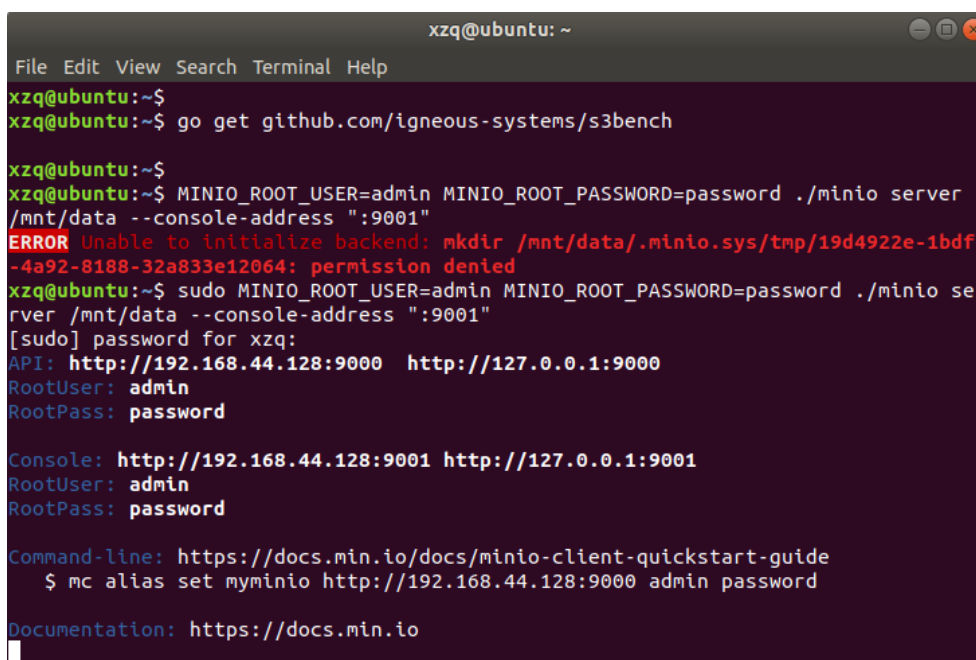


实验一：系统搭建

在Linux虚拟机（Ubuntu 18.04.5）中搭建minio服务器

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
chmod +x minio
MINIO_ROOT_USER=admin MINIO_ROOT_PASSWORD=password ./minio server /mnt/data --
console-address ":9001"
```



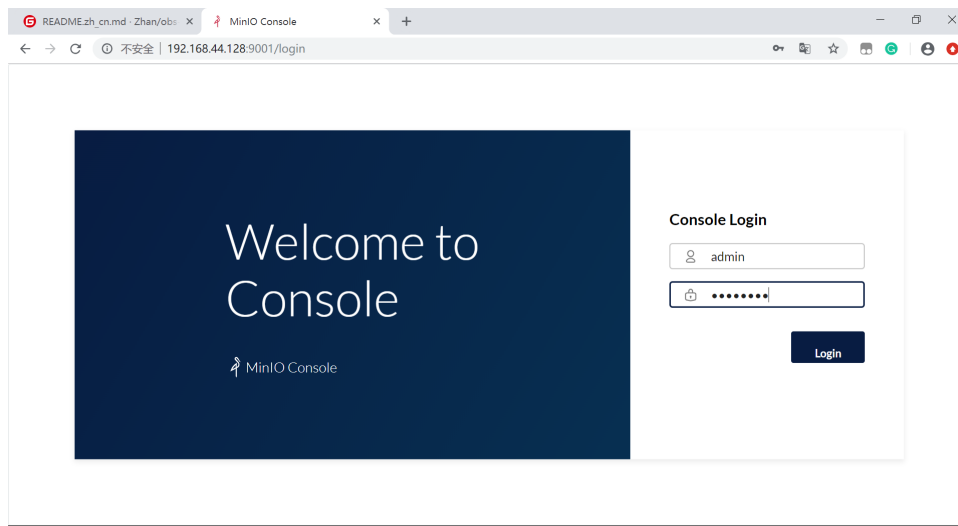
```
xzq@ubuntu: ~
File Edit View Search Terminal Help
xzq@ubuntu:~$
xzq@ubuntu:~$ go get github.com/igneous-systems/s3bench
xzq@ubuntu:~$
xzq@ubuntu:~$ MINIO_ROOT_USER=admin MINIO_ROOT_PASSWORD=password ./minio server
/mnt/data --console-address ":9001"
ERROR Unable to initialize backend: mkdir /mnt/data/.minio.sys/tmp/19d4922e-1bdf
-4a92-8188-32a833e12064: permission denied
xzq@ubuntu:~$ sudo MINIO_ROOT_USER=admin MINIO_ROOT_PASSWORD=password ./minio se
rver /mnt/data --console-address ":9001"
[sudo] password for xzq:
API: http://192.168.44.128:9000 http://127.0.0.1:9000
RootUser: admin
RootPass: password

