



2016 级

《物联网数据存储与管理》课程

# 实 验 报 告

姓 名 肖奎

学 号 U201614905

班 号 物联网 1601 班

日 期 2019.05.26



# 目 录

一、实验目的.....	1
二、实验背景.....	1
三、实验环境.....	2
四、实验内容.....	2
4.1 对象存储技术实践.....	3
4.2 对象存储性能分析.....	3
五、实验过程.....	3
5.1 minio 服务端的安装与运行.....	3
5.2 minio 客户端的安装与测试.....	4
5.3 S3bench 的安装与测试.....	6
5.4 使用 S3bench 对云存储性能进行测试.....	7
5.5 结果分析.....	10
六、实验总结.....	11
参考文献.....	11



## 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

对象存储是一种计算机数据存储架构，它将数据管理为对象，而不是像文件系统那样管理数据作为文件层次结构的其他存储架构，以及将数据作为扇区和路径中的块来管理数据的块存储。每个对象通常包括数据本身、一个变量元数据和一个全局唯一标识符。对象存储可以在多个级别实现，包括设备级别、系统级别和接口级别。在每种情况下，对象存储都试图启用其他存储架构所没有处理的功能，比如可以通过应用程序直接编程的接口，一个可以跨越物理硬件多个实例的名称空间，以及数据管理功能，如数据复制和对象级粒度的数据分布。

本次实验为对象存储入门实验，其中主要的部分有：基础环境搭建；对象存储服务器端准备；对象存储客户端准备；对象存储测评工具的使用。

Minio：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。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。

mc ( minio client ) 提供了一些 UNIX 常用命令的替代品，像 ls, cat, cp, mirror, diff 这些。它支持文件系统和亚马逊 S3 云存储服务。

S3bench 是一个 go 语言编写的测试程序。它可以针对 S3 兼容端点运行非常基

本的吞吐量基准测试，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，使用了 AWS Go SDK。

### 三、实验环境

#### 1、操作系统



#### 2、go 环境

```
aloha@aloha-hello-world:~$ go version
go version go1.12.5 linux/amd64
```

### 四、实验内容

首先搭建实验环境，安装 go 语言环境。然后安装 minio 服务器端和客户端。最后运行 S3Bench 测试程序，分析各项指标。Go 环境安装版本如图 4.1。

```
aloha@aloha-hello-world:~/lab$ go version
go version go1.12.5 linux/amd64
```

图 4.1 安装 go 环境

## 4.1 对象存储技术实践

1. 在本机上搭建一个 Minio 服务器端，将本机作为一个服务器。服务器存储空间即为本机的存储空间。
2. 在本机开启一个 Minio 客户端。可通过客户端对服务器内的文件进行操作。
3. 运行 S3Bench 来对其进行测评。

## 4.2 对象存储性能分析

对 Minio 服务器运行 S3Bench 进行测试。观察各项性能数据指标，修改测试代码，反复测试。

# 五、实验过程

## 5.1 minio 服务端的安装与运行

- 1) 从官方网站下载了 minio 后，得到一个可执行文件，如图 5.1 所示。

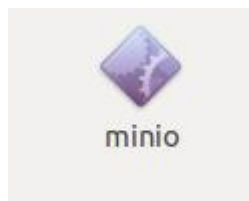


图 5.1 minio 服务端安装

- 2) 运行 minio 服务端程序

```
Endpoint: http://10.11.176.206:9000 http://172.17.0.1:9000 http://127.0.0.1:9000
AccessKey: MUXRBFTZ07XL8NVPRCVD
SecretKey: j+cEnqqIPH2s3hoXANUx+K4Xhs2ExLHo04qFGxxj

Browser Access:
http://10.11.176.206:9000 http://172.17.0.1:9000 http://127.0.0.1:9000

Command-line Access: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc config host add myminio http://10.11.176.206:9000 MUXRBFTZ07XL8NVPRCVD j+cEnqqIPH2s3hoXANUx+K4Xhs2ExLHo04qFGxxj

Object API (Amazon S3 compatible):
Go: https://docs.min.io/docs/golang-client-quickstart-guide
Java: https://docs.min.io/docs/java-client-quickstart-guide
Python: https://docs.min.io/docs/python-client-quickstart-guide
JavaScript: https://docs.min.io/docs/javascript-client-quickstart-guide
.NET: https://docs.min.io/docs/dotnet-client-quickstart-guide
```

