華中科技大學

课程设计报告

课程名称：数据中心技术

专业班级：国光硕 2104 班

学　　号：M202173483

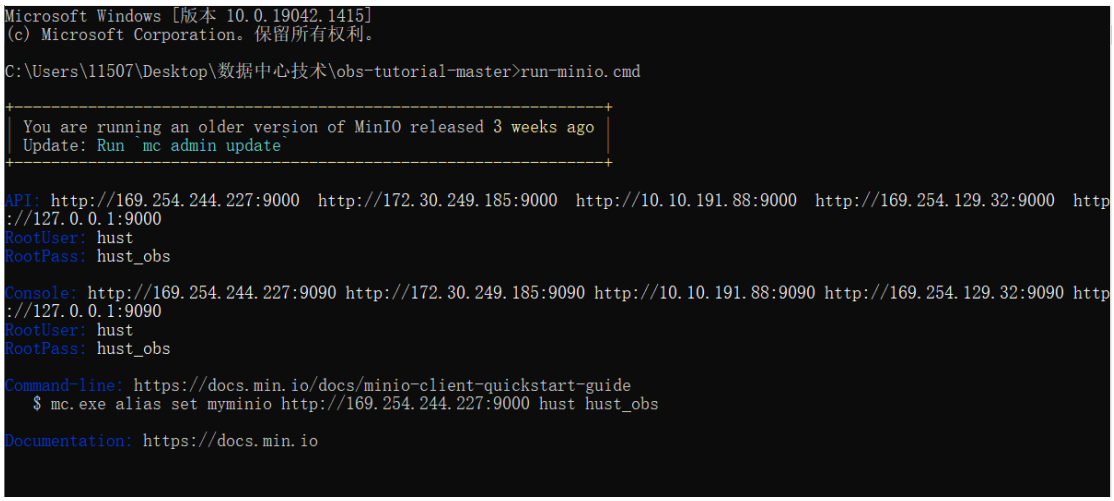
姓　　名：孙依茗

报告日期：2021 年 12 月 28 日

实验一：系统搭建

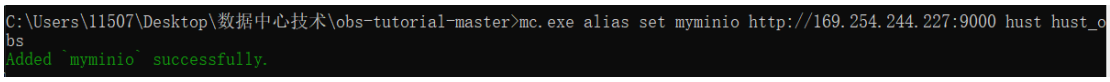
- **服务端：**

  安装 Minio，通过官网下载 Windows 版本的 minio.exe，并将其放入 run-minio.cmd 所在的文件夹中，在这个文件夹下启动命令提示符窗口，使用 run-minio.cmd 脚本运行客户端。客户端启动的截图如下：



- **客户端：**

  下载客户端 mc.exe 到同一个文件夹中，再打开一个 cmd 窗口，输入 `mc.exe alias set myminio http://169.254.244.277:9000 hust_obs`，也即启动服务器端后的页面中所给出的命令。执行成功后截图如下：



- **基本操作：**

  ✓ **创建桶：**使用在命令符窗口使用命令 `mc mb D:/minio/picture`，在 D:/minio 文件夹下创建桶 picture。桶创建成功的截图如下：

- ✓ **上传本地文件到桶中：**使用命令 `D:/source/test001.jpg D:/minio/picture`，将本地文件夹 D:/source 中的 test001.jpg 上传到创建的桶 picture 中，上传成功截图如下：

```
C:\Users\11507\Desktop\数据中心技术\obs-tutorial-master>mc cp D:/source/test001.jpg D:/minio/picture
...urce/test001.jpg:  12.12 MiB / 12.12 MiB [=============================================================] 54.72 MiB/s 0s
```

- ✓ **列出桶中所有对象：**使用命令 `mc ls D:/minio/picture` 查看当前桶中的所有对象。

```
C:\Users\11507\Desktop\数据中心技术\obs-tutorial-master>mc ls D:/minio/picture
[2021-12-23 20:25:41 CST]   12MiB test001.jpg
```

- ✓ **删除桶和对象：**使用命令 `mc rm D:/minio/picture`，将桶和其中的对象全部删除。

```
C:\Users\11507\Desktop\数据中心技术\obs-tutorial-master>mc rm D:/minio/picture
Removing `D:/minio/picture`.
```

实验二：性能观测

选择 **S3 Bench** 测评工具，下载 s3bench.exe 文件，更改 run-s3bench.cmd 中的对象大小、用户个数（线程数），来测试线程数量和对象大小对性能的影响，在命令提示符窗口中运行 run-s3bench.cmd，运行截图如下，

```
Results Summary for Write Operation(s)
Total Transferred: 256.000 MB
Total Throughput:  108.90 MB/s
Total Duration:    2.351 s
Number of Errors:  0
------------------------------------
Write times Max:       0.245 s
Write times 99th %ile: 0.241 s
Write times 90th %ile: 0.133 s
Write times 75th %ile: 0.115 s
Write times 50th %ile: 0.082 s
Write times 25th %ile: 0.062 s
Write times Min:       0.038 s
```

