

2016 级

《物联网数据存储与管理》课程

实 验 报 告

姓 名 田雨欣

学 号 U201614909

班 号 物联网 1601 班

日 期 2019.05.28

目 录

一、实验目的	1
二、实验背景	1
三、实验环境	1
四、实验内容	2
4.1 对象存储技术实践	2
4.2 对象存储性能分析	2
五、实验过程	2
六、实验总结	14
6.1 实验结果分析	14
6.2 心得体会	18
参考文献	19

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

本次实验为对象存储入门实践，对象存储是用来描述解决和处理离散单元的方法的通用术语。对象在一个层结构中不会再有层级结构，是以扩展元数据为特征的。每个对象都被分配一个唯一的标识符，允许一个服务器或者最终用户来检索对象，而不必知道数据的物理地址。这种方法对于在云计算环境中自动化和简化数据存储有帮助。

实验需完成的部分有：搭建基础环境、准备对象存储服务器端、准备对象存储客户端、使用对象存储测评工具等。

MinIO 是一个基于 Apache License v2.0 开源协议的对象存储服务。它兼容亚马逊 S3 云存储服务接口，非常适合于存储大容量非结构化的数据，例如图片、视频、日志文件、备份数据和容器/虚拟机镜像等，而一个对象文件可以是任意大小，从几 kb 到最大 5T 不等。MinIO 是一个非常轻量的服务,可以很简单的和其他应用的结合，类似 NodeJS, Redis 或者 MySQL。

S3Proxy 实现了 S3 API 和代理请求，从而实现了几个用例：从 S3 到 Backblaze B2, EMC Atmos, Google Cloud, Microsoft Azure 和 OpenStack Swift 的翻译、使用本地文件系统在 Amazon 的情况下进行测试、通过中间件扩展、嵌入到 JAVA 应用中。

s3bench 可以针对 S3 兼容端点运行非常基本的吞吐量基准测试。它执行一系列 put 操作，然后执行一系列 get 操作并显示相应的统计信息。该工具使用 AWS Go SDK。

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。

三、实验环境

本次实验由于分别在 Windows 下和 linux 下完成，因此会有两个操作系统，同时所涉及到的 Java、Python 等在 Windows 下和在 Linux 下也不尽相同。

本次实验的实验环境如表 1-1 所示：

表 1-1 实验环境：

操作系统	Win10	Ubuntu 16.04
Java 版本	1.8.0-131	1.8.0-211
Python 版本	3.7.3	2.7.12
Go 版本	go1.12.5 windows/amd64	go1.12.5 linux/amd64
Git 版本	2.21.0.windows.1	
Docker 版本	18.09.2	
服务器端	Minio	S3proxy
客户端	Minio Client	S3cmd
评测工具	s3bench、s3benchmark	

四、实验内容

首先搭建 python 和 Java 的基础环境，并完成相应的 Go 语言安装，然后搭建服务器和客户端，最后启动 s3bench、s3benchmark 进行测试。

4.1 对象存储技术实践

Windows 部分：

- 1、在 Windows 环境下配置 Python、Java、Go、Git。
- 2、下载并安装 minio 服务器端，完成配置，开启服务器；
- 3、下载并安装 mc 客户端，创建 bucket 并上传文件；
- 4、安装使用 s3benchmark 和 s3bench 进行测试；

Linux 部分：

- 1、在 Linux 环境下配置 Python、Java、Go、Docker。
- 2、下载并安装 s3proxy，将其作为服务器，开启服务器；
- 3、下载并安装 s3cmd，完成配置，将其作为客户端，创建 bucket 并上传文件；
- 4、安装使用 s3benchmark 和 s3bench 进行测试；

4.2 对象存储性能分析

- 1、比较相同服务器端和客户端在不同线程、不同规模下的性能差别。
- 2、比较不同的服务器端和客户端的在相同条件下的性能差别。

五、实验过程

1、实验环境搭建：

在实验开始前，在 windows 环境下下载安装好 Python、Java、Go、Git，并完成相应的配置。安装配置完毕如图 1-1 所示。

```
Windows PowerShell
PS C:\Users\dell\Desktop> git version
git version 2.21.0.windows.1
PS C:\Users\dell\Desktop> go version
go version go1.12.5 windows/amd64
PS C:\Users\dell\Desktop> java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
PS C:\Users\dell\Desktop> python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 1-1 windows 下搭建实验环境

安装好 ubuntu 16.04 虚拟机，并在 linux 环境下下载安装好 Python、Java、Go，并完成相应的配置。安装配置完毕如图 1-2 所示。

```
maggie@maggie-virtualbox: ~
maggie@maggie-virtualbox:~$ go version
go version go1.12.5 linux/amd64
maggie@maggie-virtualbox:~$ java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
maggie@maggie-virtualbox:~$ docker --version
Docker version 18.09.2, build 6247962
maggie@maggie-virtualbox:~$ python
Python 2.7.12 (default, Nov 12 2018, 14:36:49)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 1-2 Linux 下搭建实验环境

2、安装配置 Minio 服务器、mc 客户端并完成测试。

进入 minio 官网，下载 minio.exe 文件，打开命令行，使用 cd 命令进入安装目录，键入：

```
minio.exe server E:\minio\minio_server
```

打开服务器，获取相应的 endpoint、accessKey 和 secretKey。（如图 1-3）

```
命令提示符 - minio.exe server E:\minio\minio_server
C:\Users\dell>E:
E:\>cd minio
E:\minio>minio.exe server E:\minio\minio_server

Endpoint: http://169.254.99.227:9000 http://169.254.57.211:9000 http://169.254.92.181:9000 http://169.254.6.8:9000 http://169.254.152.7:9000 http://192.168.56.1:9000 http://127.0.0.1:9000
AccessKey: S18T2C7IPMESM6NDG5BV
SecretKey: h90rcONK30MfEoyokHNwBVLLCRgFNWhKvHrCC3sh

Browser Access:
http://169.254.99.227:9000 http://169.254.57.211:9000 http://169.254.92.181:9000 http://10.1.1.1:9000 http://169.254.6.8:9000 http://169.254.152.7:9000 http://192.168.56.1:9000 http://127.0.0.1:9000

Command-line Access: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc.exe config host add myminio http://169.254.99.227:9000 S18T2C7IPMESM6NDG5BV h90rcONK30MfEoyokHNwBVLLCRgFNWhKvHrCC3sh

Object API (Amazon S3 compatible):
Go: https://docs.min.io/docs/golang-client-quickstart-guide
```

