

**2016** 级

《物联网数据存储与管理》课程

**实 验 报 告**

**姓 名 陈奕芃**

**学 号 U201614885**

**班 号 物联网1601班**

**日 期 2019.05.25**

**目 录**

[一、实验目的 1](#_Toc9857773)

[二、实验背景 1](#_Toc9857774)

[三、实验环境 1](#_Toc9857775)

[四、实验内容 2](#_Toc9857776)

[4.1 对象存储技术实践 2](#_Toc9857777)

[4.2 对象存储性能分析 3](#_Toc9857778)

[五、实验过程 3](#_Toc9857779)

[六、实验总结 12](#_Toc9857780)

[参考文献 13](#_Toc9857781)

# 一、实验目的

1. 熟悉对象存储技术，代表性系统及其特性；

2. 实践对象存储系统，部署实验环境，进行初步测试；

3. 基于对象存储系统，架设实际应用，示范主要功能。

# 二、实验背景

近年来，随着软硬件和计算机的告诉发展，数据存储的几个重要的需求逐渐显现：①：⾼容量，海量，且可扩展。②：⾼性能，快速的访问、定位。③：⾼可⽤，不怕个别磁盘失效，甚⾄整个中⼼被毁。④：低成本：TCO= D(部署) + M(维护) \* t(时间)。我们这次实验所用的软件有：Minio，MinioClient，S3Proxy，S3cmd，S3 benchmark。

Minio：Minio是Apache License v2.0下发布的对象存储服务器。它与Amazon S3云存储服务兼容。它最适合存储非结构化数据，如照片，视频，日志文件，备份和容器/ VM映像。对象的大小可以从几KB到最大5TB Minio服务器足够轻，可以与应用程序堆栈捆绑在一起，类似于NodeJS，Redis和MySQL。Minio Cloud Storage 是一款轻量级对的象存储服务，兼容Amazon S3。通过简单的命令，就可以搭建服务器，能实现可以通过浏览器访问的简易网盘功能。利用Minio客户端MC可以在命令行中对云中的内容进行管理。

S3Proxy实现 S3 的API，还可以实现从S3到其他多种云服务的翻译，可以通过使用本地文件系统进行无Amazon测试，可嵌入Java应用程序等功能。

S3cmd是一款免费的命令行工具和客户端，用于在Amazon S3和其他使用S3协议的云存储服务提供商中上传，检索和管理数据。

s3-benchmark是Wasabi为对象执行S3操作（PUT，GET和DELETE）而提供的性能测试工具。除了桶配置之外，还可以针对不同的测试给出对象大小和线程数。该测试工具基于用于测试性能的Nasuni性能基准测试方法。不同的云存储提供商。

# 三、实验环境

实验前的软硬件环境如下表3.1所示。

|  | | **Minio/mc实验及S3proxy/s3cmd实验** |
| --- | --- | --- |
| 硬件环境 | 系统类型 | x64-based PC |
| 处理器 | Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz |
| 软件环境 | 操作系统 | Ubuntu 16.04 LTS |
| 其他 | go version go 1.11.5 linux/amd64  java version "1.8.0\_191"  Python 2.7.12 |

表3.1 软硬件环境

# 四、实验内容

本次实验的主要内容如下：

1. 下载git客户端，对git进行了解和熟悉，使用git上传文件。
2. 配置实验环境，安装Go语言运行环境，docker等软件。
3. 对对象存储系统进行实践，采用目前几种典型的服务，如minio/mc，s3proy/s3cmd配置服务器端和客户端，并进行简单的创建或删除bucket、上传或删除文件等操作。
4. 对对象存储系统进行测试，采用s3-benchmark进行负载测试。

## 4.1 对象存储技术实践

1. 在linux系统下配置minio server端，通过浏览器登陆127.0.0.1查看效果，进行简单的创建bucket，上传文件，删除文件和bucket操作，通过指令查看创建结果。
2. 在linux下配置minio客户端mc，在命令行下输入命令实现创建bucket，上传文件，删除文件和bucket操作，通过访问网址查看新建及删除的结果。
3. 在虚拟机linux的docker环境下配置s3proxy，浏览器登陆127.0.0.1查看效果。采用curl命令，实现创建bucket，上传文件，删除文件和bucket操作。
4. 在虚拟机linux下配置s3cmd，在命令行下输入命令实现创建bucket，上传文件，删除文件和bucket操作。