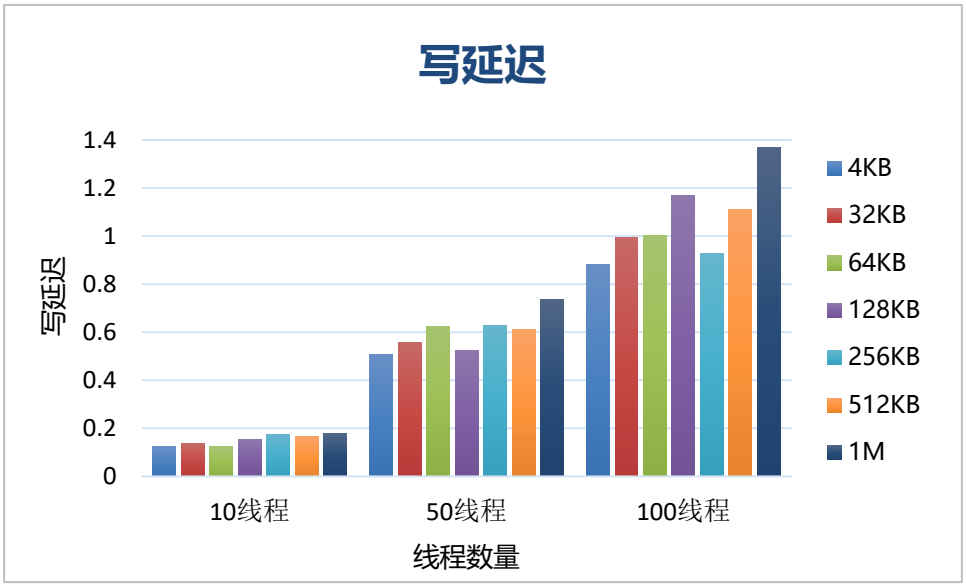
上图所示为写操作的输出，可以看出，总传输量为 256MB，总吞吐量为 108.90MB/s，写操作中最大延迟为 0.245s，最小延迟为 0.038s。大约有 99%的文

件写延迟不超过 0.241s。

```
Results Summary for Read Operation(s)
Total Transferred: 256.000 MB
Total Throughput:  897.92 MB/s
Total Duration:    0.285 s
Number of Errors:  0
-----------------------------------
Read times Max:       0.031 s
Read times 99th %ile: 0.026 s
Read times 90th %ile: 0.016 s
Read times 75th %ile: 0.012 s
Read times 50th %ile: 0.010 s
Read times 25th %ile: 0.008 s
Read times Min:       0.003 s
```

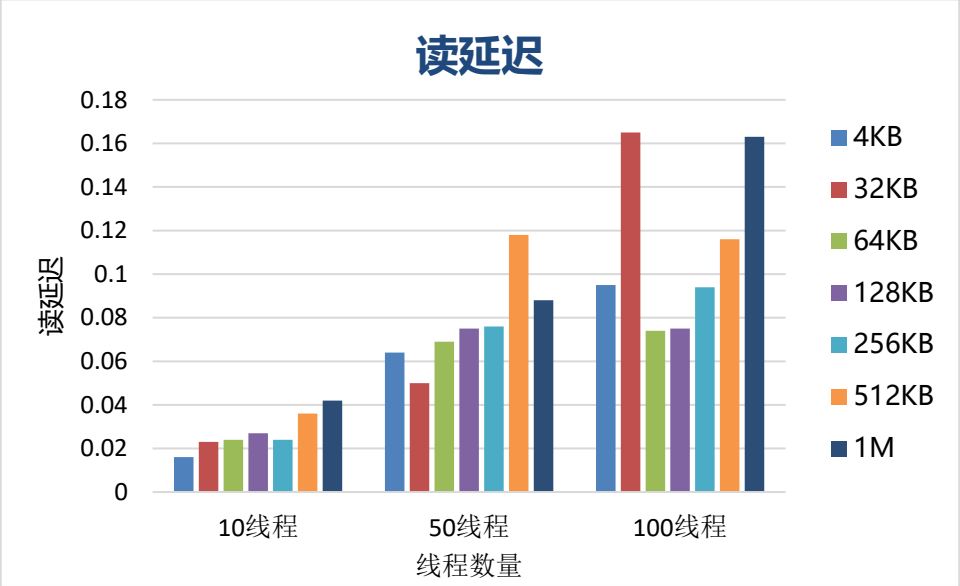
上图所示为读操作的输出，可以看出总传输量为 256MB，总吞吐量为 897.92MB/s，最大的读延迟为 0.031s，最小读延迟为 0.003s，99%的文件读操作的延迟不超过 0.026s。

测试结果统计如下：



可以看出，随着对象大小增加、线程数量的增加，写延迟逐渐变大。但是还存在一些特殊情况，例如对象大小为 128KB 和 256KB 时，在 50 个线程同时写入和在 100 线程同时写入时写延迟情况不一样。

