实验一：系统搭建

系统：windows

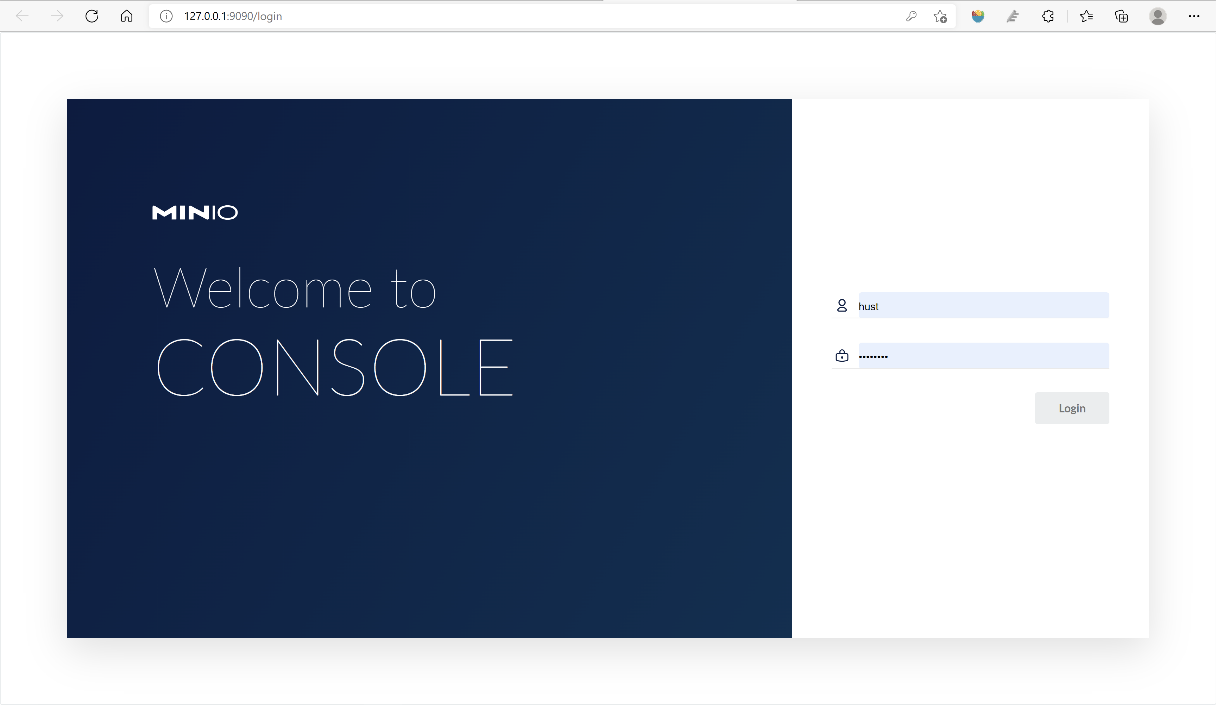
服务端：minio

客户端：osm

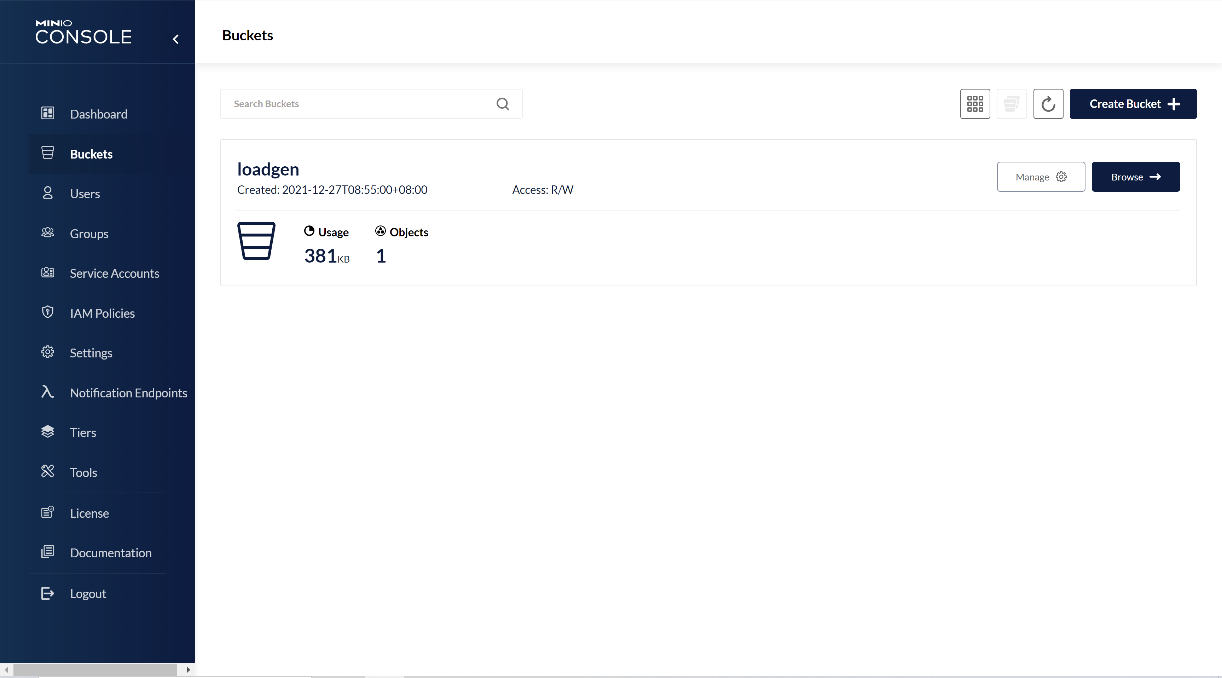
性能测试：s3bench,boto3

实验步骤：

1. 到指定地址下载minio osm s3bentch的可执行文件（分别为minio.exe osm.exe s3bench.exe）
2. 将课程github项目克隆到本地，并将1中的可执行文件放入目录下
3. 开启一个终端，在项目目录下运行run-minio.cmd脚本文件，即可按脚本文件中的参数运行minio，之后可在本地9090端口看到minio的控制台