Console: http://192.168.44.128:9001 http://127.0.0.1:9001
RootUser: admin
RootPass: password

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc alias set myminio http://192.168.44.128:9000 admin password

Documentation: https://docs.min.io
```

在内网中的浏览器地址栏输入地址，访问客户端。

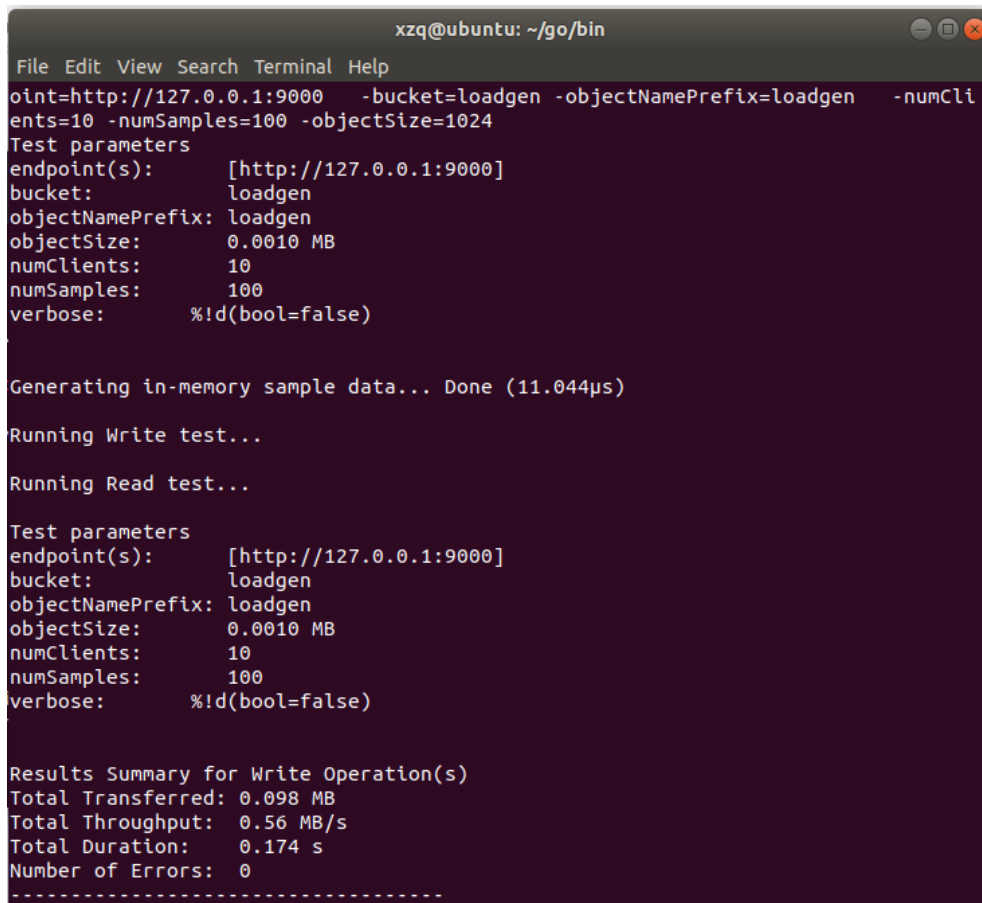


实验二：性能观测

配置go环境，新建一个bucket，使用s3bench作为性能测试工具，配置bucket、objectSize等参数运行命令。