## 4.2 对象存储性能分析

1. 读写性能对比。
2. 在linux下运行minio数据库，提交负载样例到<http://10.0.2.15:9000，观察在thread=10>下运行和结束后的各种状态指标。
3. 在linux下配置s3proxy和s3cmd，提交负载样例到http://127.0.0.1:80，观察运行和结束后的各种状态指标。
4. 将测试案例的duration改为30s，进行测量。
5. 将测试样例的测试轮数改为2，进行测试。

# 五、实验过程

首先在Windows下熟悉minio，下载Windows版本的minio并安装后，在cmd中连接数据库。效果如图5.1。

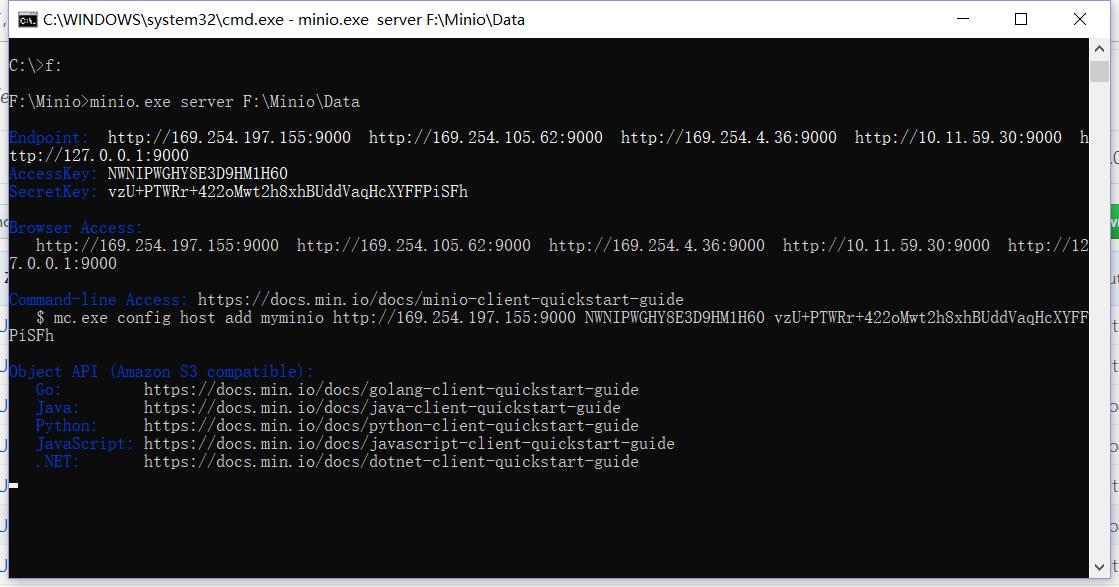


图5.1 Windows中连接minio

稍微熟悉对minio的操作后，进入linux Ubuntu 16.04，下载minio并连接。连接minio数据库语句如图5.2：

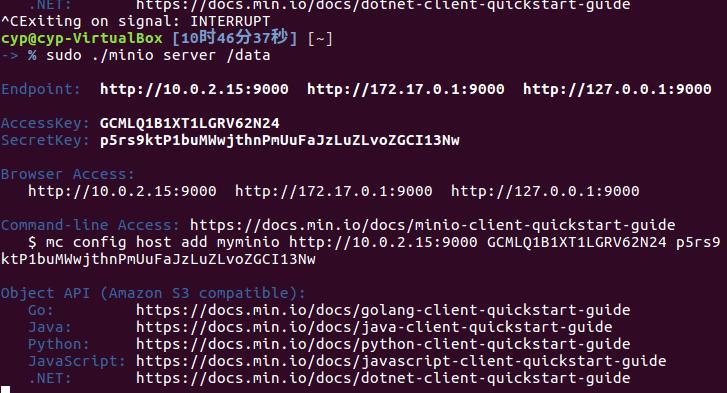


图5.2 连接minio

从图中可以看出，AccessKey是用户名，SecretKey是用户密码。用这两个登录http：//10.0.2.15:9000可以登录网页的minio，效果如图5.3：