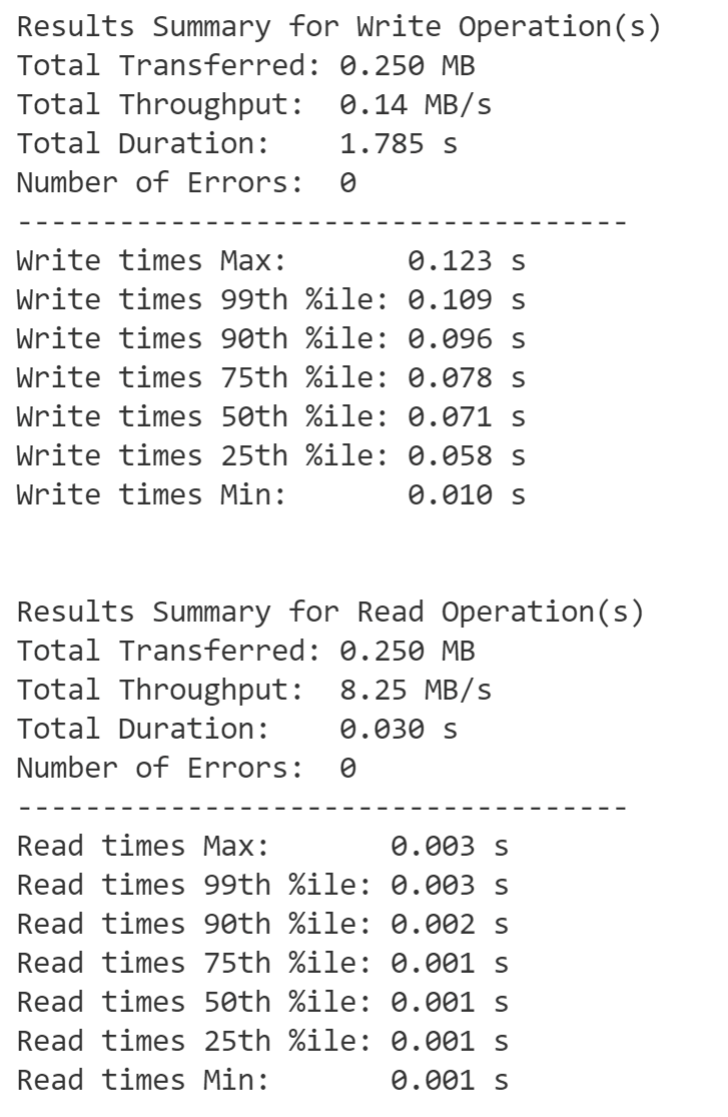


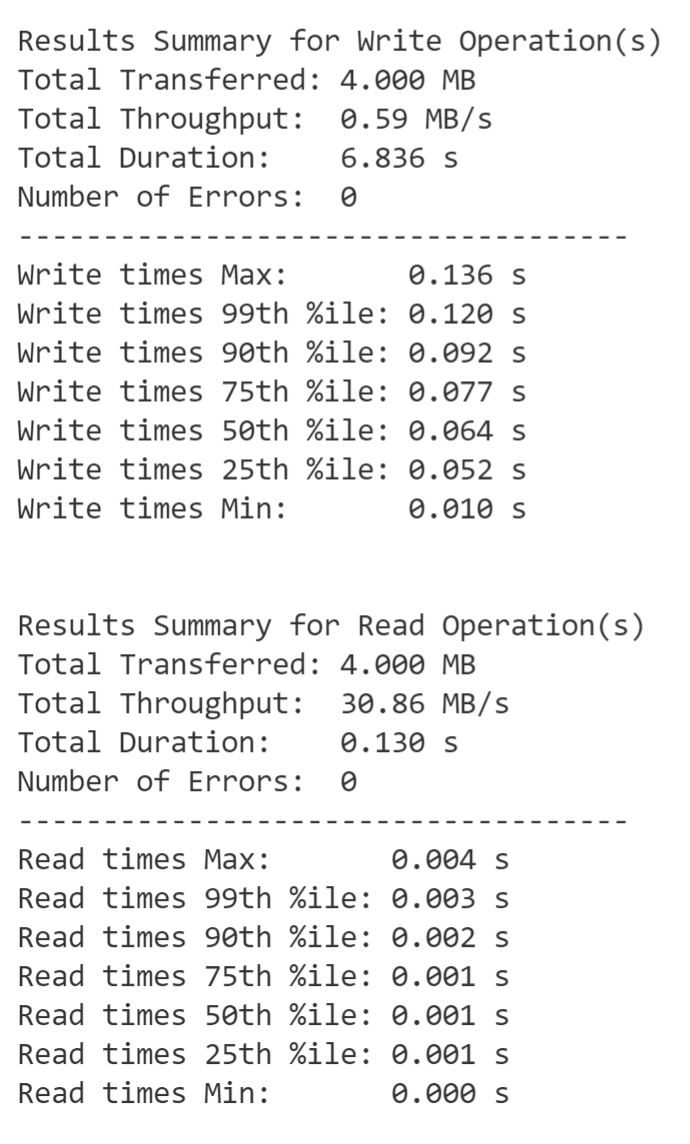
1. 执行.\osm.exe mc loadgen创建一个bucket



实验二：性能观测

使用s3bench测试，直接执行项目中的脚本run-s