```
s3bench \  
-accessKey=admin \  
-accessSecret=password \  
-bucket=loadgen \  
-endpoint=http://127.0.0.1:9000 \  
-numClients=10 \  
-numSamples=100 \  
-objectNamePrefix=loadgen \  
-objectSize=1024
```

go的编译需要在连得上github的网络环境下进行。

A terminal window titled 'xzq@ubuntu: ~/go/bin' showing the execution of s3bench. The command line at the top is: 'oint=http://127.0.0.1:9000 -bucket=loadgen -objectNamePrefix=loadgen -numClients=10 -numSamples=100 -objectSize=1024'. The output shows test parameters, data generation time (11.044µs), and results for a write operation. The results summary indicates a total transfer of 0.098 MB, a throughput of 0.56 MB/s, a duration of 0.174 s, and zero errors.

```
xzq@ubuntu: ~/go/bin  
File Edit View Search Terminal Help  
oint=http://127.0.0.1:9000 -bucket=loadgen -objectNamePrefix=loadgen -numCli  
ents=10 -numSamples=100 -objectSize=1024  
Test parameters  
endpoint(s):      [http://127.0.0.1:9000]  
bucket:           loadgen  
objectNamePrefix: loadgen  
objectSize:       0.0010 MB  
numClients:       10  
numSamples:       100  
verbose:          %!d(bool=false)  
  
Generating in-memory sample data... Done (11.044µs)  
Running Write test...  
Running Read test...  
  
Test parameters  
endpoint(s):      [http://127.0.0.1:9000]  
bucket:           loadgen  
objectNamePrefix: loadgen  
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:  0.56 MB/s  
Total Duration:    0.174 s  
Number of Errors:  0  
-----
```

写操作最大的耗时为44ms，最小仅为4ms；90%的写操作在29ms内完成，吞吐量为56MB/s，可以看出一定的长尾效应。同理，读操作结果如下图所示。

```
xzq@ubuntu: ~/go/bin
File Edit View Search Terminal Help
numClients:      10
numSamples:      100
verbose:         %!d(bool=false)

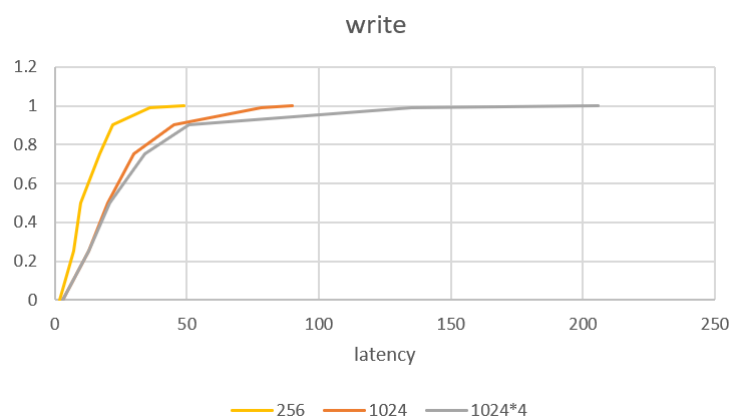
Results Summary for Write Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  0.56 MB/s
Total Duration:    0.174 s
Number of Errors:  0
-----
Write times Max:      0.044 s
Write times 99th %ile: 0.044 s
Write times 90th %ile: 0.029 s
Write times 75th %ile: 0.021 s
Write times 50th %ile: 0.015 s
Write times 25th %ile: 0.010 s
Write times Min:      0.004 s