图 1-3 打开 minio 服务器

打开浏览器，在地址栏中键入：

<http://127.0.0.1:9000>

进入 MinIO Browser 界面。（如图 1-4）

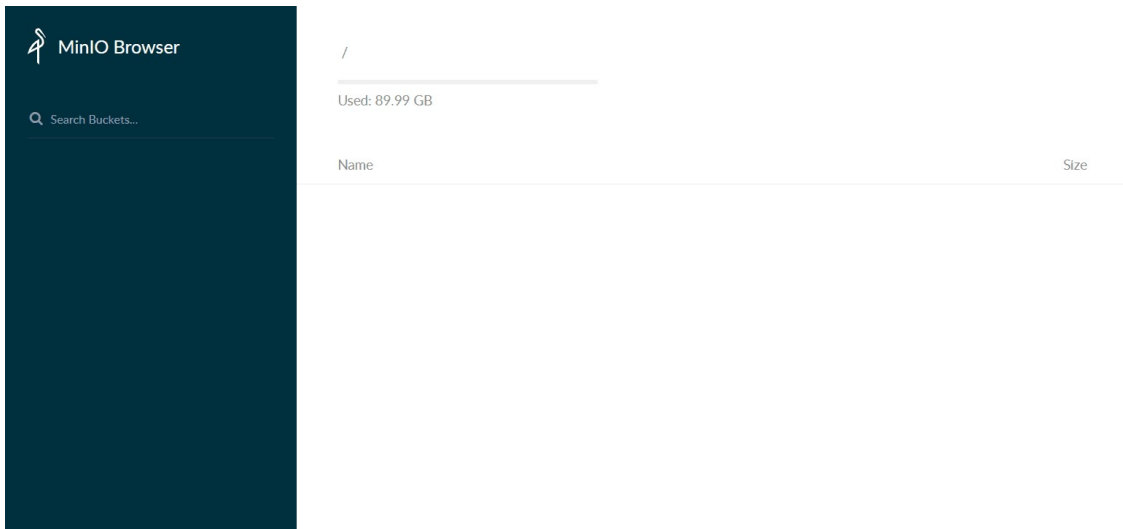


图 1-4 进入服务器页面

在该用户页面下修改 **accessKey** 和 **secretKey**，便于后续登陆工作。（如图 1-5）

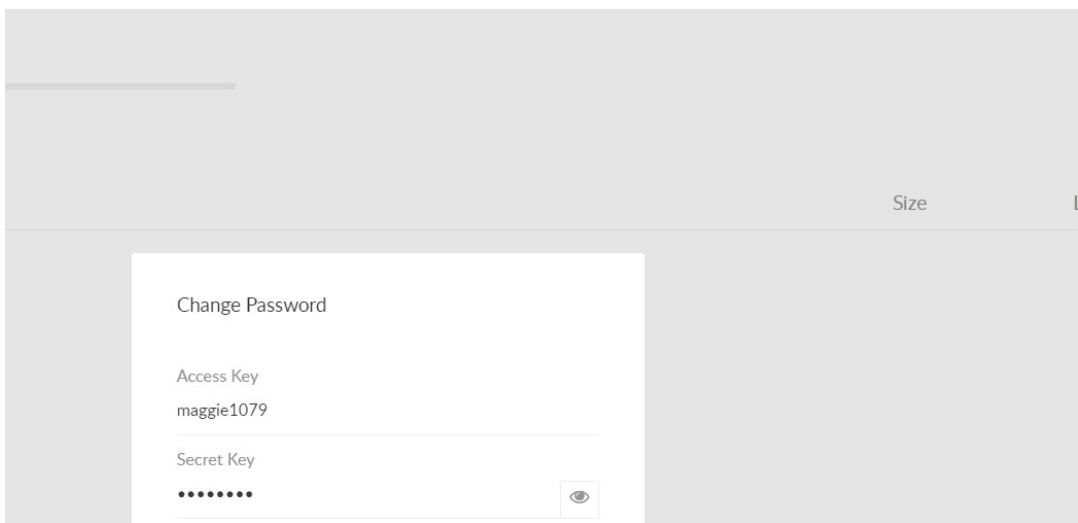


图 1-5 修改 **accessKey** 和 **secretKey**

修改完密码后，打开命令行，使用 **cd** 命令进入安装目录，再次键入：

minio.exe server E:\minio\minio_server

打开服务器，获取修改完用户名密码之后的 **endpoint**、**accessKey** 和 **secretKey**。
（如图 1-6）

```
C:\ 命令提示符 - minio.exe server E:\minio\minio_server
Microsoft Windows [版本 10.0.17134.706]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\dell>E:

E:\>cd minio

E:\minio>minio.exe server E:\minio\minio_server

Endpoint: http://169.254.99.227:9000 http://169.254.57.211:9000 http://169.254.92.181:9000 http://169.254.6.8:9000 http://169.254.152.7:9000 http://192.168.56.1:9000 http://127.0.0.1:9000

AccessKey: maggie1079
SecretKey: 19980206

Browser Access:
http://169.254.99.227:9000 http://169.254.57.211:9000 http://169.254.92.181:9000 http://10.1.1.1:9000 http://169.254.6.8:9000 http://169.254.152.7:9000 http://192.168.56.1:9000 http://127.0.0.1:9000

Command-line Access: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc.exe config host add myminio http://169.254.99.227:9000 maggie1079 19980206

Object API (Amazon S3 compatible):
$ mc.exe config host add myminio https://docs.min.io/docs/golang-client-quickstart-guide
```

图 1-6 修改 accessKey 和 secretKey 成功

下载 mc 客户端，打开 powershell，键入：

```
E:\minio\mc.exe config host add Maggie http://127.0.0.1:9000 maggie1079 19980206 S3v4
```

添加客户端主机。其中 maggie 是主机名，后三个参数分别是 endpoint、accessKey 和 secretKey，最后则是选择的 API 签名，默认设置为“S3v4”。（如图 1-7）

（powershell 本质与 cmd 差不多，但是会有颜色区分，更加美观）

```
PS E:\minio> E:\minio\mc.exe config host add maggie http://127.0.0.1:9000 maggie1079 19980206 S3v4
Added maggie successfully.
PS E:\minio>
```

图 1-7 添加客户端主机

服务器中所有的文件是放在不同的 bucket 中的，客户端可以新建自己的 bucket。键入：

```
E:\minio\mc.exe mb Maggie/maggiebucket
```

创建 bucket（如图 1-8）。

```
Windows PowerShell
PS E:\minio> E:\minio\mc.exe config host add maggie http://127.0.0.1:9000 maggie1079 19980206 S3v4
Added maggie successfully.
PS E:\minio> E:\minio\mc.exe mb maggie/maggiebucket
Bucket created successfully maggie/maggiebucket .
PS E:\minio>
```

图 1-8 创建 bucket

mb 即为 makebucket 的简写，服务器主机名为 maggie，bucket 名为 maggiebucket。创建成功后即可在 Minio Browser 中看到新的 bucket，（如图 1-9）

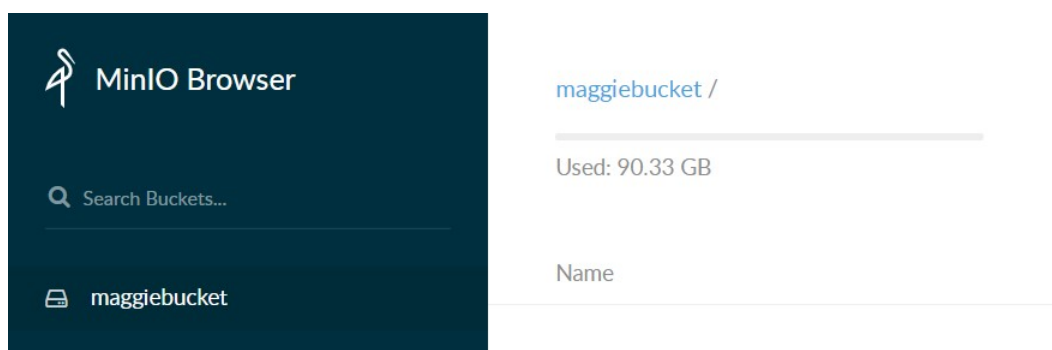


图 1-9 创建 bucket 成功

接下来上传文件。依旧在 minio 文件夹下，键入：

```
E:\minio\mc.exe cp C:\Users\dell\Desktop\mytest.txt maggie/maggiebucket
```