3bench.cmd，使用脚本中的参数测试得到如下结果

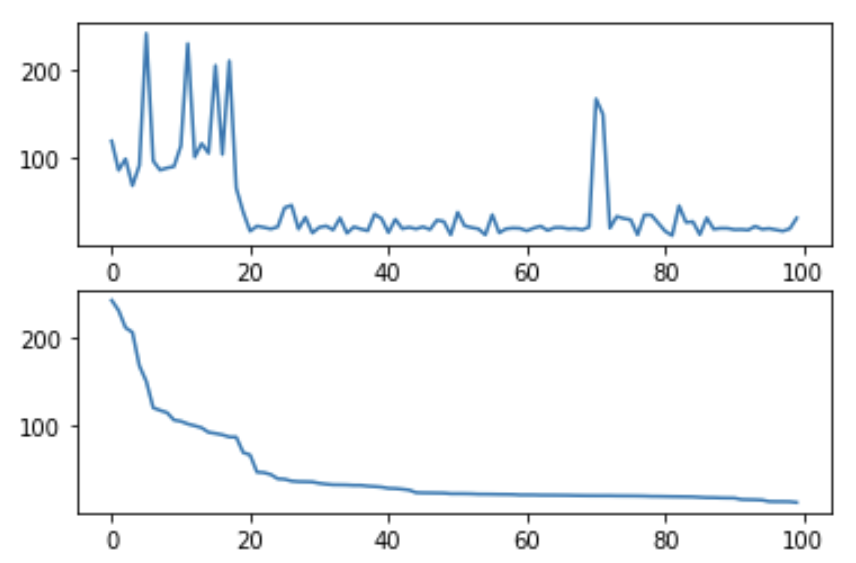


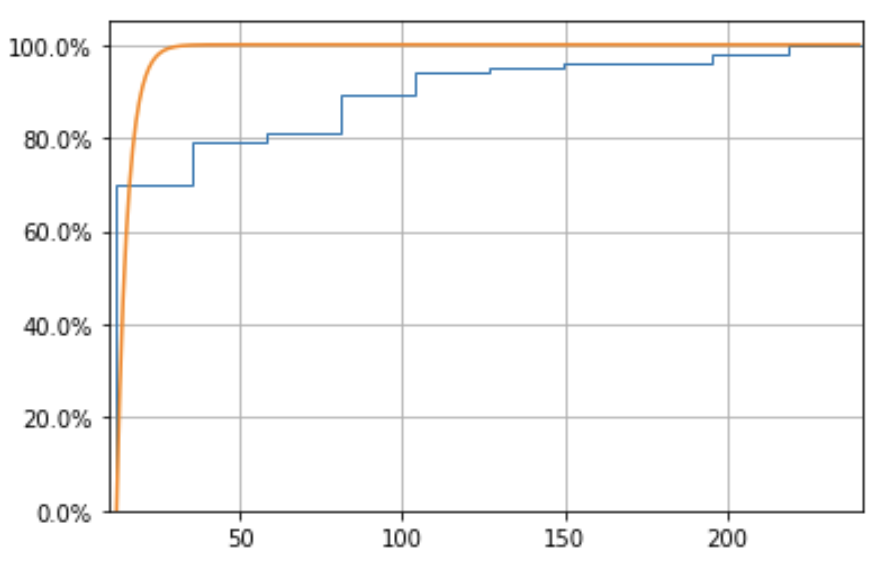
更改参数样例数为1024，大小为4096测试结果如下

实验3 尾延迟挑战