图 5.2 开启 minio 服务端

3) 在浏览器输入 `http://127.0.0.1:9000` 进入服务端页面,输入 Accesskey 与 Secretkey ,结果如图 5.3 所示。

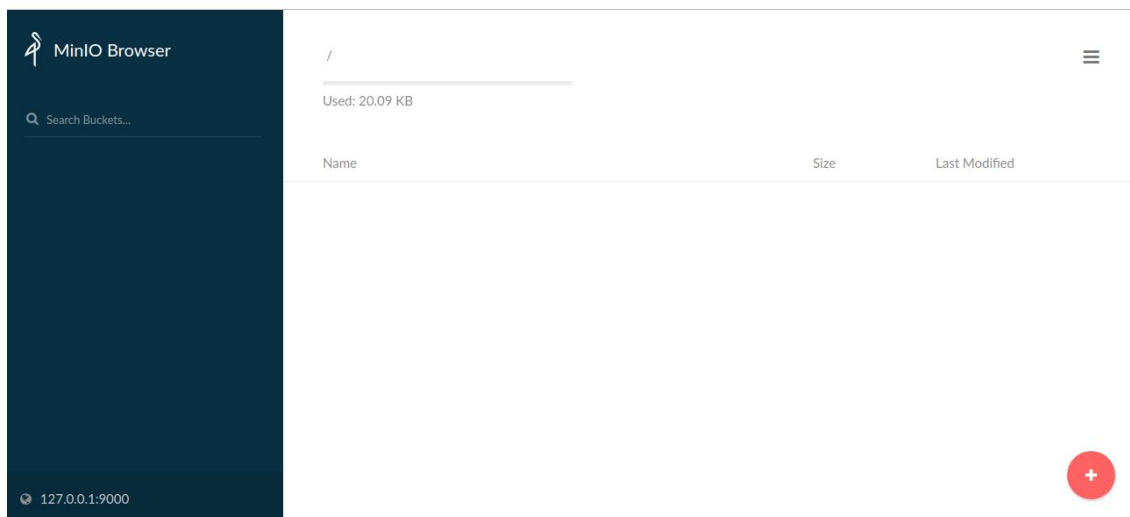


图 5.3 minio 浏览器界面

## 5.2 minio 客户端的安装与测试

1) 从官网下载 minio 客户端得到 mc 可执行程序, 如图 5.4。

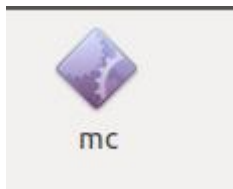


图 5.4 客户端程序

2) 使用命令 `./mc --help` 查看 mc 客户端可用参数及其描述。如图 5.5



```

aloha@aloha-hello-world:~/lab$ ./mc --help
NAME:
  mc - MinIO Client for cloud storage and filesystems.

USAGE:
  mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]

COMMANDS:
  ls      list buckets and objects
  mb      make a bucket
  rb      remove a bucket
  cat     display object contents
  head   display first 'n' lines of an object
  pipe   stream STDIN to an object
  share  generate URL for temporary access to an object
  cp     copy objects
  mirror synchronize object(s) to a remote site
  find   search for objects
  sql    run sql queries on objects
  stat   show object metadata
  diff   list differences in object name, size, and date between two buckets
  rm     remove objects
  event  configure object notifications
  watch  listen for object notification events
  policy manage anonymous access to buckets and objects
  admin  manage MinIO servers
  session resume interrupted operations
  config configure MinIO client
  update update mc to latest release
  version show version info

GLOBAL FLAGS:
  --config-dir value, -C value path to configuration folder (default: "/home/aloha/.mc")
  --quiet, -q                 disable progress bar display
  --no-color                  disable color theme
  --json                      enable JSON formatted output