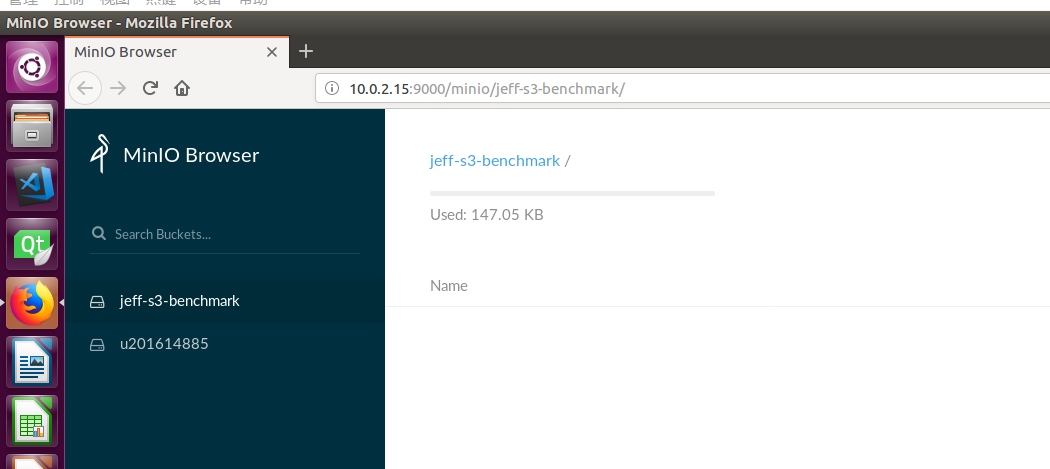


图5.3 网页中浏览minio

在minio中新建文件夹，如图中的最先的u201614885，文件夹开头字母不能是大写字母。

在linux中安装minio server和minio client语句和配置方法下述代码：

安装minio server：

wget https://dl.min.io/server/minio/release/linux-amd64/minio

chmod +x minio

./minio server /mnt/data

安装minio client：

wget https://dl.min.io/client/mc/release/linux-amd64/mc

chmod +x mc

./mc --help

由于老师推荐使用s3-benchmark，我就没有使用Cosbench，而是用s3-benchmark做接下来的实验。

首先从网上下载s3-benchmark，利用语句：

go get -u github.com/chinglinwen/s3-benchmark

进行下载。但是由于该版本存在缺少项，不能支持minio。故我又从其他地方找到了s3-benchmark.go的源代码，复制后要重新用go编译，才能运行，

在做到这一步时，我用我的电脑重新进行编译时，编译不通过，显示的错误是缺少关键文件。具体报错如图5.4：

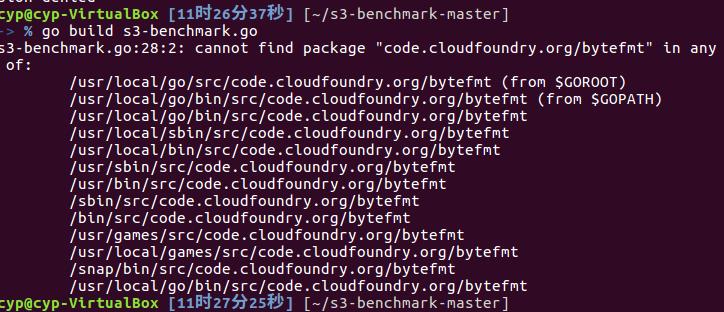


图5.4 编译失败 显示缺少文件

在一开始发现这个错误时一时无法解决。我打开要编译的文件，发现该文件确实在代码中使用到了，也就是说还不能忽略这个问题。于是我打开go的安装地址，在src文件中查找，发现文件确实缺少，但是正常安装下来为什么会缺少这个文件，我也不得而知，可能是之前安装时中断过一次，可能是更新go时尝试更改go的安装地址导致。现在的主要问题就是把丢失的文件重新放回去。

于是我上网查找相关文件，又对比同学的安装后的文件系统，最后选择复制了一份安装时就应该装在src文件下的code.cloudfoundry.org，将该文件夹复制到src文件夹下，再次编译就可以编译成功了，复制文件代码及编译过程如图5.5所示:

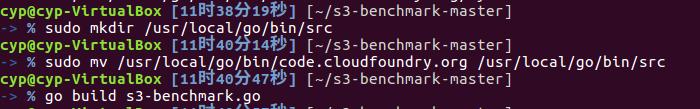


图5.5 复制文件代码及编译过程

成功运行的效果如图5.6所示：

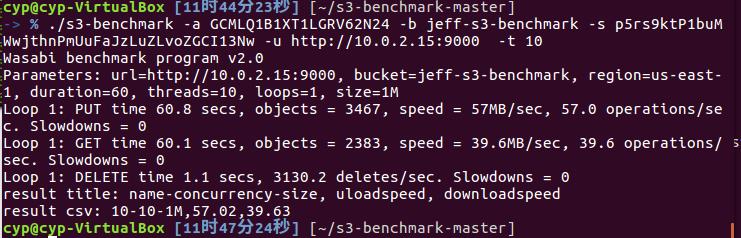


图5.6 最后运行结果

用minio测试s3-benchmark成功之后，用s3proxy进行测试。首先安装s3proxy和s3cmd，这两个分别是服务器和客户端。安装代码和效果如图5.7，5.8所示：

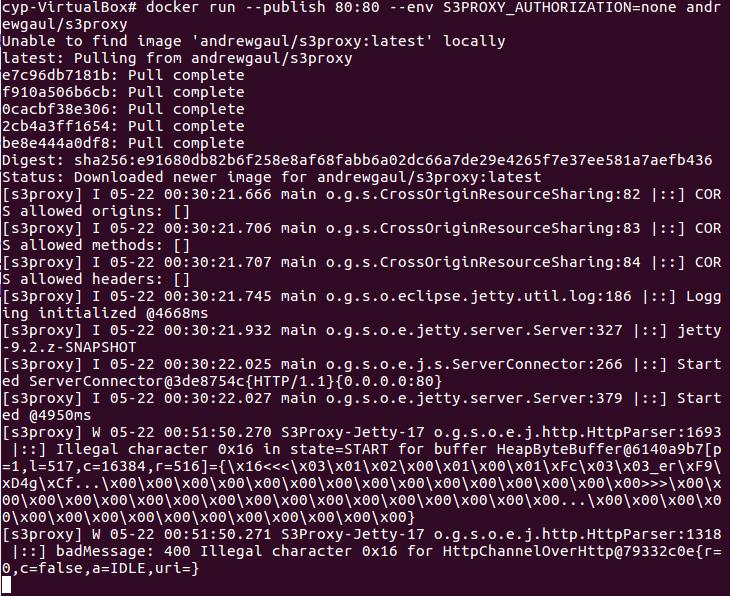


图5.7 安装s3proxy

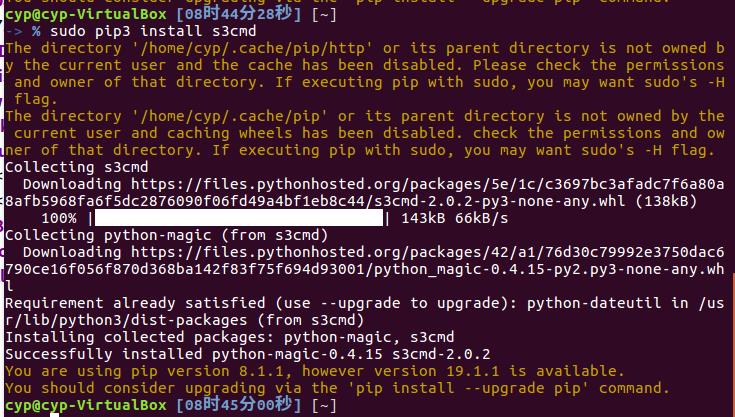


图5.8 安装s3cmd

安装时要对s3cmd configuration进行设置，效果如图5.9：

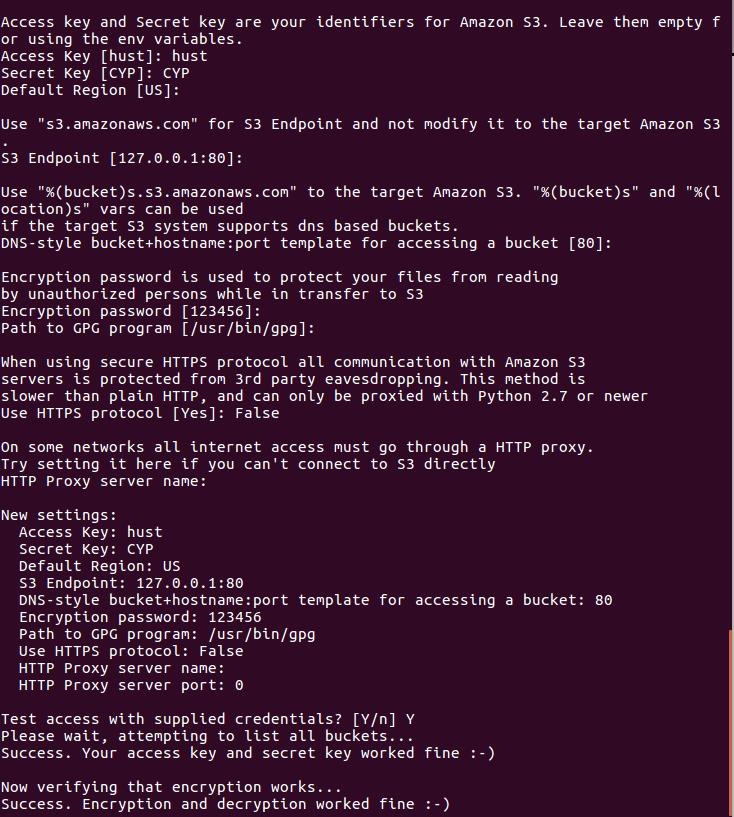


图5.9 成功设置s3cmd configuration