格式为：mc.exe cp 要上传的文件名 目的服务器/目的 bucket（如图 1-10）

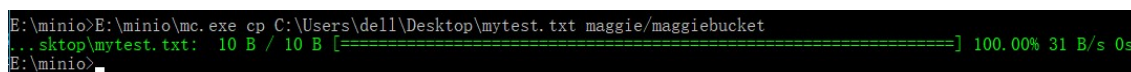


图 1-10 上传文件

在 MinIO Browser 页面中可以看到刚才上传的文件。（如图 1-11）

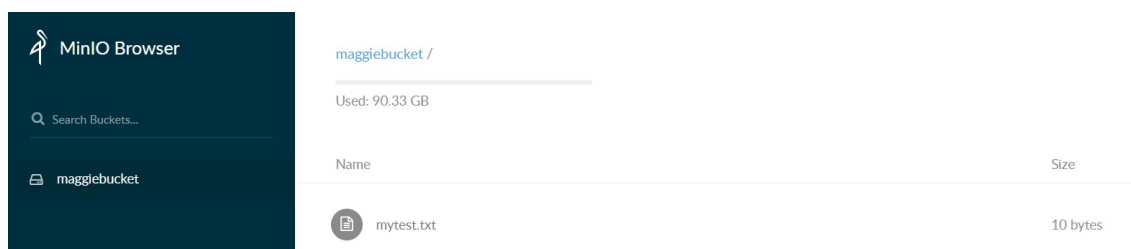


图 1-11 上传文件成功

然后安装测试工具。首先安装 s3benchmark，键入：

```
set http_proxy=http://127.0.0.1:1080
go get -v -u -insecure github.com/chinglinwen/s3-benchmark
```

如图 1-12 所示。

其中第一行是为了设置代理，便于安装 go，如果不进行代理的设置和后续-v、-u 参数的设置，将会受到防火墙的限制，使得 go 命令难以正常执行。



图 1-12 安装 s3benchmark

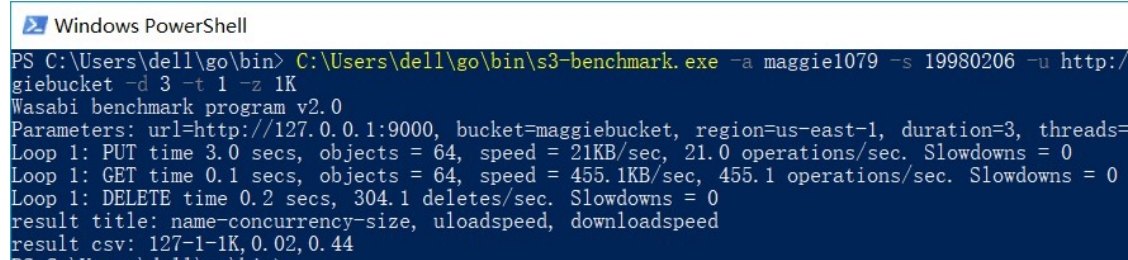
进入可执行文件安装目录 C:\Users\dell\go\bin，键入：


```
C:\Users\dell\go\bin\s3-benchmark.exe -a maggie1079 -s 19980206 -u http://127.0.0.1:9000 -b maggiebucket -d 3 -t 1 -z 1K
```

各参数含义如下：

```
-a string
    Access key
-b string
    Bucket for testing (default "wasabi-benchmark-bucket")
-d int
    Duration of each test in seconds (default 60)
-l int
    Number of times to repeat test (default 1)
-s string
    Secret key
-t int
    Number of threads to run (default 1)
-u string
    URL for host with method prefix (default "http://s3.wasabisys.com")
-z string
    Size of objects in bytes with postfix K, M, and G (default "1M")
```

对刚才的 bucket 进行 put、get、delete 测试。（如图 1-13）



```
Windows PowerShell
PS C:\Users\dell\go\bin> C:\Users\dell\go\bin\s3-benchmark.exe -a maggie1079 -s 19980206 -u http://127.0.0.1:9000 -b maggiebucket -d 3 -t 1 -z 1K
Wasabi benchmark program v2.0
Parameters: url=http://127.0.0.1:9000, bucket=maggiebucket, region=us-east-1, duration=3, threads=1
Loop 1: PUT time 3.0 secs, objects = 64, speed = 21KB/sec, 21.0 operations/sec. Slowdowns = 0
Loop 1: GET time 0.1 secs, objects = 64, speed = 455.1KB/sec, 455.1 operations/sec. Slowdowns = 0
Loop 1: DELETE time 0.2 secs, 304.1 deletes/sec. Slowdowns = 0
result title: name-concurrency-size, uploadspeed, downloadspeed
result csv: 127-1-1K,0.02,0.44
PS C:\Users\dell\go\bin>
```


图 1-13 s3benchmark 测试

接着安装 s3bench，键入：

```
set http_proxy=http://127.0.0.1:1080
go get -v -u --insecure github.com/igneous-systems/s3bench
```

