目录

[一、 系统搭建 2](#_Toc92567054)

[实验环境 2](#_Toc92567055)

[服务器搭建：Minio 2](#_Toc92567056)

[客户端安装：Minio Client 3](#_Toc92567057)

[二、 性能观测 4](#_Toc92567058)

[安装s3bench工具 4](#_Toc92567059)

[使用s3bench测试 6](#_Toc92567060)

[三、 尾延迟挑战 7](#_Toc92567061)

[原始尾延迟 7](#_Toc92567062)

[关联请求Tied Request 8](#_Toc92567063)

# 系统搭建

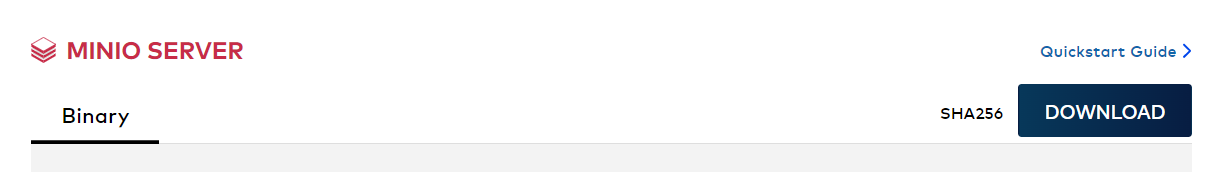
## 实验环境

操作系统：Windows10

## 服务器搭建：Minio

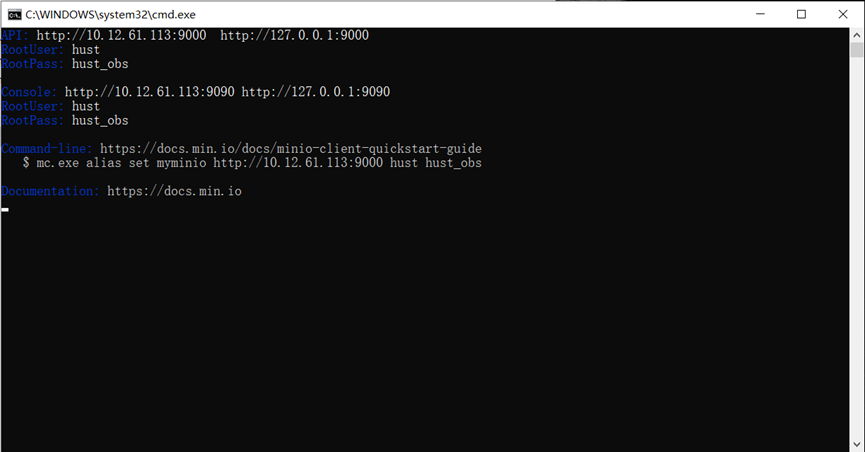
第一步，下载Minio Server

最新版网址： <https://minio.io/downloads.html>，进入网页，点击download。