```

图 5.5 mc --help 执行结果

3) 使用 mc 配置主机并创建 bucket。结果如图 5.6。

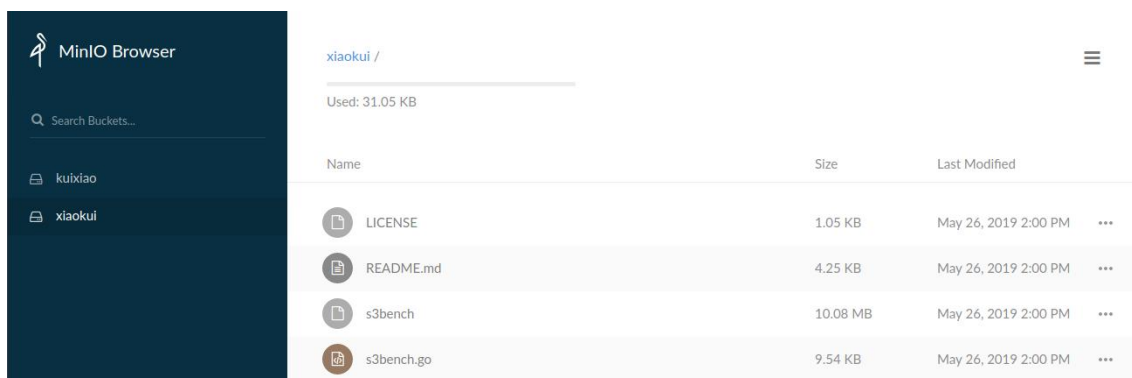
```

aloha@aloha-hello-world:~/lab$ ./mc config host add myminio http://127.0.0.1:9000
0 MUXRBFTZ07XL8NVPRCVD j+cEnqQIPH2s3hoXANUX+K4Xhs2ExLHo04qFGxxj
Added `myminio` successfully.
aloha@aloha-hello-world:~/lab$ ./mc mb myminio/xiaokui
Bucket created successfully `myminio/xiaokui`.

```

图 5.6 创建 butket

4) 在 minio 浏览器界面上传文件，使用客户端 mc 查看，结果如图 5.7，5.8。



Name	Size	Last Modified
LICENSE	1.05 KB	May 26, 2019 2:00 PM
README.md	4.25 KB	May 26, 2019 2:00 PM
s3bench	10.08 MB	May 26, 2019 2:00 PM
s3bench.go	9.54 KB	May 26, 2019 2:00 PM

图 5.7 上传文件

```

aloha@aloha-hello-world:~/lab$ ./mc ls myminio/xiaokui
aloha@aloha-hello-world:~/lab$ ./mc ls myminio/xiaokui
[2019-05-26 14:00:38 CST] 1.0KiB LICENSE
[2019-05-26 14:00:38 CST] 4.2KiB README.md
[2019-05-26 14:00:38 CST] 10MiB s3bench
[2019-05-26 14:00:38 CST] 9.5KiB s3bench.go
aloha@aloha-hello-world:~/lab$

```

图 5.8 查看上传文件

### 5.3 S3bench 的安装与测试

1) 使用 go get 命令安装 S3bench 测试程序，结果如图 5.6。



图 5.9 安装 S3bench 测试程序

2) 执行 s3bench，查看测试结果。

```

aloha@aloha-hello-world:~/golang/go/src/github.com/igneous-systems/s3bench$ ./s3
bench -accessKey=MUXRBFTZ07XL8NVPRCVD -accessSecret=j+cEnqqIPH2s3hoXANUX+K4Xhs2E
xLHo04qFGxxj -bucket=xiaokui -endpoint=http://127.0.0.1:9000 -numClients=2 -numS
amples=10 -objectNamePrefix=xiaokui -objectSize=1024
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:          xiaokui
objectNamePrefix: xiaokui
objectSize:      0.0010 MB
numClients:      2
numSamples:      10
verbose:         %!d(bool=false)

Generating in-memory sample data... Done (104.09µs)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:          xiaokui
objectNamePrefix: xiaokui
objectSize:      0.0010 MB
numClients:      2
numSamples:      10
verbose:         %!d(bool=false)