如图 1-14 所示。

第一行与安装 s3benchmark 同理，是为了设置代理，便于安装 go。



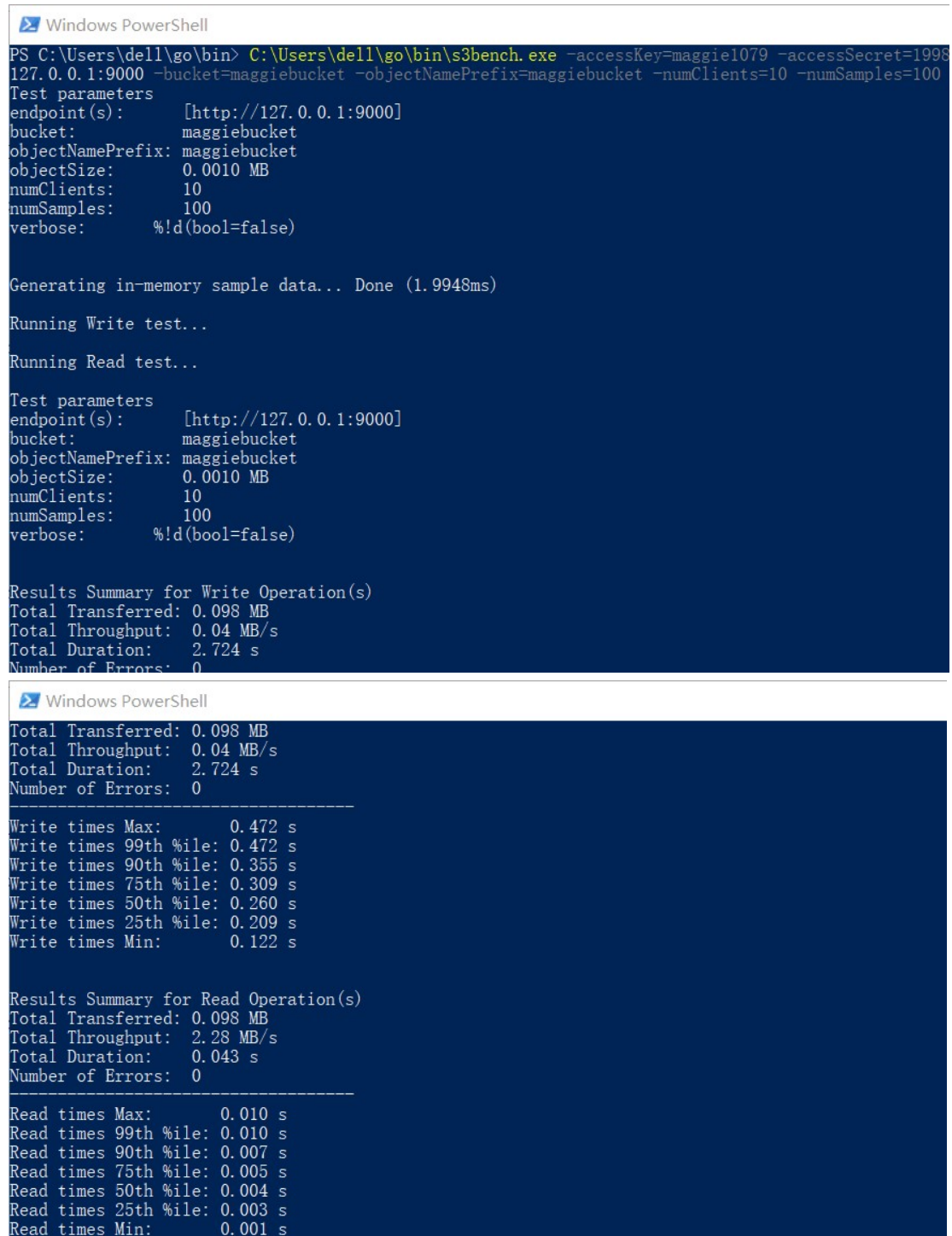
```
Windows PowerShell
PS C:\Users\dell\go\bin> set http_proxy=http://127.0.0.1:1080
PS C:\Users\dell\go\bin> go get -v -u --insecure github.com/igneous-systems/s3bench
github.com/igneous-systems/s3bench (download)
github.com/aws/aws-sdk-go (download)
PS C:\Users\dell\go\bin>
```

图 1-14 安装 s3bench

进入可执行文件安装目录 C:\Users\dell\go\bin，键入：

```
C:\Users\dell\go\bin\s3bench.exe -accessKey=maggie1079
-accessSecret=19980206 -endpoint=http://127.0.0.1:9000 -bucket=maggiebucket
-objectNamePrefix=maggiebucket -numClients=10 -numSamples=100
-objectSize=1024
```

对刚才的 bucket 进行读写测试：（如图组 1-16）



```
Windows PowerShell
PS C:\Users\dell\go\bin> C:\Users\dell\go\bin\s3bench.exe -accessKey=maggie1079 -accessSecret=19980206 -endpoint=http://127.0.0.1:9000 -bucket=maggiebucket -objectNamePrefix=maggiebucket -numClients=10 -numSamples=100
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           maggiebucket
objectNamePrefix: maggiebucket
objectSize:       0.0010 MB
numClients:       10
numSamples:       100
verbose:          %!d(bool=false)

Generating in-memory sample data... Done (1.9948ms)
Running Write test...
Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           maggiebucket
objectNamePrefix: maggiebucket
objectSize:       0.0010 MB
numClients:       10
numSamples:       100
verbose:          %!d(bool=false)

Results Summary for Write Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  0.04 MB/s
Total Duration:    2.724 s
Number of Errors:  0

Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  2.28 MB/s
Total Duration:    0.043 s
Number of Errors:  0

Write times Max:      0.472 s
Write times 99th %ile: 0.472 s
Write times 90th %ile: 0.355 s
Write times 75th %ile: 0.309 s
Write times 50th %ile: 0.260 s
Write times 25th %ile: 0.209 s
Write times Min:      0.122 s

Read times Max:       0.010 s
Read times 99th %ile: 0.010 s
Read times 90th %ile: 0.007 s
Read times 75th %ile: 0.005 s
Read times 50th %ile: 0.004 s
Read times 25th %ile: 0.003 s
Read times Min:       0.001 s
```

图组 1-16 s3bench 测试

3、安装配置 s3proxy 服务器和 s3cmd 客户端并完成测试。

首先在 linux 系统下用 docker 运行 s3proxy，键入：

```
Sudo docker run --publish 80:80 --env S3PROXY_AUTHORIZATION=none  
andrewgaul/s3proxy
```

成功运行 s3proxy 服务器（如图 1-17）

```
maggie@maggie-virtualbox:~/s3proxy-1.6.1$ sudo docker run --publish 80:80 --env  
S3PROXY_AUTHORIZATION=none andrewgaul/s3proxy  
[s3proxy] I 05-22 02:58:15.821 main o.g.s.CrossOriginResourceSharing:82 [::] COR  
S allowed origins: []  
[s3proxy] I 05-22 02:58:15.826 main o.g.s.CrossOriginResourceSharing:83 [::] COR  
S allowed methods: []  
[s3proxy] I 05-22 02:58:15.827 main o.g.s.CrossOriginResourceSharing:84 [::] COR  
S allowed headers: []  
[s3proxy] I 05-22 02:58:15.844 main o.g.s.o.eclipse.jetty.util.log:186 [::] Logg  
ing initialized @2182ms  
[s3proxy] I 05-22 02:58:15.952 main o.g.s.o.e.jetty.server.Server:327 [::] jetty  
-9.2.z-SNAPSHOT  
[s3proxy] I 05-22 02:58:16.038 main o.g.s.o.e.j.s.ServerConnector:266 [::] Start  
ed ServerConnector@681622f6[HTTP/1.1]{0.0.0.0:80}  
[s3proxy] I 05-22 02:58:16.039 main o.g.s.o.e.jetty.server.Server:379 [::] Start  
ed @2376ms
```