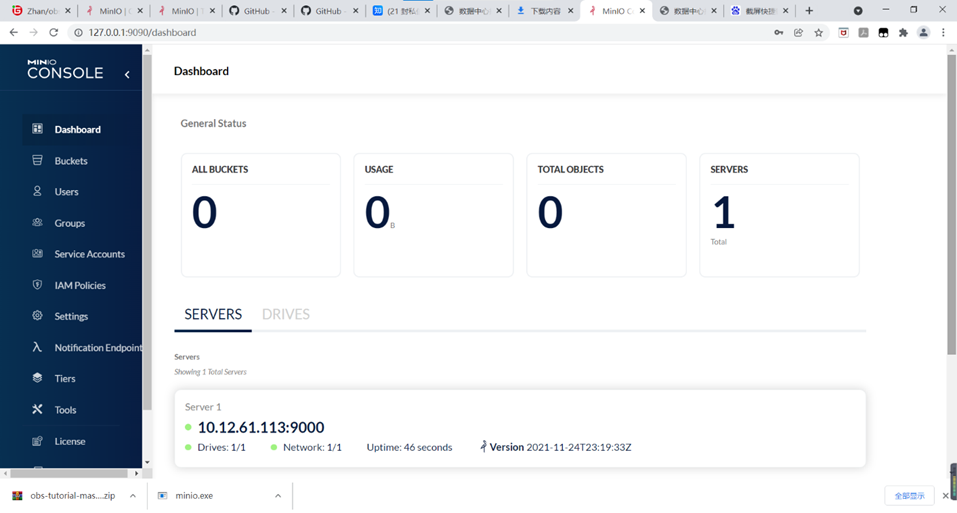
下载minio.exe后，放入实验文件夹。

第二步，执行实验文件夹中的run-minio.sh，配置账号密码。



如图所示，用户名为：hust；用户密码为：hust\_obs。服务器页面端口为：<http://127.0.0.1:9000>。

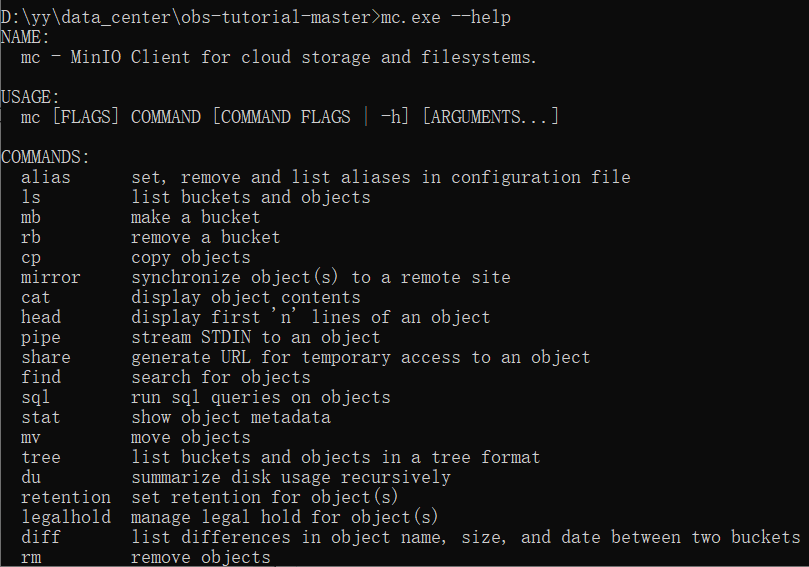
第三步，登录页面，管理服务端。



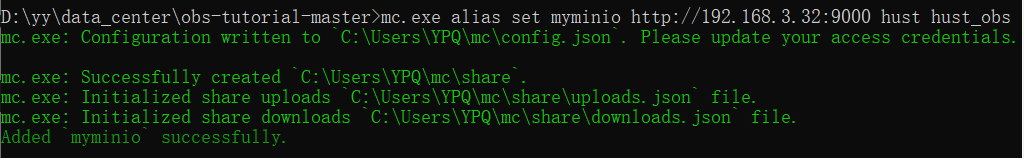
可以看见刚刚配置的1个服务器。

## 客户端安装：Minio Client

第一步，在<https://docs.min.io/>上下载mc.exe，放入实验文件夹下。打开cmd窗口，输入mc.exe –help，查看命令。



第二步，输入mc.exe alias set myminio http://192.168.3.32:9000 hust hust\_obs，执行结果如下：



第三步，创建bucket：输入mc mb play/mybucket