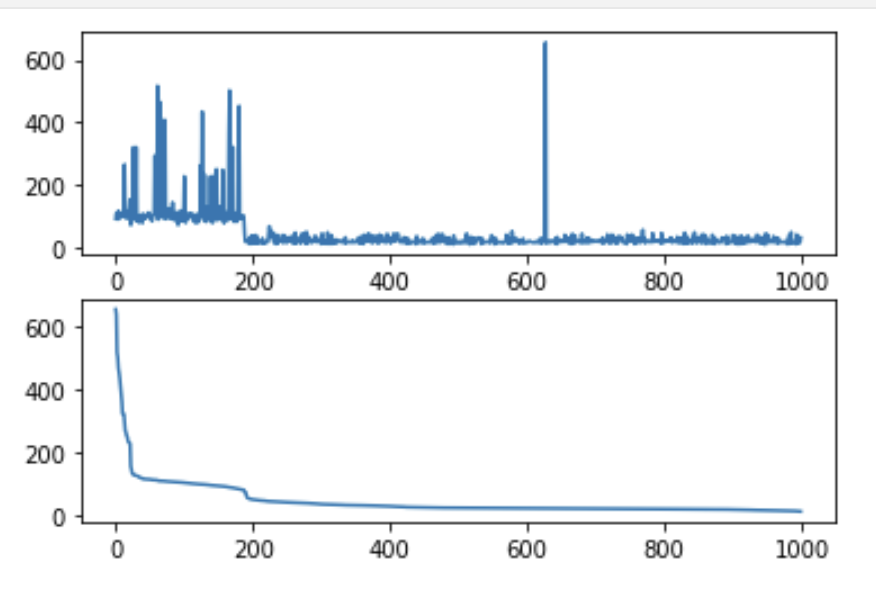
针对100，1000两种规模的数据的写，测试其延迟，并用排队论拟合实测数据，以下分别为无处理，关联请求，对冲请求下的实验结果，每次测试有2个结果，第一个为延迟测试的直接结果，第二个为将结果排序后绘制的曲线，第三个图是延迟分布的阶梯图和排队论拟合的结果