图 1-17 运行 s3proxy

使用 pip 下载 s3cmd，键入：

```
sudo pip install s3cmd
```

成功下载 s3cmd（如图 1-18）

```
maggie@maggie-virtualbox: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
d by the current user and the cache has been disabled. Please check the permissi  
ons and owner of that directory. If executing pip with sudo, you may want sudo's  
-H flag.  
The directory '/home/maggie/.cache/pip' or its parent directory is not owned by  
the current user and caching wheels has been disabled. check the permissions and  
owner of that directory. If executing pip with sudo, you may want sudo's -H fla  
g.  
Collecting s3cmd  
Collecting python-dateutil (from s3cmd)  
  Downloading https://files.pythonhosted.org/packages/41/17/c62facbfbdb163c7f57f  
3844689e3a78bae1f403648a6afb1d0866d87fbb/python_dateutil-2.8.0-py2.py3-none-any.  
whl (226kB)  
    100% |████████████████████████████████████████| 235kB 244kB/s  
Collecting python-magic (from s3cmd)  
  Downloading https://files.pythonhosted.org/packages/42/a1/76d30c79992e3750dac6  
790ce16f056f870d368ba142f83f75f694d93001/python_magic-0.4.15-py2.py3-none-any.wh  
l  
Requirement already satisfied: six>=1.5 in /usr/local/lib/python2.7/dist-package  
s (from python-dateutil->s3cmd) (1.12.0)  
Installing collected packages: python-dateutil, python-magic, s3cmd  
Successfully installed python-dateutil-2.8.0 python-magic-0.4.15 s3cmd-2.0.2  
You are using pip version 19.0.3, however version 19.1.1 is available.  
You should consider upgrading via the 'pip install --upgrade pip' command.  
maggie@maggie-virtualbox:~$
```

图 1-18 下载 s3cmd

接下来需要对下载好的 s3cmd 进行配置，键入：

```
s3cmd --configure
```

完成配置如图 1-19 所示。


```
maggie@maggie-virtualbox: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

slower than plain HTTP, and can only be proxied with Python 2.7 or newer
Use HTTPS protocol [Yes]: no

On some networks all internet access must go through a HTTP proxy.
Try setting it here if you can't connect to S3 directly
HTTP Proxy server name:

New settings:
Access Key: maggie1079
Secret Key: 19980206
Default Region: US
S3 Endpoint: s3.amazonaws.com
DNS-style bucket+hostname:port template for accessing a bucket: %(bucket)s.s3.
amazonaws.com
Encryption password: 19980206
Path to GPG program: /usr/bin/gpg
Use HTTPS protocol: False
HTTP Proxy server name:
HTTP Proxy server port: 0

Test access with supplied credentials? [Y/n] n

Save settings? [y/N] y
Configuration saved to '/home/maggie/.s3cfg'
```

图 1-19 完成 s3cmd 配置

接着进入打开/home/maggie/.s3cfg 文件，修改部分信息如图 1-20 所示。

```
.s3cfg (~/) - gedit
打开(O) 保存(S)

delete_after_fetch = False
delete_removed = False
dry_run = False
enable_multipart = True
encrypt = False
expiry_date =
expiry_days =
expiry_prefix =
follow_symlinks = False
force = False
get_continue = False
gpg_command = /usr/bin/gpg
gpg_decrypt = %(gpg_command)s -d --verbose --no-use-agent --
batch --yes --passphrase-fd %(passphrase_fd)s -o %
(output_file)s %(input_file)s
gpg_encrypt = %(gpg_command)s -c --verbose --no-use-agent --
batch --yes --passphrase-fd %(passphrase_fd)s -o %
(output_file)s %(input_file)s
gpg_passphrase = 19980206
guess_mime_type = True
host_base = 127.0.0.1:80
host_bucket = 80
human_readable_sizes = False
invalidate_default_index_on_cf = False
invalidate_default_index_root_on_cf = True
invalidate_on_cf = False
kms_key =
limit = -1
limitrate = 0
list_md5 = False
log_target_prefix =
long_listing = False

纯文本 制表符宽度: 8 行 40, 列 11 插入
```

图 1-20 修改.s3cfg 文件

然后进行上传文件测试。键入：

```
s3cmd mb s3://maggiebucket
```

创建新的 bucket（如图 1-21）

```
Test access with supplied credentials? [Y/n] n
```

图 1-21 创建新的 bucket

接着键入：

```
s3cmd put /home/maggie/readme.txt s3://maggiebucket
```

将测试文件 readme.txt 文件上传至 maggiebucket（如图 1-22）

```
maggie@maggie-virtualbox:~$ s3cmd put /home/maggie/readme.txt s3://maggiebucket
upload: '/home/maggie/readme.txt' -> 's3://maggiebucket/readme.txt' [1 of 1]
16 of 16 100% in 0s 256.66 B/s done
maggie@maggie-virtualbox:~$
```

