

**数据中心技术实验报告**

|  |  |
| --- | --- |
| **学 院** | **计算机科学与技术** |
| **班 级** | **2021硕** |
| **姓 名** | **胡可心** |
| **学 号** | **M202173878** |
| **实验名称** | **对象存储入门实验** |

2021 **年** 1 **月** 7

# 一 环境搭建

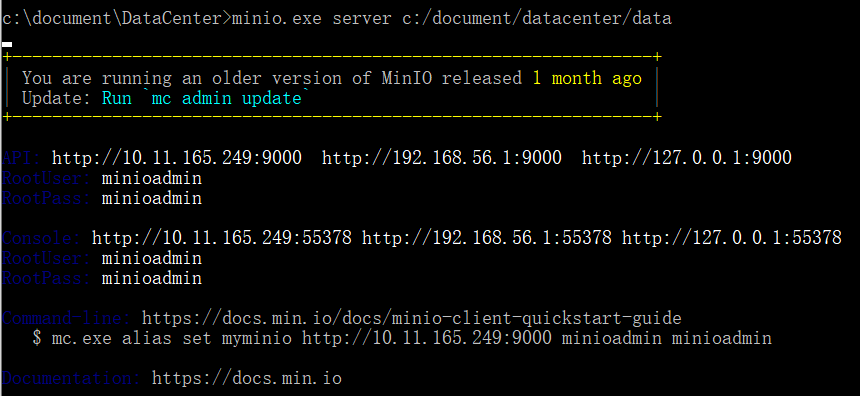
1. ） 实验环境

|  |  |
| --- | --- |
| 操作系统 | Windows10 |
| 实验语言 | Python |
| 实验平台 | Anaconda |
| 服务端 | Minio |
| 客户端 | Minio Client |