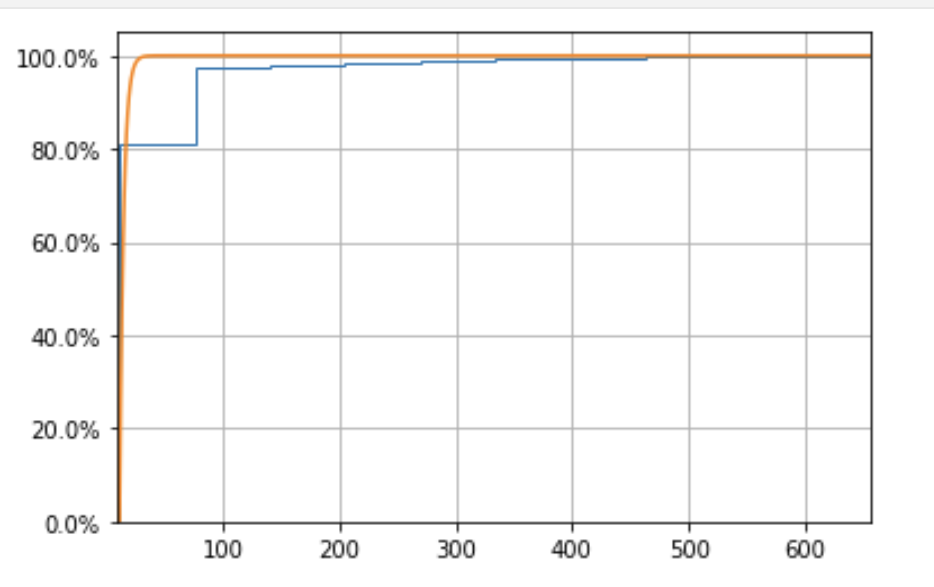
无处理100



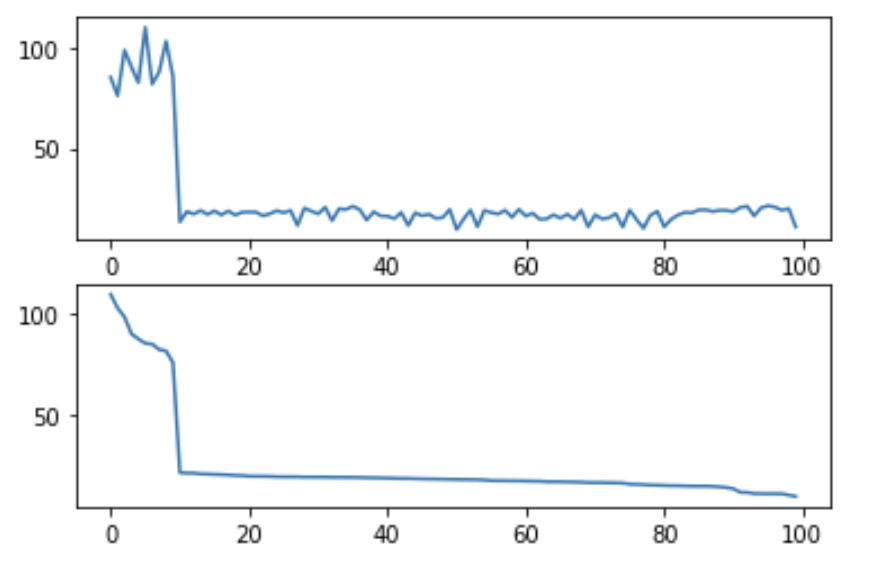


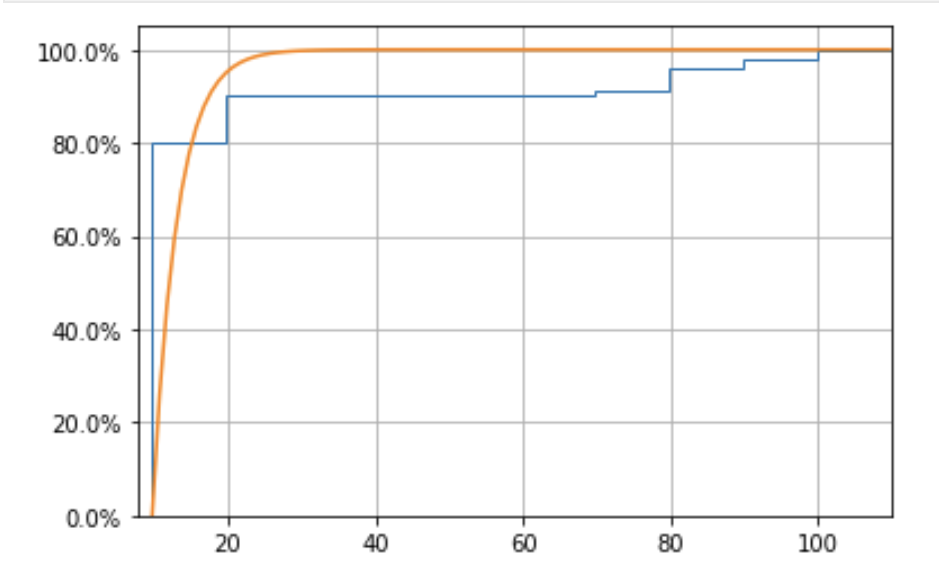