图 1-22 上传文件

打开浏览器，键入：

```
127.0.0.1
```

也能看到新添加的 bucket，如图 1-23 所示。

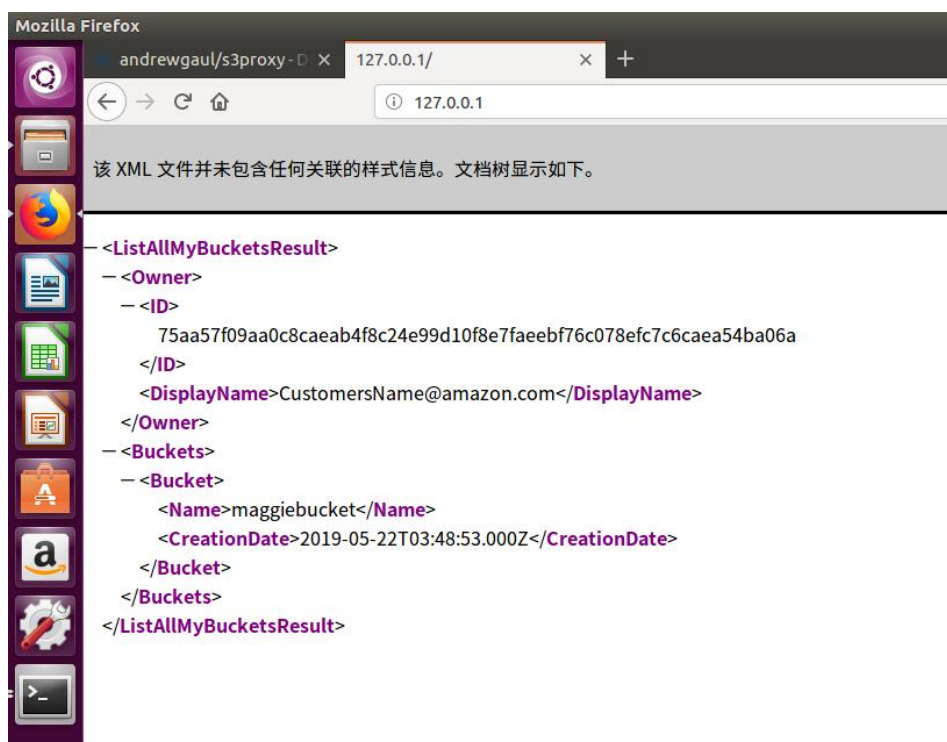


图 1-23 查看添加的 bucket

接着，键入：

```
127.0.0.1/maggiebucket
```

能看到新添加的文件 readme.txt（如图 1-24）。

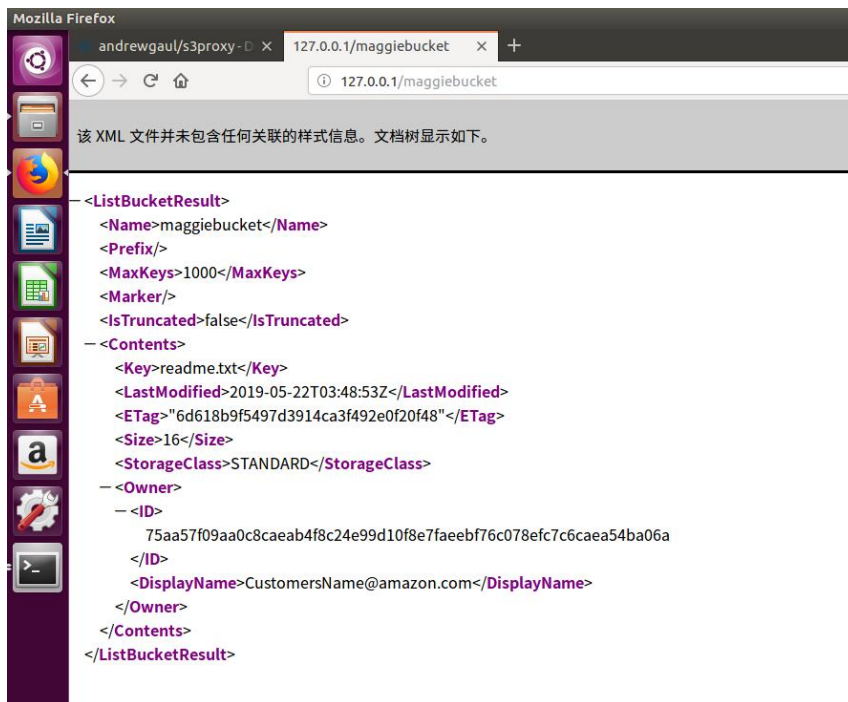


图 1-24 查看添加的文件

至此，说明客户端与服务器端连接成功。

接下来是安装并测试 s3benchmark。键入：

```
set http_proxy=http://127.0.0.1:1080
go get -v -u -insecure github.com/chinglinwen/s3-benchmark
```

如图 1-25 所示。

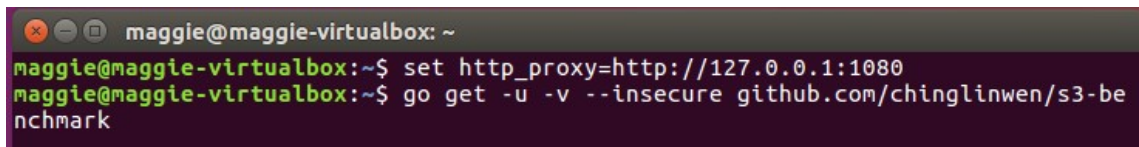


图 1-25 安装 s3benchmark

接下来用 s3benchmark 进行测试，进入~/go/bin 目录，键入：

```
./s3-benchmark -a maggiel079 -s 19980206 -u http://127.0.0.1:80 -b maggielbucket -d 3 -t 1 -z 1K
```

可得测试结果如图 1-26 所示。

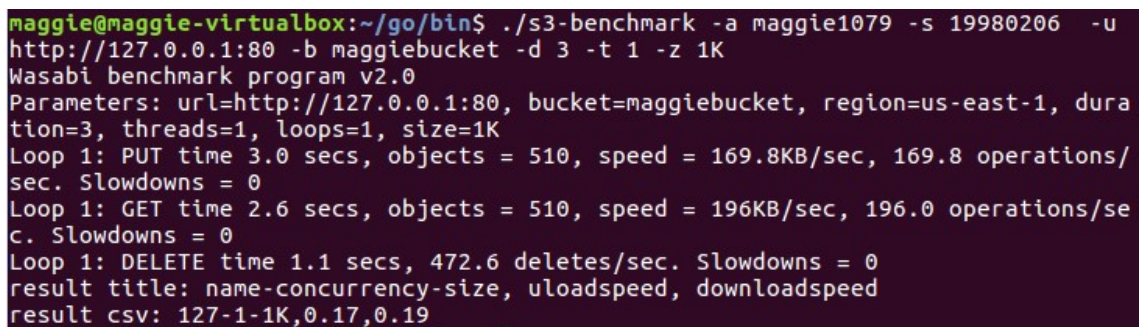



图 1-26 s3-benchmark 测试结果

接下来是安装并测试 s3bench。键入：

```
set http_proxy=http://127.0.0.1:1080  
go get -v -u -insecure github.com/igneous-systems/s3bench
```

如图 1-27 所示。



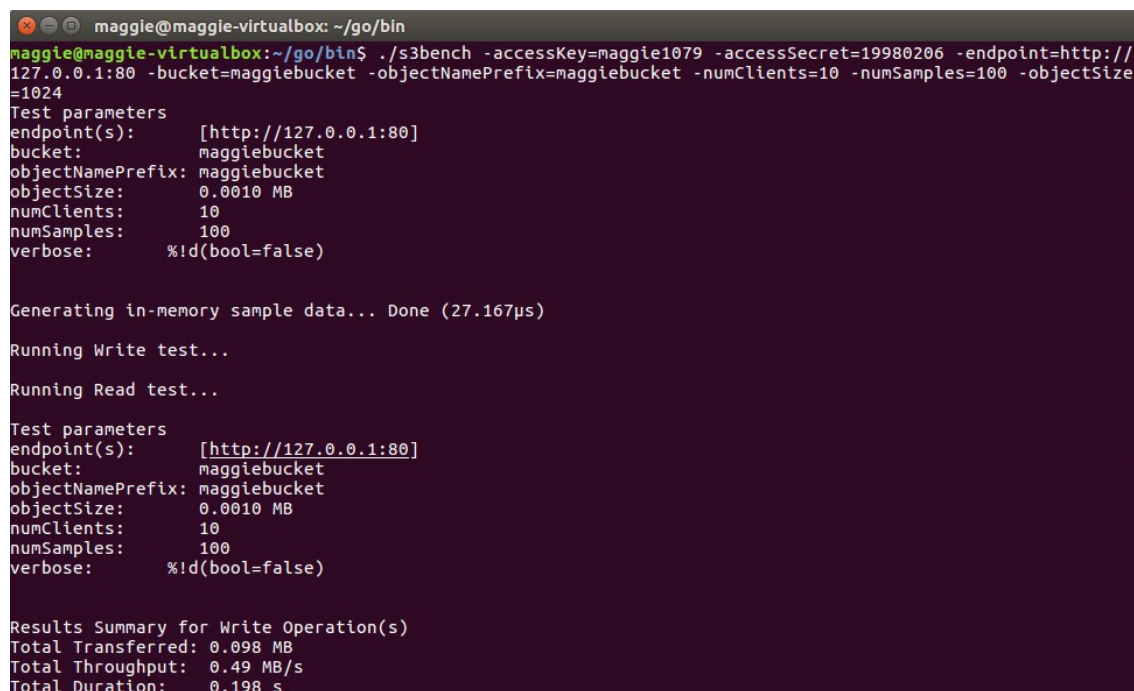
```
maggie@maggie-virtualbox:~$ set http_proxy=http://127.0.0.1:1080  
maggie@maggie-virtualbox:~$ go get -u -v --insecure github.com/igneous-systems/s3bench  
github.com/igneous-systems/s3bench (download)  
github.com/aws/aws-sdk-go (download)  
github.com/igneous-systems/s3bench
```