```

图 5. 10 S3bench 测试结果 1

```
Results Summary for Write Operation(s)
Total Transferred: 0.010 MB
Total Throughput: 0.17 MB/s
Total Duration: 0.056 s
Number of Errors: 0
-----
Write times Max: 0.019 s
Write times 99th %ile: 0.019 s
Write times 90th %ile: 0.019 s
Write times 75th %ile: 0.012 s
Write times 50th %ile: 0.010 s
Write times 25th %ile: 0.008 s
Write times Min: 0.008 s

Results Summary for Read Operation(s)
Total Transferred: 0.010 MB
Total Throughput: 0.60 MB/s
Total Duration: 0.016 s
Number of Errors: 0
-----
Read times Max: 0.004 s
Read times 99th %ile: 0.004 s
Read times 90th %ile: 0.004 s
Read times 75th %ile: 0.003 s
Read times 50th %ile: 0.003 s
Read times 25th %ile: 0.003 s
Read times Min: 0.002 s

Cleaning up 10 objects...
Deleting a batch of 10 objects in range {0, 9}... Succeeded
Successfully deleted 10/10 objects in 5.108734ms
```

图 5. 11S3bench 测试结果 2

由图 5.11 可看出读写速度分别为 0.17M/s 和 0.60M/s，读操作快于写操作。

## 5.4 使用 S3bench 对云存储性能进行测试

1) 增加存储对象大小，设置为 100MB。

运行 S3bench 测试程序。如图 5.12 所示：

```
aloha@aloha-hello-world:~/golang/go/src/github.com/igneous-systems/s3bench$ ./s3bench -accessKey=MUXRBFTZ07XL8NVPRCVD -accessSecret=j+cEnqqIPH
2s3hoXANUx+K4Xhs2ExLHo04qFGxxj -bucket=xlaokui -endpoint=http://127.0.0.1:9000 -numClients=2 -numSamples=10 -objectNamePrefix=xlaokui -objectS
ize=104857600
Test parameters
endpoint(s): [http://127.0.0.1:9000]
bucket: xlaokui
objectNamePrefix: xlaokui
objectSize: 100.0000 MB
numClients: 2
numSamples: 10
verbose: %d(bool=false)
```

图 5. 12 设置存储对象大小为 100M

测试结果如图 5.13：



```

Results Summary for Write Operation(s)
Total Transferred: 1000.000 MB
Total Throughput: 280.83 MB/s
Total Duration: 3.561 s
Number of Errors: 0
-----
Write times Max: 1.093 s
Write times 99th %ile: 1.093 s
Write times 90th %ile: 1.093 s
Write times 75th %ile: 0.967 s
Write times 50th %ile: 0.519 s
Write times 25th %ile: 0.501 s
Write times Min: 0.490 s

Results Summary for Read Operation(s)
Total Transferred: 1000.000 MB
Total Throughput: 2359.00 MB/s
Total Duration: 0.424 s
Number of Errors: 0
-----
Read times Max: 0.162 s
Read times 99th %ile: 0.162 s
Read times 90th %ile: 0.162 s
Read times 75th %ile: 0.075 s
Read times 50th %ile: 0.067 s
Read times 25th %ile: 0.066 s
Read times Min: 0.058 s

Cleaning up 10 objects...
Deleting a batch of 10 objects in range {0, 9}... Succeeded
Successfully deleted 10/10 objects in 119.832936ms

```

图 5.13 测试结果 1

2) 增加客户端数量，设置客户端数量为 4。

执行 S3bench 测试程序，如图 5.14：

```

aloha@aloha-hello-world:~/golang/go/src/github.com/igneous-systems/s3bench$ ./s3
bench -accessKey=MUXRBFTZ07XL8NVPRCVD -accessSecret=j+cEnqqIPH2s3hoXANUx+K4Xhs2E
xLHo04qFGxxj -bucket=xiaokui -endpoint=http://127.0.0.1:9000 -numClients=4 -numS
amples=10 -objectNamePrefix=xiaokui -objectSize=104857600
Test parameters
endpoint(s): [http://127.0.0.1:9000]
bucket: xiaokui
objectNamePrefix: xiaokui
objectSize: 100.0000 MB
numClients: 4
numSamples: 10
verbose: %!d(bool=false)