尝试使用代码在服务器中添加bucket，添加代码如图5.10：

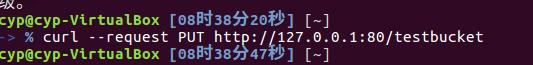


图5.10 用代码添加bucket

添加后，通过登录网址<http://127.0.0.1:80>，或者通过代码查看，可以检查是否创建成功，效果分别如图5.11,5.12：

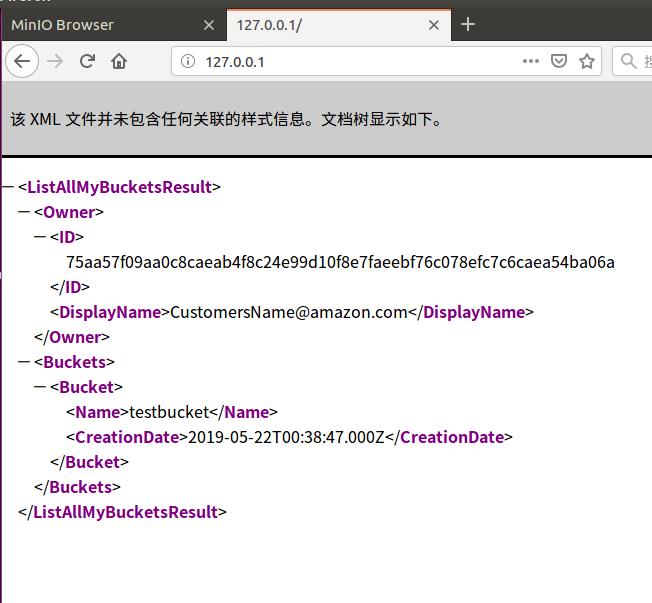


图5.11 登录网址查看创建情况

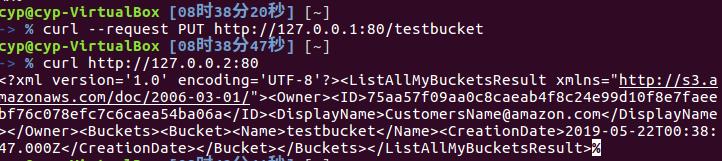


图5.12 用代码查看创建情况

从图5.11中可以看出有Bucket一栏，显示了创建时间等信息，从图5.12中的显示结果的第四行中间部分可以看到testbucketd的一项，这就是刚刚创建的bucket。

将s3-benchmark的测试代码中的传递网址改为s3proxy的网址，再次测试，效果如图5.13:

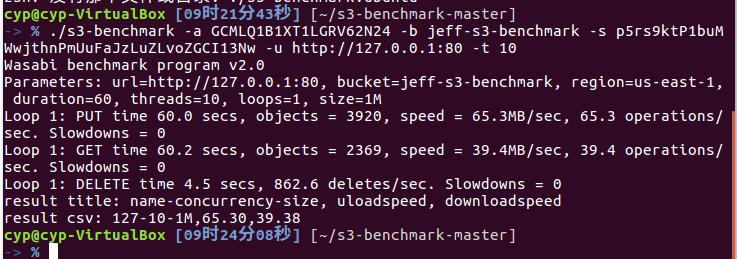


图5.13 用s3proxy测试程序

将go文件中的duration改为30，loops改为2，重新编译并测试，效果如图5.14：

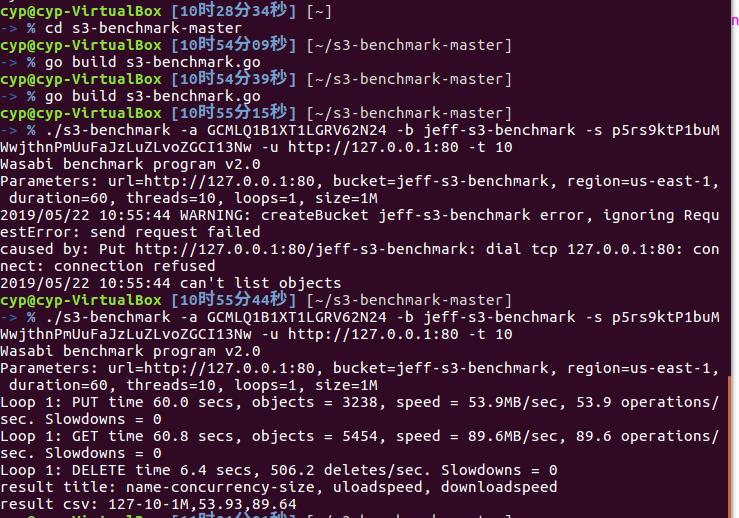


图5.14 1M文件 10线程

改为对500K的文件进行操作，即在指令最后加上-z=500k，测试效果如下：

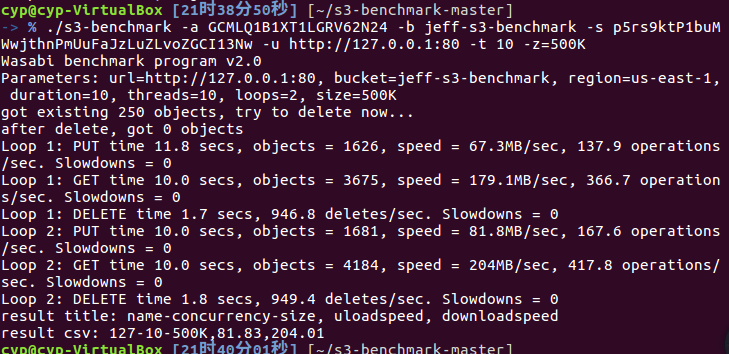


图5.15 500K文件 10线程

改为100k的文件大小：

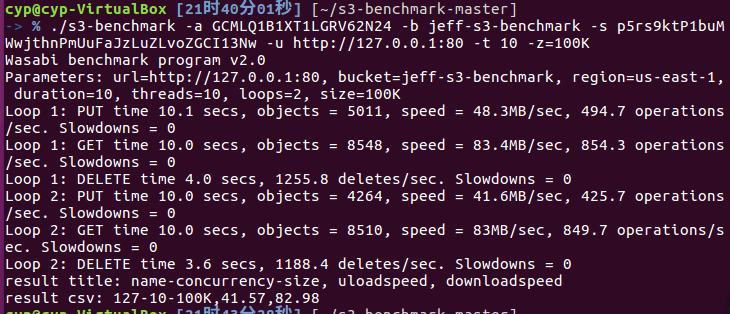


图5.16 100K文件 10线程

改为1k大小，效果如图：

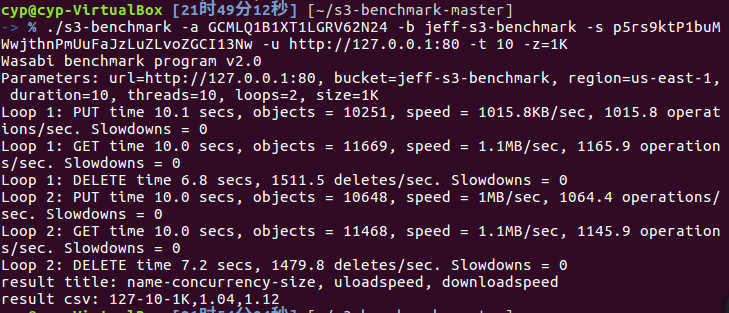


图5.15 1K文件 10线程

改为线程数为5，再次测试：

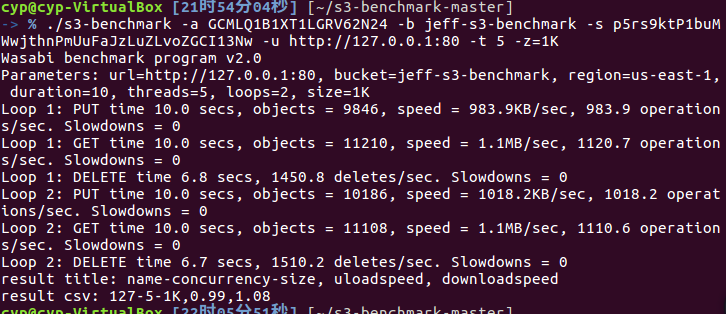


图5.16 1K文件 5线程

将测试结果汇总在表格中，效果如表5.1：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size | Threads | PUT | GET | Operation |
| 1K | 10 | 1015.8KB/sec | 1.1MB/sec | 1015.8，1165.9 operations/sec |
| 100K | 10 | 48.3MB/sec | 83.4MB/sec | 494.7，854.3 operations/sec |
| 500K | 10 | 67.3MB/sec | 179.1MB/sec | 137.9，366.7 operations/sec |
| 1M | 10 | 50MB/sec | 166.1MB/sec | 50.0，166.1 operations/sec |
| 1K | 5 | 983.9KB/sec | 1.1MB/sec | 983.9，1120.7 operations/sec |
| 100K | 5 | 43.8MB/sec | 79MB/sec | 448.8，982.6 operations/sec |
| 200K | 5 | 69MB/sec | 147.4MB/sec | 353.1，754.9 operations/sec |

表5.1

将表格中信息绘制成折线图，如下图5.17，5.18：

图5.17 threads=10 传输速率与传输文件大小的关系

图5.18 threads=10 吞吐率与传输文件大小的关系

