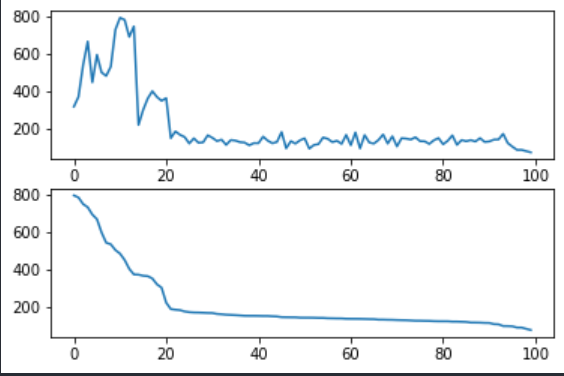
  

图 9 并发线程为1，2，4时尾延迟分布

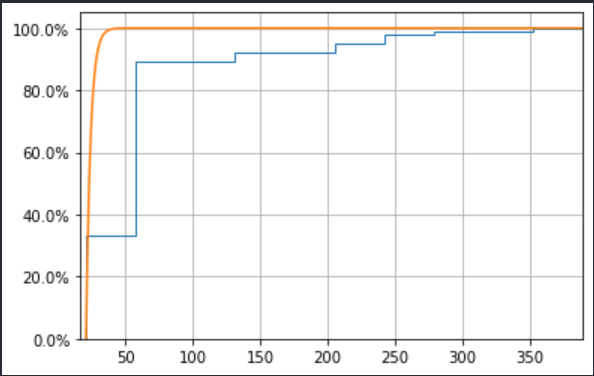
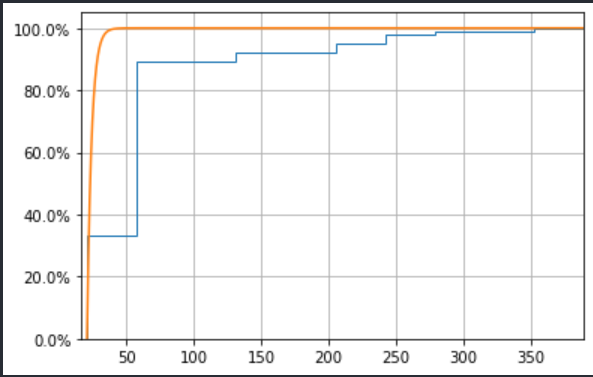
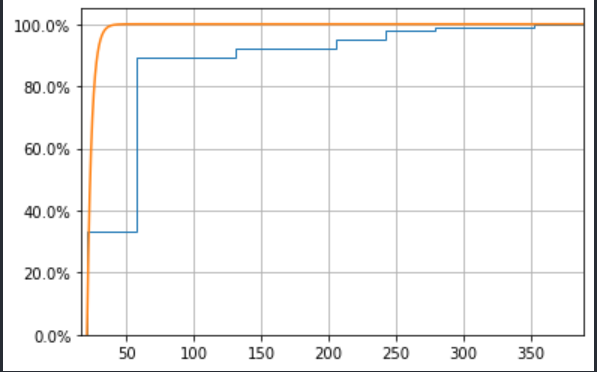
  

图 10 并发线程为1，2，4时尾延迟分布

可以看出，并发度越大，尾延迟也越大，尾延迟现象更明显。

* 并发性对吞吐量的影响

测试并发线程为1，2，4，对象尺寸为1024KB时，系统吞吐量情况。结果如下图所示。

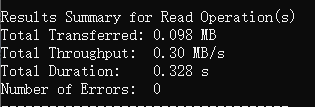
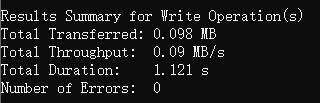


图 11 并发线程为1时的读写吞吐量

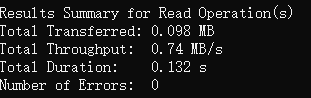
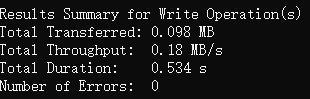


图 12 并发线程为2时的读写吞吐量

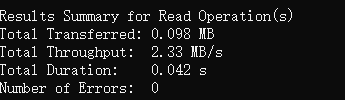
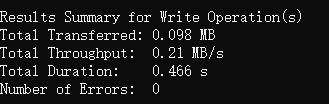


图 13 并发线程为4时的读写吞吐量

可以看出，并发度越大，吞吐量也越大。

1. 尾延迟挑战

使用对冲请求来缓解尾延迟现象。由于是在单机上做的实验，这里只对对冲请求进行模拟。重复两次实验，最后结果取两个实验结果对应位置较低的那个延迟。在实验中表现为，上传200项任务，结果取min(latency[x], latency[100 + x]) for x in range(100)即可。