第四步，上传文件到bucket中：输入mc cp latency.csv play/mybucket，将latency.csv上传到mybucket中。



第五步，列出bucket中对象：输入mc ls play/mybucket



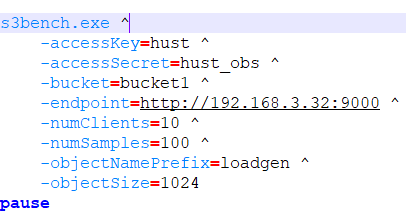
第六步，删除桶和对象：输入mc rm play/mybucket

# 性能观测

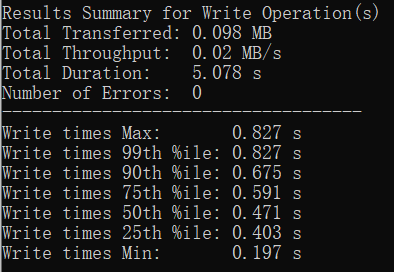
## 安装s3bench工具

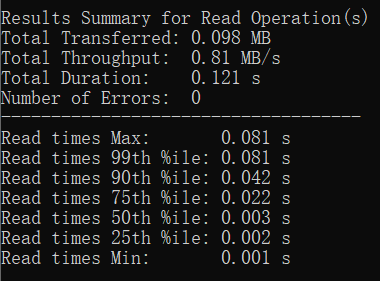
第一步，下载s3bench.exe，放在实验文件夹下。

第二步，修改run-s3bench.cmd中的参数，将线程数修改为10，对象大小修改为1024。



运行结果：



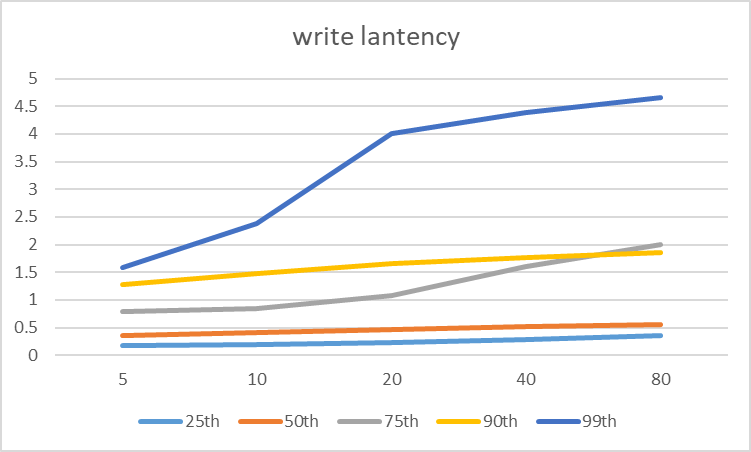


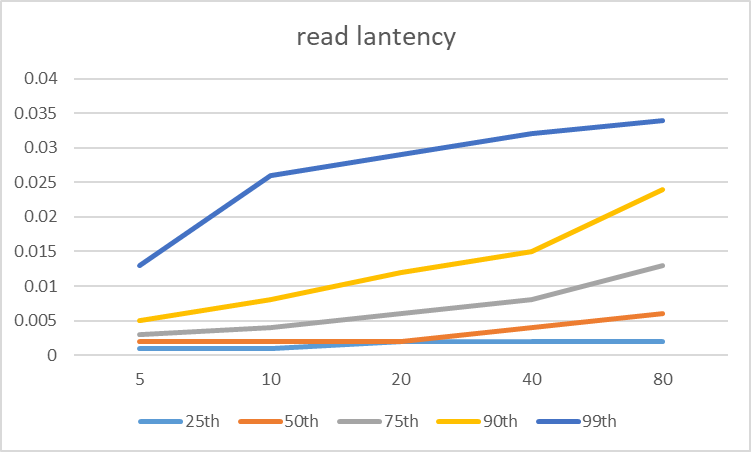
## 使用s3bench测试

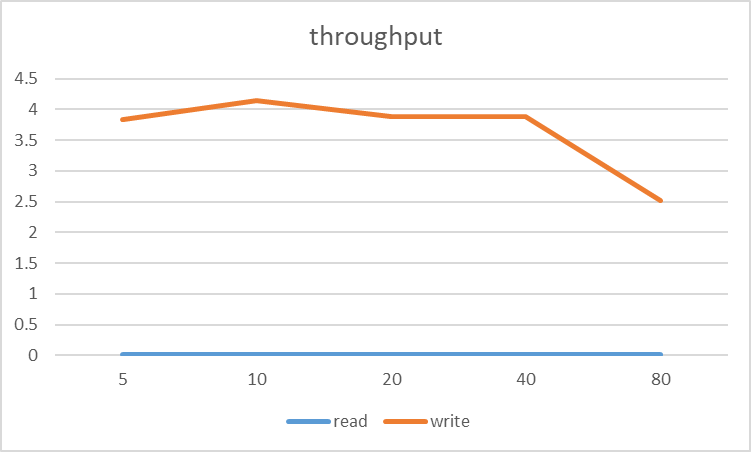
在本次实验中，我们测试服务器数对延迟和吞吐量的影响。