无处理1000



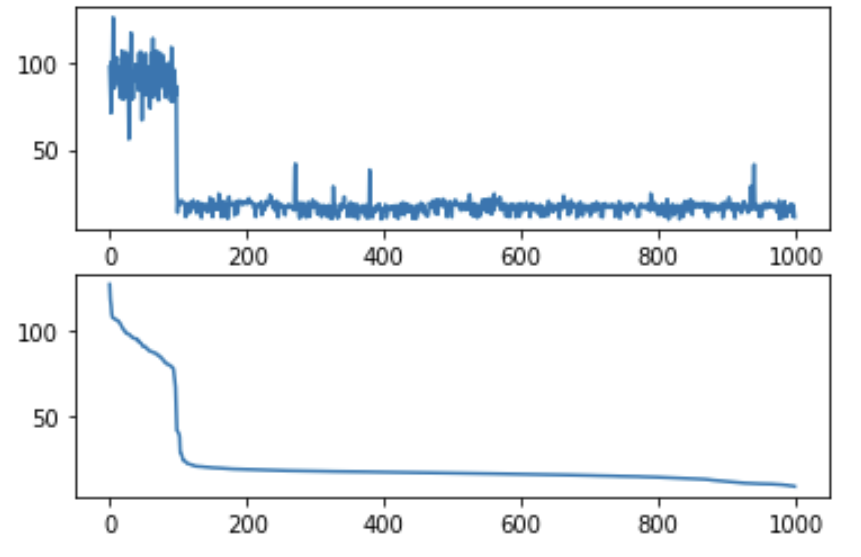


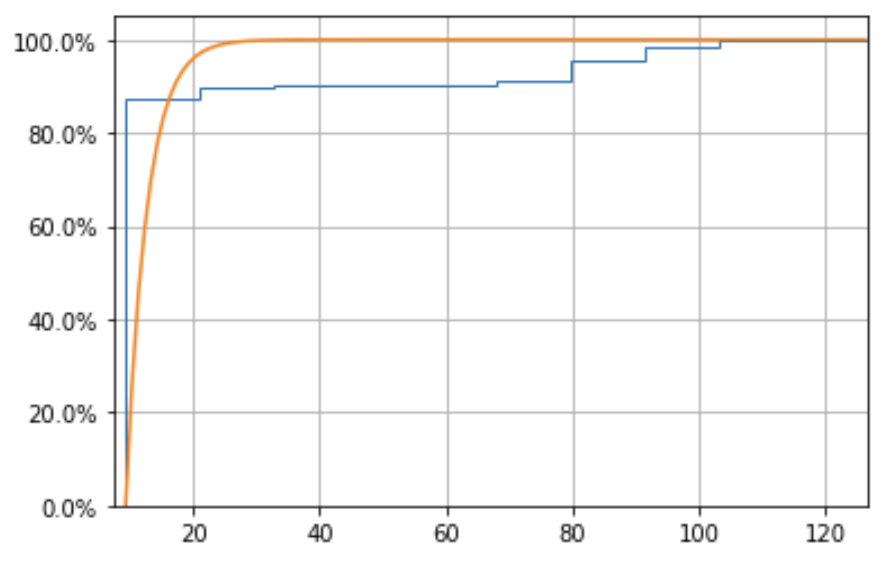
关联100