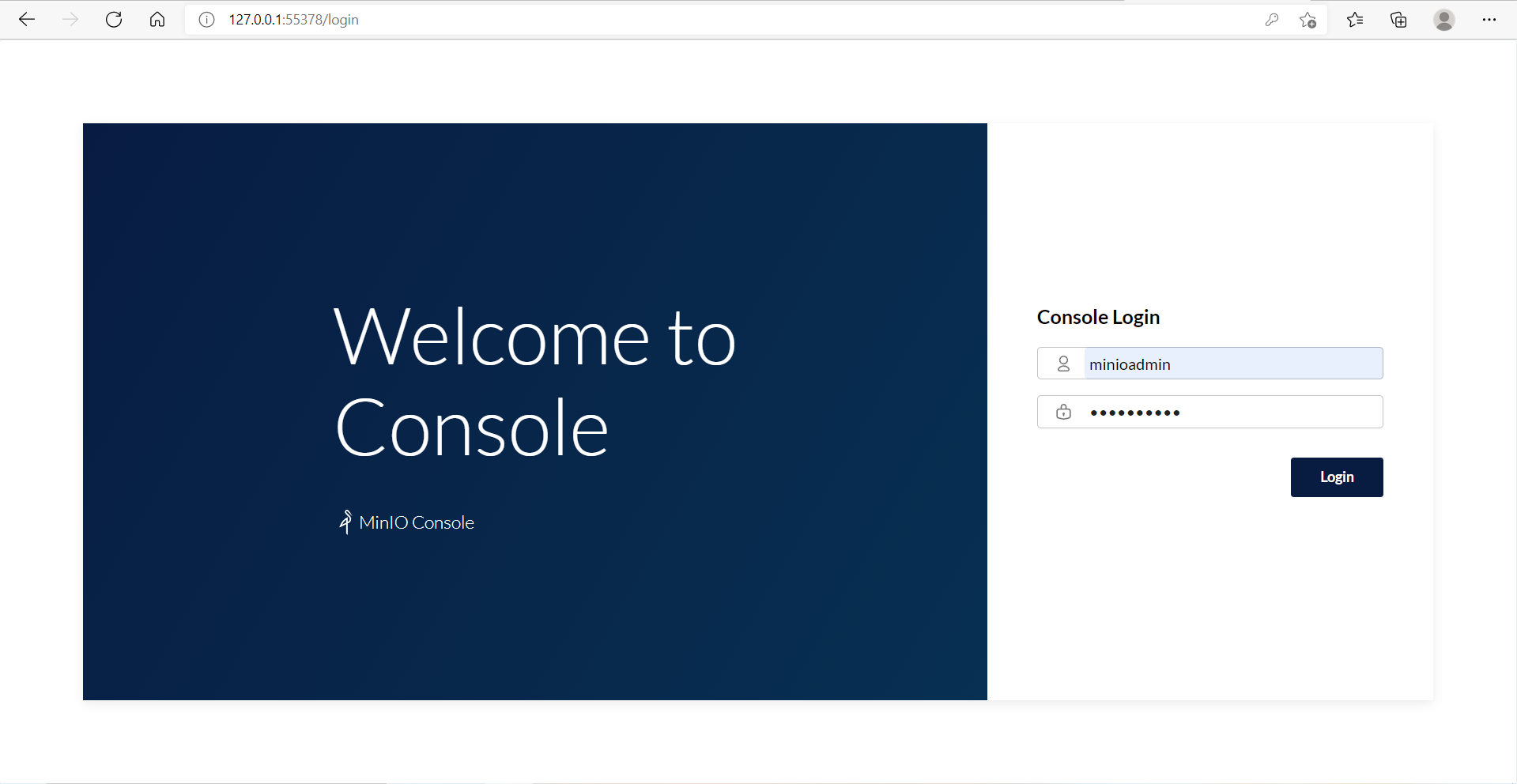
1. 实验步骤

在[https://minio.io/downloads.html](https://gitee.com/link?target=https://minio.io/downloads.html)下载并安装Minio