图 1-27 安装 s3bench

安装完毕后，用 s3bench 进行测试。键入：

```
./s3bench -accessKey=maggie1079 -accessSecret=19980206  
-endpoint=http://127.0.0.1:9000 -bucket=maggiebucket  
-objectNamePrefix=maggiebucket -numClients=10 -numSamples=100  
-objectSize=1024
```

可得测试结果如图组 1-28 所示。



```
maggie@maggie-virtualbox: ~/go/bin  
maggie@maggie-virtualbox:~/go/bin$ ./s3bench -accessKey=maggie1079 -accessSecret=19980206 -endpoint=http://127.0.0.1:80 -bucket=maggiebucket -objectNamePrefix=maggiebucket -numClients=10 -numSamples=100 -objectSize=1024  
Test parameters  
endpoint(s): [http://127.0.0.1:80]  
bucket: maggiebucket  
objectNamePrefix: maggiebucket  
objectSize: 0.0010 MB  
numClients: 10  
numSamples: 100  
verbose: %!d(bool=false)  
  
Generating in-memory sample data... Done (27.167µs)  
  
Running Write test...  
  
Running Read test...  
  
Test parameters  
endpoint(s): [http://127.0.0.1:80]  
bucket: maggiebucket  
objectNamePrefix: maggiebucket  
objectSize: 0.0010 MB  
numClients: 10  
numSamples: 100  
verbose: %!d(bool=false)  
  
Results Summary for Write Operation(s)  
Total Transferred: 0.098 MB  
Total Throughput: 0.49 MB/s  
Total Duration: 0.198 s
```



```

maggie@maggie-virtualbox: ~/go/bin
Total Transferred: 0.098 MB
Total Throughput: 0.49 MB/s
Total Duration: 0.198 s
Number of Errors: 0
-----
Write times Max: 0.063 s
Write times 99th %ile: 0.063 s
Write times 90th %ile: 0.040 s
Write times 75th %ile: 0.026 s
Write times 50th %ile: 0.014 s
Write times 25th %ile: 0.011 s
Write times Min: 0.002 s

Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput: 0.46 MB/s
Total Duration: 0.211 s
Number of Errors: 0
-----
Read times Max: 0.079 s
Read times 99th %ile: 0.079 s
Read times 90th %ile: 0.046 s
Read times 75th %ile: 0.031 s
Read times 50th %ile: 0.015 s
Read times 25th %ile: 0.006 s
Read times Min: 0.002 s

Cleaning up 100 objects...
Deleting a batch of 100 objects in range {0, 99}... Succeeded
Successfully deleted 100/100 objects in 1.035152782s
maggie@maggie-virtualbox:~/go/bin$

```

图组 1-28 s3bench 测试结果

至此，基础实验和进阶实验已全部完成。

六、实验总结

6.1 实验结果分析

1、横向对比分析：

- 在键入的参数一定时，将测试的 minio 的 s3benchmark 数据和测试的 s3proxy 的 s3benchmark 数据进行对比得表 1-1。

表 1-1 Minio 和 s3proxy 性能测试（s3benchmark）

Minio					
	time	objects	speed		Slowdowns
PUT	3.0 secs	64	21KB/sec	21.0 operations/sec	0
GET	0.1 secs	64	455.1KB/sec	455.1 operations/sec	0
DELETE	0.2 secs		304.1 deletes/sec		0
result csv	127-1-1K,0.02,0.44				

s3proxy					
	time	objects	speed		Slowdowns
PUT	3.0 secs	510	169.8KB/sec	169.8 operations/sec	0
GET	2.6 secs	510	196KB/sec	196.0 operations/sec	0
DELETE	1.1 secs		472.6 deletes/sec		0
result csv	127-1-1K,0.17,0.19				

分析此表可知：就 objects 而言，s3proxy 比 Minio 更多一些，速度方面的话，PUT 的速度 s3proxy 是远大于 Minio 的，它的 Delete 速度也很快，而 Minio 的 GET 速度比 s3proxy 快很多。s3proxy 的 DELETE 速度明明比 Minio 快，但是它要用更长的时间，是因为它要处理的 objects 数更多。因

此,Minio 在 GET 方面速度表现很好但是在 PUT 方面表现不好,而 s3proxy 就更加平均一些,表现的非常均衡。

- 在键入的参数一定时,将测试的 minio 的 s3bench 数据和测试的 s3proxy 的 s3bench 数据进行对比得表 1-2。

表 1-2 Minio 和 s3proxy 性能测试 (s3bench)

Minio			
	Transferred	Throughput	Duration
Write	0.098MB	0.04MB/s	2.724s
Read	0.098MB	2.28MB/s	0.043s
Delete	117.6891ms		

s3proxy			
	Transferred	Throughput	Duration
Write	0.098MB	0.49MB/s	0.198s
Read	0.098MB	0.46MB/s	0.211s
Delete	1.035s		

分析此表可知: Minio 的读速度非常快但是写速度非常慢,而 s3proxy 的速度比较均衡。该结论与 s3benchmark 的分析结论类似,因此不再赘述。注意: 横向对比分析的数据并非完全的控制变量法,仅控制输入参数一定,但仍会存在一定误差,因为二者可能会受到操作系统的影响,毕竟 minio 是在 windows 下而 s3proxy 是在 linux 下。真正的控制变量对比分析还需要更深入的测试和比对。但是数据细微的差距不影响整体的结论分析,因此该结论仍然具有相当的可信度。

2、纵向对比分析:

- 对 Minio 的纵向分析 (以 s3benchmark 为例): 固定其他参数不变,将时间延长为 10s,改变 size 和 thread,结果如表 1-3 所示。

表 1-3 Minio 在不同参数下性能比较 (s3benchmark)

Minio		10/1/1K			
	time	objects	speed		Slowdowns
PUT	10.0 secs	181	18KB/sec	18 operations/sec	0
GET	0.3 secs	181	545.4KB/sec	545.4 operations/sec	0
DELETE	0.4 secs		451.3 deletes/sec		0
result csv	127-1-1K,0.02,0.53				
Minio		10/5/1K			
	time	objects	speed		Slowdowns
PUT	10.0 secs	304	30.1KB/sec	30.1 operations/sec	0
GET	1.1 secs	1520	1.3MB/sec	1381.9 operations/sec	0
DELETE	0.3 secs		1212.0 deletes/sec		0
result csv	127-5-1K,0.03,1.35				
Minio		10/5/2K			
	time	objects	speed		Slowdowns
PUT	10.1 secs	314	61.9KB/sec	30.9 operations/sec	0
GET	0.8 secs	1570	3.7MB/sec	1870.5 operations/sec	0
DELETE	0.5 secs		690.4 deletes/sec		0
result csv	127-5-2K,0.06,3.65				

分析此表可知：随着线程数 thread 的增加，其 GET 的 objects 将会显著增多，其速度将有较明显的提升。随着规模 size 的增加，objects 数略微增长，其速度有显著提升，当 size 提升为原来的 2 倍，PUT 和 GET 的速度提升为原来的 2 倍还多。

- 对 Minio 的纵向分析（以 s3bench 为例）：固定其他参数不变，改变 numClients、numSamples 和 objectSize，结果如表 1-4 所示。