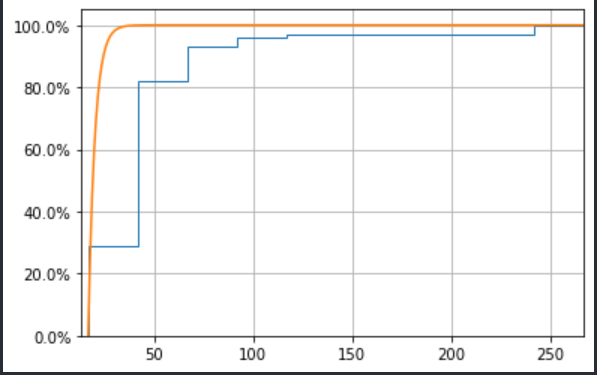
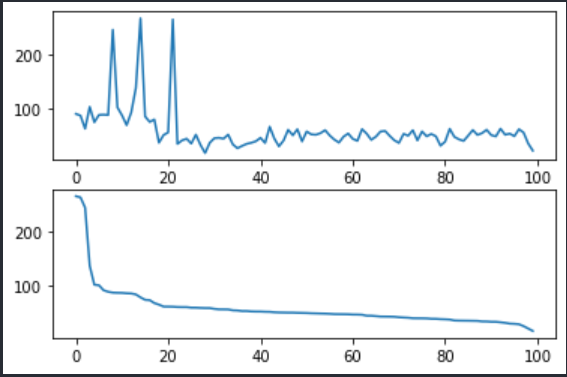


图 14 原始尾延迟

原始尾延迟见上图所示，进行了模拟对冲请求后的尾延迟见下图所示。

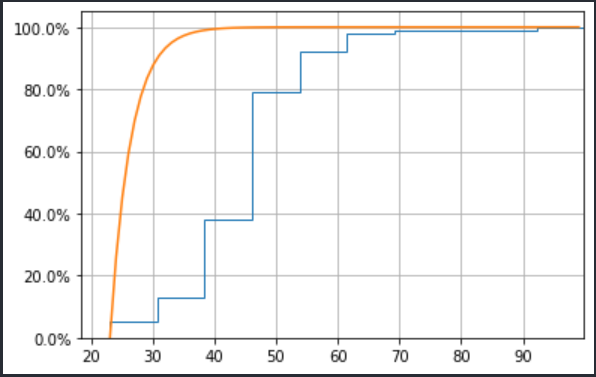
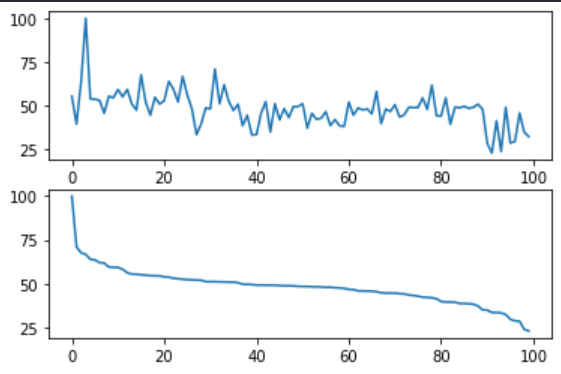


图 15 进行了模拟对冲请求后的尾延迟

可以看到，此时尾延迟相较于原始的减少了。但是由于都是单机上的测试，模拟存在一定的误差。不过模拟的结果可以说明对冲请求是有利于缓解尾延迟现象的。

同样，使用相同的策略来模拟关联请求。

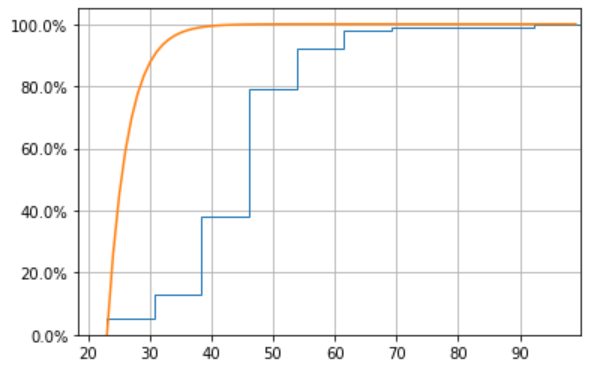
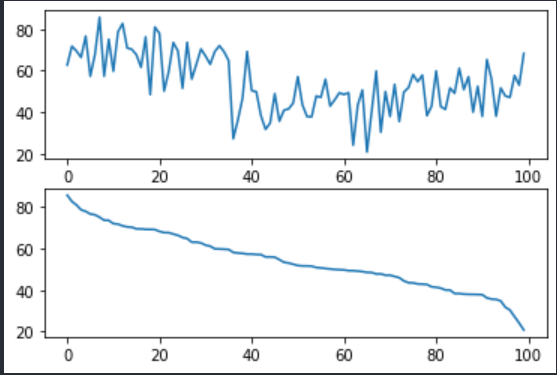


图 16 进行了模拟关联请求后的尾延迟

可以看到，此时的尾延迟相对于原始的减少了，同样，由于是单机测试，结果只能说明关联请求是可以缓解尾延迟现象的，但是不能定量。

因此，综上所述，对冲请求和关联请求是可以缓解尾延迟现象的。