# 六、实验总结

从表格中的实验数据可以看出，当进行操作的文件很小，只有1K大小时，上传和下载速率都较小，接近于正常值，约在每秒一兆左右。但是当进行操作的文件大小增大时，这二者的速率会先增大再减小，大于1M的文件我就没有再进行测试，因为此时电脑已经比较卡顿了。而且随着文件的大小的增大，operation也会减小。

这次实验分三次完成。第一次我一开始先是按照老师要求，尝试同git上传文件到GitHub上。说来惭愧，一直都没有使用git客户端通过命令上传过文件，于是整这个东西整了大半节课。在最后剩下一点时间，我在Windows下安装了minio，并且通过指令连接上了minio，在该网页中完成创建文件，修改密码等操作。第二节课上，老师说建议不要使用Cosbench来进行使用，我就选择了S3-benchmark来进行实验。在实验中，因为安装软件时中断过几次，又在中途更改安装位置，导致我试图用go语言编译文件时显示缺少文件。几经波折，终于把缺少的文件补上，也成功编译了，在最终，在第二节课的最后两分钟，把测试程序跑了出来。第三节课上主要更改实验参数来重新测试，进行对比。而且安装了s3proxy和s3cmd来进行测试。实验中通过更改线程数量，文件大小来进行测试。整的来说，存储实验虽然只有三次，但是实验结构紧密，让我感觉也挺有收获。

存储实验的整体难度不很大，实验中虽然也有挫折，也有的时候出现莫名其妙的错误，明明没有问题但是就是编译不过，但是还好最后也是找到问题并且解决了问题。安装软件时如果不注意有可能会出错，但是按照网上教学来，也可以很大程度的避免一些不容易发现的问题。

存储是计算机中很重要的，不可或缺的一部分。要学好与存储有关的相关知识，这样在之后工作，生活中也很有用处。最后也要感谢老师对我们的悉心教导，每节课上课前都嘱咐我们一些需要注意的事项。真的帮助我们很多，给我们在一头露水时点明迷津，再次感谢老师为我们提供的帮助。

# 参考文献

[1] ARNOLD J. OpenStack Swift[M]. O’Reilly Media, 2014.

[2] ZHENG Q, CHEN H, WANG Y等. COSBench: A Benchmark Tool for Cloud Object Storage Services[C]//2012 IEEE Fifth International Conference on Cloud Computing. 2012: 998–999.

[3] WEIL S A, BRANDT S A, MILLER E L等. Ceph: A Scalable, High-performance Distributed File System[C]//Proceedings of the 7th Symposium on Operating Systems Design and Implementation. Berkeley, CA, USA: USENIX Association, 2006: 307–320.