表 1-4 Minio 在不同参数下性能比较（s3bench）

Minio	10/100/1024		
	Transferred	Throughput	Duration
Write	0.098MB	0.04MB/s	2.724s
Read	0.098MB	2.28MB/s	0.043s
Delete	117.6891ms		
Minio	50/100/1024		
	Transferred	Throughput	Duration
Write	0.098MB	0.09MB/s	1.121s
Read	0.098MB	1.41MB/s	0.069s
Delete	130.5326ms		
Minio	50/500/1024		
	Transferred	Throughput	Duration
Write	0.488MB	0.06MB/s	7.874s
Read	0.488MB	2.10MB/s	0.232s
Delete	583.2976ms		
Minio	50/500/2048		
	Transferred	Throughput	Duration
Write	0.977MB	0.13MB/s	7.322s
Read	0.977MB	4.22MB/s	0.231s
Delete	805.8451ms		

分析此表可知：随着客户端数 numClients 的增加，其读写速度均有所提升，随着样本容量的 numSamples 增加，其写的速度会有所降低，但是读的速度会有所提升，随着对象规模 objectSize 的增加，其读写速度均有显著提升，当 objectSize 变为原来的 2 倍时，其读写速度变为原来的两倍还多。

- 对 s3proxy 的纵向分析（以 s3benchmark 为例）：固定其他参数不变，将时间延长为 10s，改变 size 和 thread，结果如表 1-5 所示。

表 1-5 s3proxy 在不同参数下性能比较（s3benchmark）

s3proxy	10/1/1K				
	time	objects	speed		Slowdowns
PUT	10.0 secs	6679	667.8KB/sec	667.8 operations/sec	0
GET	7.5 secs	6679	891.6KB/sec	891.6 operations/sec	0
DELETE	5.8 secs		1155.3 deletes/sec		0
result csv	127-1-1K,0.65,0.87				
s3proxy	10/5/1K				
	time	objects	speed		Slowdowns
PUT	10.0 secs	7310	730.8KB/sec	730.8 operations/sec	0
GET	10 secs	12625	1.2MB/sec	1262.4 operations/sec	0
DELETE	4.4 secs		1675.9 deletes/sec		0
result csv	127-5-1K,0.71,1.23				
s3proxy	10/5/2K				
	time	objects	speed		Slowdowns
PUT	10.0 secs	12169	2.4MB/sec	1216.3 operations/sec	0
GET	10 secs	14355	2.8MB/sec	1434.8 operations/sec	0
DELETE	6.8 secs		1779.4 deletes/sec		0
result csv	127-5-2K,2.38,2.80				

分析此表可知：随着线程数 thread 的增加，其 GET 的 objects 将会显著增多，其速度将有相应的提升。随着规模 size 的增加，Objects 数有相应增长，其速度有显著提升，当 size 提升为原来的 2 倍，速度提升为原来的 2 倍还多。

- 对 s3proxy 的纵向分析（以 s3bench 为例）：固定其他参数不变，改变 numClients、numSamples 和 objectSize，结果如表 1-6 所示。

表 1-6 s3proxy 在不同参数下性能比较（s3bench）

s3proxy 10/100/1024			
	Transferred	Throughput	Duration
Write	0.098MB	0.49MB/s	0.198s
Read	0.098MB	0.46MB/s	0.211s
Delete	1.035s		
s3proxy 50/100/1024			
	Transferred	Throughput	Duration
Write	0.098MB	0.89MB/s	0.109s
Read	0.098MB	1.07MB/s	0.092s
Delete	36.01702ms		
s3proxy 50/500/1024			
	Transferred	Throughput	Duration
Write	0.488MB	1.27MB/s	0.383s
Read	0.488MB	1.25MB/s	0.391s
Delete	133.72ms		
s3proxy 50/500/2048			
	Transferred	Throughput	Duration
Write	0.977MB	2.49MB/s	0.393s
Read	0.977MB	2.48MB/s	0.393s
Delete	130.6027ms		

分析此表可知：随着客户端数 numClients 的增加，其读写速度均有所提升，随着样本容量的 numSamples 增加，其读写速度也有提升，且读写速度几乎持平，随着对象规模 objectSize 的增加，其读写速度均有显著提升，当 objectSize 变为原来的 2 倍时，其读写速度变为原来的两倍以上。

6.2 心得体会

本次实验总的来说难度并不大，虽然一开始上手可能会有点懵不知道该做什么，不过搞明白了要做什么，跟着相应的步骤一步步做下来的话其实思路非常清晰明了，配服务器——配客户端——进行测试，三步走。

（题外话：这让我不禁想起了当初我搭建 `minecraft` 服务端的时候，就是自己去下载 `mc` 服务端、`mc` 客户端，然后对 `mc` 服务端进行内外网的配置等等，之前自己去配 `svn` 也是类似的步骤，需要在服务器上配置好服务端，然后在主机上配置好客户端，感觉其实很多东西都有其互通的地方。）

扯远了，通过简单比较其实可以发现本次实验也就多一个测试的步骤，而测试的结果分析也属于本次实验很重要的一个部分。

难确实不太难，思路也很清晰，就是配环境配了挺长时间，经常出现一些神奇而玄学的 `bug`，在 `windows` 下还好，各个软件的安装等等没有出现太大的问题，但是 `linux` 就出现了一个 `go` 语言装炸了的问题。使用 `sudo apt-get install golang` 命令安装的 `go` 语言版本很低，只有 1.6，而且安装的地址也很迷幻，`go version` 可以正常运行，但是想去下载 `github` 上的内容的时候就出现了 `GOPATH` 的环境配置问题，通过上网查阅，在 `/etc/profile` 加上相应的环境变量……也没有什么用 Orz，该有的问题一点儿没少。后来通过询问同学，决定卸载自动安装的 `golang`，转而去官网下载 1.12 的源码并解压来进行安装，事实证明还是源码靠谱！根据网上的源码教程一步步安装完 `go`，一点儿问题都没出，非常感动。

然后在运行测试工具那里，我发现我在 `windows` 下没有办法使用相对路径运行 `exe` 文件，只能每次都把绝对路径写上去，感觉有点儿麻烦，但是至少能运行我也就不去纠结了。

在数据分析那里，由于用了两个测试工具，所以工程量还蛮大的，于是就没有测非常多的数据进行画图，而是使用控制变量法两两比对得出结论，导致结论可能会存在一定的偏差，但是分析所得的结论大体上还是没有什么问题的。

我之前其实没有怎么用过 `github`，顶多就是 `download` 一些需要的资源，然后给资源提供者标一个 `star` 关注一下，然后就没有然后了。自己的话，之前也没有使用过 `git`，这次实验让我对 `github` 有了更深的理解，同时认识到了 `git` 这个神奇的操作！感觉自己之前不怎么关注 `github`，不怎么去了解 `git` 好亏呀，因为 `github` 自带版本管理，因此在之后可能遇到的多人协作方面就非常有用，而且很轻便，注意一下不要冲突更改就可以了，感觉会非常好用。这也让我在一定程度上明白了为什么会有人说不会用 `github` 的不能算程序员（这话本身正确与否不论）。

总的来说，本次实验我觉得非常有趣，我做的挺开心的，也学到了挺多东西，认识了解了许多软件和工具，感觉收获非常大呀，很喜欢这个实验！

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.