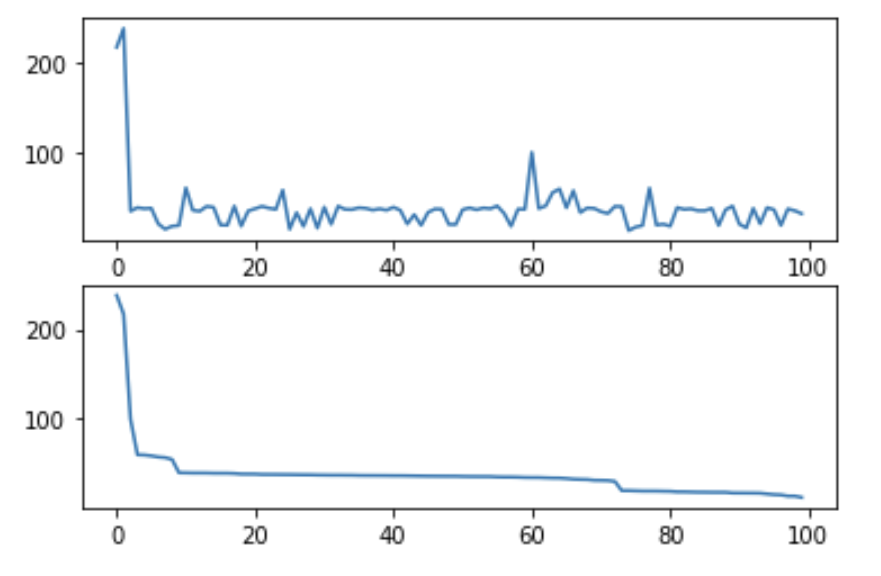


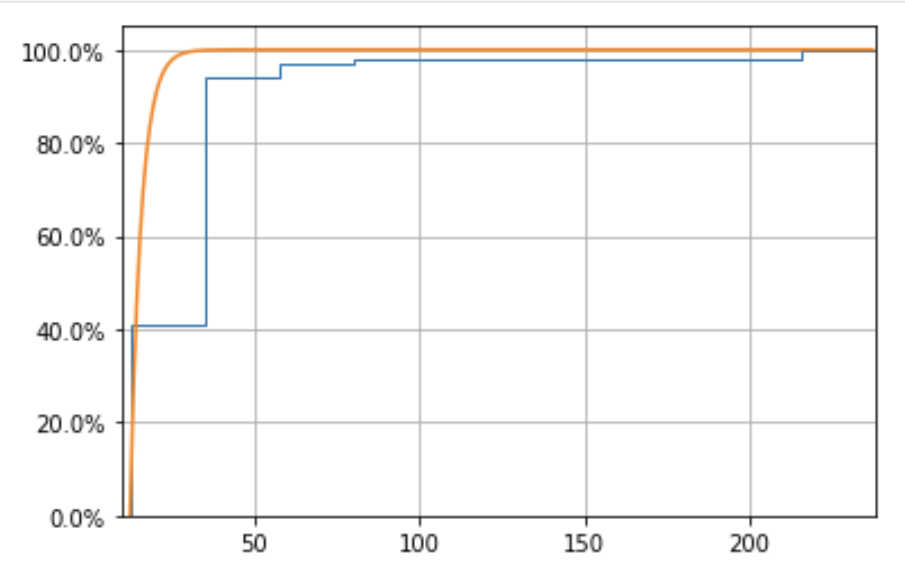
关联1000



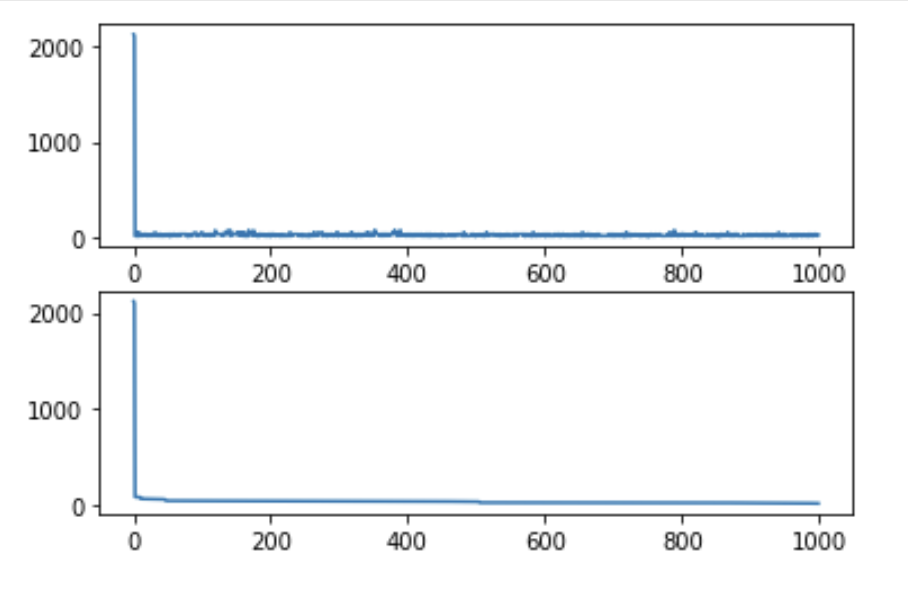