第一步，将对象大小设置为1024，样本数设置为100，服务器数分别设置为5、10、20、40、80，分别记录读与写的延迟与吞吐量。

第二步，将实验结果绘制成图。如下分别为写延迟随服务器数量变化图、读延迟随服务器数量变化图、读写吞吐量随服务器数量变化图。





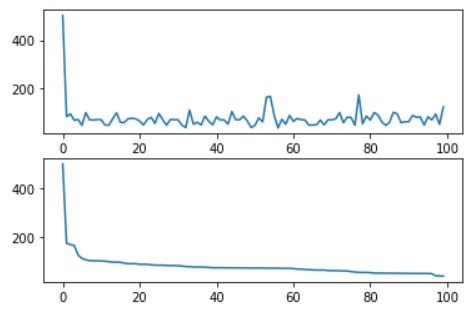


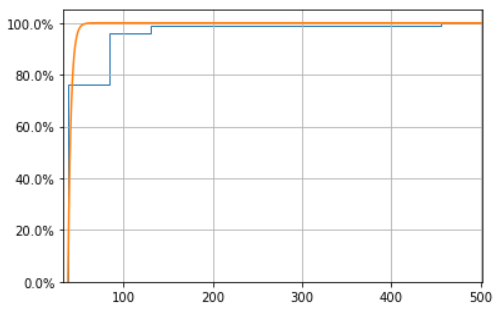
观察可得以下结论：（1）并发数越大，读写延迟越大。（2）对于该环境，当并发数低于40时，写吞吐量基本不变；当并发数高于40时，写吞吐量随服务器数量而降低。

因此为保证吞吐率达到最佳，在该环境下服务器数最好不要超过40。

# 尾延迟挑战

## 原始尾延迟

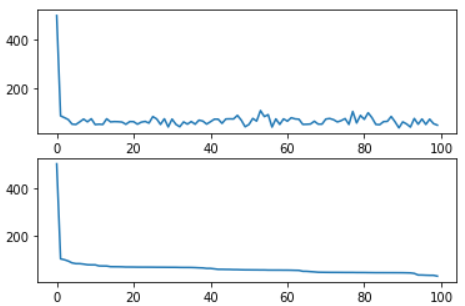


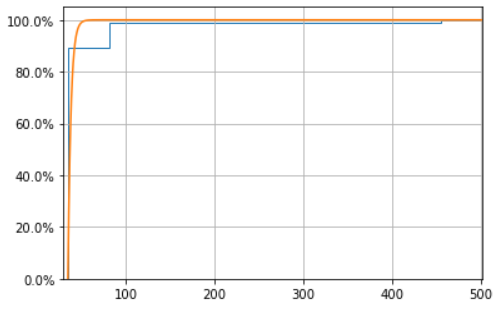


## 关联请求Tied Request

原理：多个服务器同时发出同一个请求，哪个延迟短使用哪个。

实验结果：





由实验结果可知，关联请求尾延迟结果相对原始结果有较大改善，99%从约130ms下降为约75ms