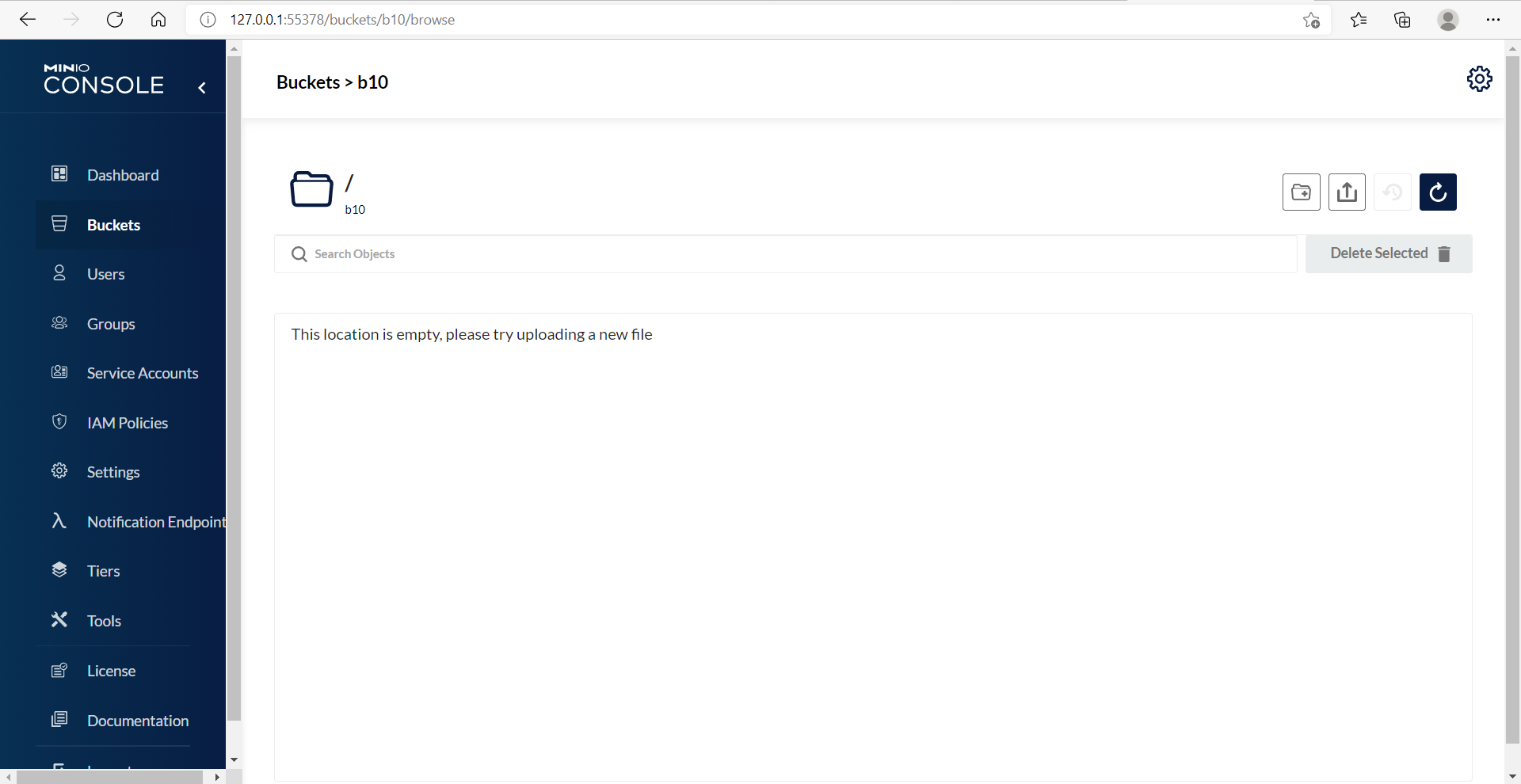
通过命令行在下载的minio.exe地址文件下运行服务



输入端口地址并输入上面图片中给定的用户名与密钥进行连接



尝试创建一个桶

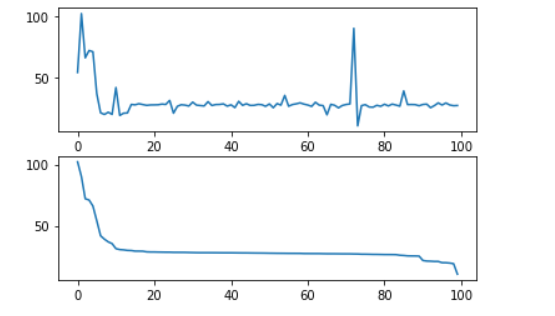


**二 性能观测**

参照<https://github.com/cs-course/obs-tutorial>的课程实验指导，测试不同并发任务数量以及不同文件大小下的性能。

修改latency-collect.ipynb中的代码，并运行，将延迟记录到本地CSV文件中，并修改运行latency-plot.ipynb中的代码，当并发线程数量为1，任务数量为100时，观察并分析结果。

延迟分布情况如下图所示：



# 用排队论来对测试的数据进行拟合，结果如下图所示：

# 

# 通过观察以上实验结果可以发现，90%的任务可以在25ms内完成，仍有少部分任务延迟时间较长。

**三 尾延迟**

通过实验二可以观测到尾延迟的现象，在本阶段使用中，使用对冲的方法来减少延迟，并对其进行实验验证。

本实验设计的对冲方法思想是，通过在每个任务完成后对延迟进行判断,如果延迟大于25ms，则重新进行该任务，并将重新进行的任务的延迟作为改作业的延迟进行记录，如果延迟在25ms内，则直接记录延迟。

对上述思想进行实验，修改latency\_collect代码，具体如下所示：

latency = []

failed\_requests = []

futures = []

with tqdm(desc="Accessing S3", total=100) as pbar:      # 进度条设置，合计执行 100 项上传任务 (见 submit 部分)，进度也设置为 100 步

    with ThreadPoolExecutor(max\_workers=1) as executor: # 通过 max\_workers 设置并发线程数

        for i in range(100):

            first = executor.submit(

                arrival\_rate\_max,

                session.resource('s3', endpoint\_url=local\_s3),i);

            #判断第一次的延迟是否符合条件

            if(first.result()>25):

                #如果不符合，则重新进行该任务，并记录延迟

                futures.append( executor.submit(

                arrival\_rate\_max,

                session.resource('s3', endpoint\_url=local\_s3),i));

            else:

                #否则直接记录

                futures.append(first);

        for future in as\_completed(futures):

            if future.exception():

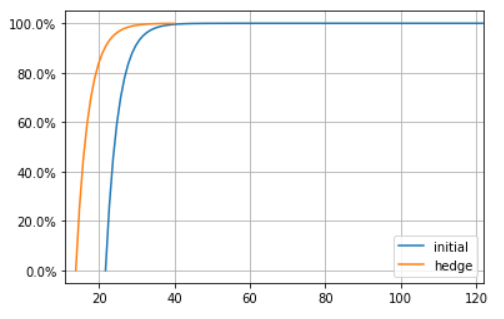
                failed\_requests.append(futures[future])

            else:

                latency.append(future.result()) # 正确完成的请求，采集延迟

            pbar.update(1)

对latency\_plot进行修该，结果如下图所示：



可以发现延迟情况相较于之前要好许多。