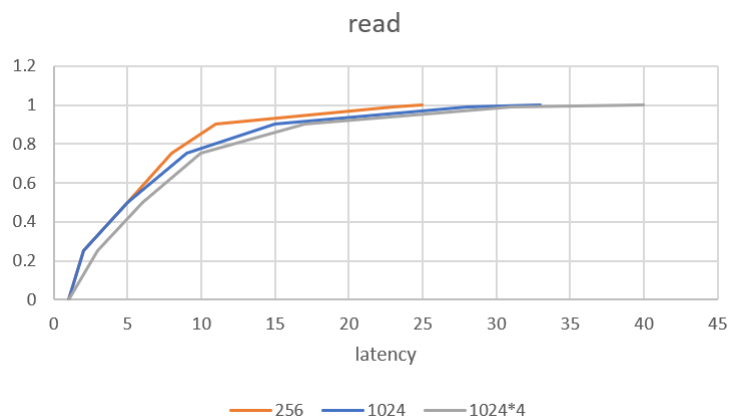
Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  0.92 MB/s
Total Duration:    0.106 s
Number of Errors:  0
-----
Read times Max:      0.035 s
Read times 99th %ile: 0.035 s
Read times 90th %ile: 0.026 s
Read times 75th %ile: 0.016 s
Read times 50th %ile: 0.006 s
Read times 25th %ile: 0.003 s
Read times Min:      0.001 s

Cleaning up 100 objects...
Deleting a batch of 100 objects in range {0, 99}... Succeeded
Successfully deleted 100/100 objects in 52.388742ms
xzq@ubuntu:~/go/bin$
```

修改参数，绘制曲线，结果如图表所示。可以看出对象尺寸越大，长尾现象越明显，吞吐量越大。

指标：吞吐率Throughput、延迟Latency，以及环境参数：对象尺寸object size、并发性、服务器数量。

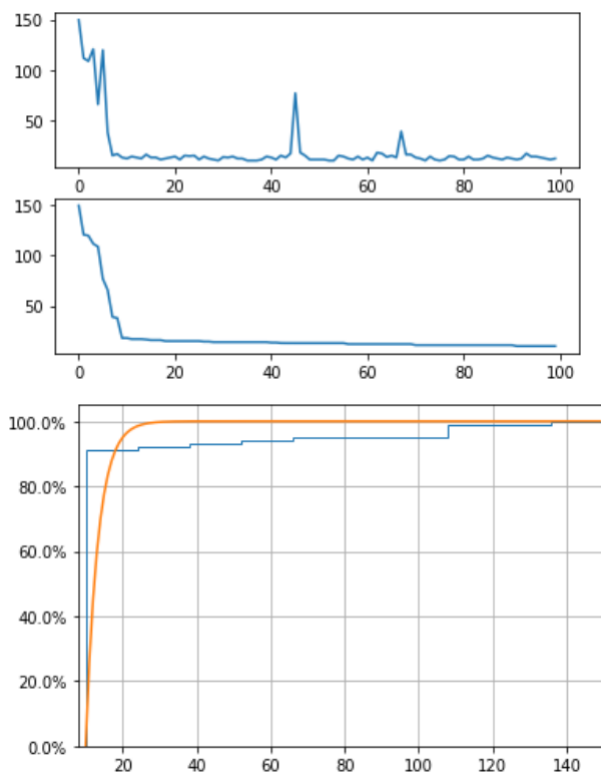




throughput	256	1024	1024*4
write	0.18	0.67	2.00
read	0.062	1.36	2.15

实验三：尾延迟挑战

修改并运行python脚本，当有100个上传项时，尾延迟分布数据以及排队论模型拟合如下图所示，小部分的数据的写用时远超其他数据。



尝试对冲请求，客户端一旦收到第一个结果则取消发送剩余请求，结果如下图所示。

依旧存在尾延迟，但延迟有所改善。