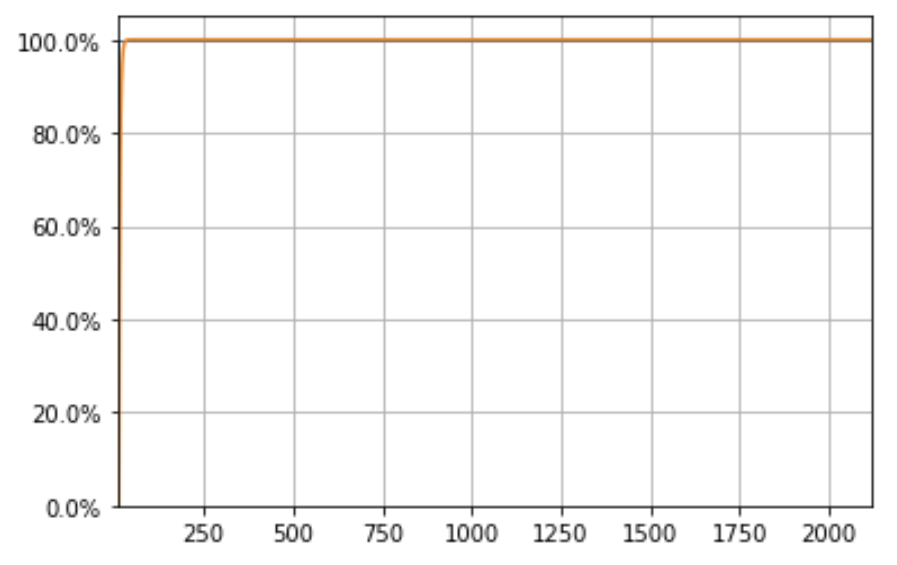
对冲100





对冲1000





从测试结果可以看出，关联请求和对冲请求的尾延迟相对于无处理都有显著降低。