同样，读延迟的结果如下所示：



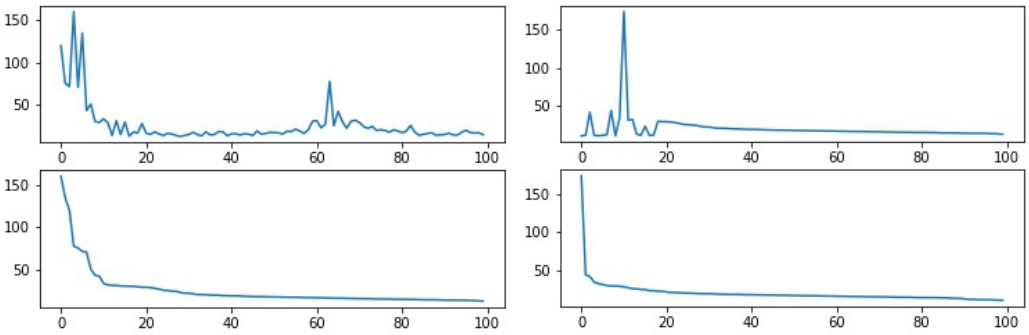
可以看出，延迟也基本是随着对象大小的增加以及线程数量的增加导致延迟增大。同样也存在一些特殊情况，我猜测出现反常情况（如对象大小为 32KB 在 100 个线程下延迟非常大）是因为测试时电脑运行其他的程序导致测试结果反常。

因此，基于上述的实验，我认为作为应用的存储，应该在不同的机型、不同的负载条件下测试后，选择最合适的对象大小和并行的线程数量，将应用的写入数据拆分成合适的对象大小进行读写，从而达到最低的延迟和最高的读写效率。如果需要保证服务质量，则需要限制并行客户的数量。

实验三：尾延迟挑战

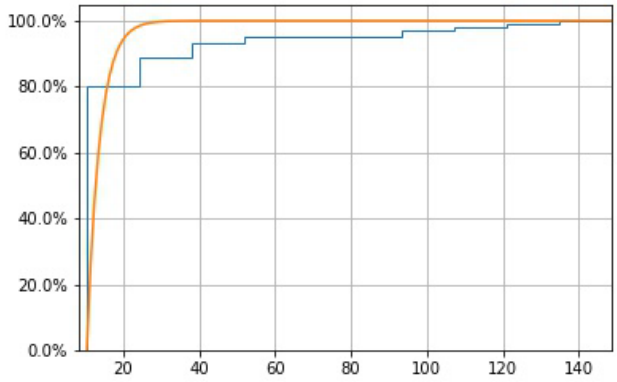
- **对冲请求：**

  对冲请求是指同时向多个服务器发送相同的请求，然后使用返回最快的服务器的结果。对冲请求客户端先向自己认为最合适的服务器发送一个请求，在短暂的延迟后再发送第二个请求，一旦收到第一个返回的结果，客户端就取消剩余未完成的请求，这种方式在略微增加负载的同时，实现了减少大多数延迟的效果。一种对冲请求的方式是第一个请求完成，且该请求的延迟超过预期延迟的 95%，则再发送第二个请求。本实验中在首次运行 100 个对象的上传后，统计其中延迟大于 20ms 的请求，再次发送这些延迟较大的请求，从而减少尾延迟。实验结果统计图如下：



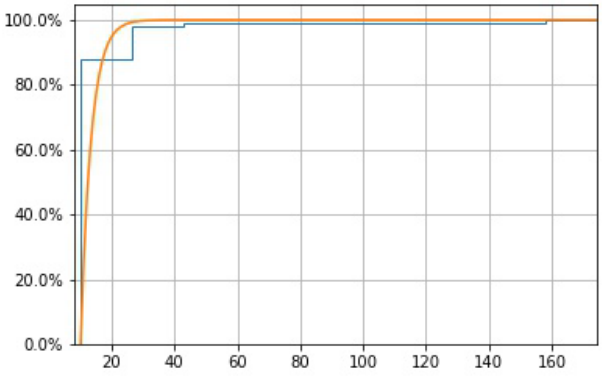
初始延迟统计图　　对冲请求后延迟统计图

从图中可以看出，在初始情况下，存在一些请求的延迟远大于其他请求，也即尾延迟现象；而进行对冲请求后，只有几个请求延迟大的。



初始延迟统计和排队论预测图

根据上述的对冲请求的原理，应选择第一次请求时延迟大于预期延迟 95%的请求进行第二次请求的发送，上图是第一次请求发出所得到的延迟，取 30ms 作为界限，延迟大于 30ms 的请求再次发送请求并执行，获得以下统计图：



初始延迟统计和排队论预测图

可以看出，在进行对冲请求后，原本延迟很大的请求的延迟变小了，大部分请求的延迟变得可以接受。