```

图 5.14 设置客户端数量为 4

测试结果如图 5.15：

```

Results Summary for Write Operation(s)
Total Transferred: 1000.000 MB
Total Throughput: 399.65 MB/s
Total Duration: 2.502 s
Number of Errors: 0
-----
Write times Max: 1.081 s
Write times 99th %ile: 1.081 s
Write times 90th %ile: 1.081 s
Write times 75th %ile: 0.970 s
Write times 50th %ile: 0.808 s
Write times 25th %ile: 0.722 s
Write times Min: 0.611 s

Results Summary for Read Operation(s)
Total Transferred: 1000.000 MB
Total Throughput: 2603.61 MB/s
Total Duration: 0.384 s
Number of Errors: 0
-----
Read times Max: 0.239 s
Read times 99th %ile: 0.239 s
Read times 90th %ile: 0.239 s
Read times 75th %ile: 0.189 s
Read times 50th %ile: 0.129 s
Read times 25th %ile: 0.100 s
Read times Min: 0.077 s

Cleaning up 10 objects...
Deleting a batch of 10 objects in range {0, 9}... Succeeded
Successfully deleted 10/10 objects in 119.506797ms

```

图 5.15 测试结果 2

3) 增加样本数量，设置样本数量为 20

执行 S3bench 测试程序，如图 5.16：

```

aloha@aloha-hello-world:~/golang/go/src/github.com/igneous-systems/s3bench$ ./s3
bench -accessKey=MUXRBFTZ07XL8NVPKVD -accessSecret=j+cEnqqIPH2s3hoXANUx+K4Xhs2E
xLHo04qFGxxj -bucket=xiaokui -endpoint=http://127.0.0.1:9000 -numClients=4 -numS
amples=20 -objectNamePrefix=xiaokui -objectSize=104857600
Test parameters
endpoint(s): [http://127.0.0.1:9000]
bucket: xiaokui
objectNamePrefix: xiaokui
objectSize: 100.0000 MB
numClients: 4
numSamples: 20
verbose: %!d(bool=false)

```

图 5.16 设置样本数量为 20

测试结果如图 5.17：

```
Results Summary for Write Operation(s)
Total Transferred: 2000.000 MB
Total Throughput: 322.56 MB/s
Total Duration: 6.200 s
Number of Errors: 0
-----
Write times Max: 1.805 s
Write times 99th %ile: 1.805 s
Write times 90th %ile: 1.798 s
Write times 75th %ile: 1.686 s
Write times 50th %ile: 1.038 s
Write times 25th %ile: 0.963 s
Write times Min: 0.642 s

Results Summary for Read Operation(s)
Total Transferred: 2000.000 MB
Total Throughput: 3194.51 MB/s
Total Duration: 0.626 s
Number of Errors: 0
-----
Read times Max: 0.150 s
Read times 99th %ile: 0.150 s
Read times 90th %ile: 0.148 s
Read times 75th %ile: 0.137 s
Read times 50th %ile: 0.123 s
Read times 25th %ile: 0.110 s
Read times Min: 0.080 s

Cleaning up 20 objects...
Deleting a batch of 20 objects in range {0, 19}... Succeeded
Successfully deleted 20/20 objects in 226.331163ms
```

图 5.17 测试结果 3

## 5.5 结果分析

测试结果如下表所示：

objectSize	numClients	numSamples	读速度	写速度	Errors
0.1M	2	10	0.60M/	0.17M/s	0
100M	2	10	2359.00 MB/s	280.83 MB/s	0
200M	2	10	2669.03 MB/s	281.35 MB/s	0

100M	4	10	2603.61 MB/s	399.65 MB/s	0
100M	4	20	3194.51 MB/s	322.56 MB/s	0

由表可知，初步增大存储对象大小可以显著提高读写速率。其次，客户端数量与样本数量的增加也会略微增加读写速率。在内存充足情况下，未发现有读写操作失败的情况。

## 六、实验总结

本次实验主要是针对对象存储的入门级实验。由于缺乏对对象存储的认识，实验基本上是按部就班的操作一遍，围绕着对象存储服务器的部署、配置以及测试来展开。

而在这次实验中，给我感受最深的是云存储的便利性。通过客户端与云存储服务器的交互，用户无需在本地进行大批量数据的存储管理，而是借助云平台，实现随时随地的存取。在 5G 与物联网快速兴起的今天，这种存储方式将发挥重要作用。

另一方面，本次实验也让我接触了 go 语言、git 和 docker 等工具，通过对相关资料和文献的阅读，学习到了许多，极大激发了我的学习兴趣。

## 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.