



2017 级

《物联网数据存储与管理》课程

实 验 报 告

姓 名 陈望

学 号 U201614903

班 号 物联网 1701 班

日 期 2020.06.01

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	3
5.1 搭建 Minio 服务器	3
5.2 安装 Minio 客户端	3
5.3 安装 S3Bench	5
5.4 性能测试及其结果	5
六、实验总结	9
参考文献	9

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

存储局域网（SAN）和网络附加存储（NAS）是目前两种主流网络存储架构，而对象存储(OSD)是一种新的网络存储架构。总体上来讲，对象存储（Object-Based Storage, OBS)综合了 NAS 和 SAN 的优点，同时具有 SAN 的高速直接访问和 NAS 的分布式数据共享等优势，提供了具有高性能、高可靠性、跨平台以及安全的数据共享的存储体系结构。

对象存储架构的核心是将数据通路（数据读或写）和控制通路（元数据）分离，并且基于对象存储设备构建存储系统，每个对象存储设备具有一定的职能，能够自动管理其上的数据分布。对象存储结构由对象、对象存储设备、元数据服务器、对象存储系统的客户端四部分组成。其结构如下图 2.1 所示：

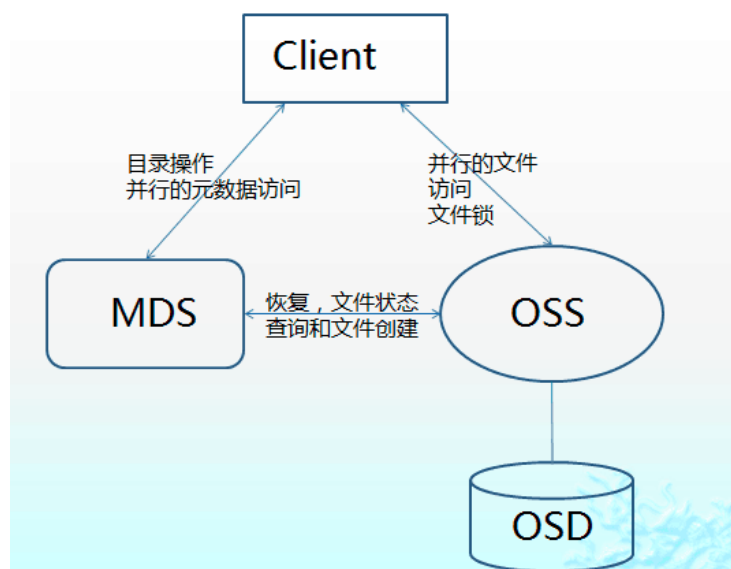


图 2.1 对象存储结构

三、实验环境

本次实验在 Linux 环境下进行。
操作系统如下图 3.1 所示：



图 3.1 操作系统版本及其参数

go 语言环境如下图 3.2 所示：

```
chenwang@aloha-hello-world:~$ go version  
go version go1.14.3 linux/amd64
```

图 3.2 go 语言版本

四、实验内容

本次实验为对象存储入门实验，其中主要内容有：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。本实验选择 Minio 的服务器和客户端，评测工具选择 S3Bench。

具体实验步骤为：搭建 go 语言环境，然后安装 Minio 服务器和客户端，最后运行 S3Bench 测试程序进行测试，分析各项指标。

4.1 对象存储技术实践

1. 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. 安装一个 Minio 客户端，可通过客户端对服务器内的文件进行操作。
3. 上传测试文件测试服务器和客户端搭建成功

4.2 对象存储性能分析

1. 对 Minio 服务器运行 S3Bench 程序进行测试。
2. 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 搭建 Minio 服务器

- 1、在 Minio 官网下载 Minio 服务器，即可得到一个 minio 可执行文件，直接搭建 minio 服务器。命令行如下：

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
chmod +x minio
./minio server /mnt/data
```

- 2、运行 minio 服务器，结果如下图 5.1 所示：

```
Endpoint: http://10.11.176.206:9000 http://172.17.0.1:9000 http://127.0.0.1:9000
AccessKey: MUXRBFTZ07XL8NVPRCVD
SecretKey: j+cEnqIPH2s3hoXANUX+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+cEnqIPH2s3hoXANUX+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 5.1 运行 minio 服务器

- 3、打开浏览器，输入 <http://127.0.0.1:9000> 进入服务器登录界面，输入 AccessKey 和 SecretKey 登录，结果如下图 5.2 所示

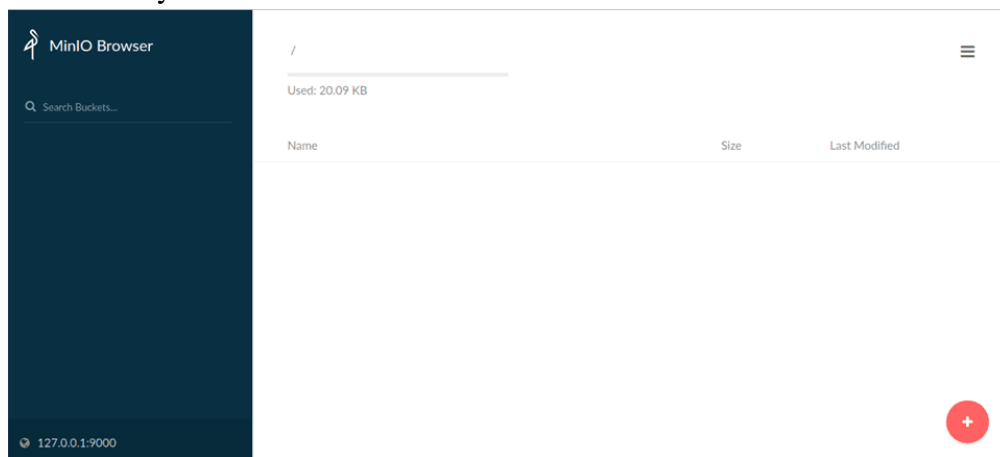


图 5.2 登录 minio 服务器页面

5.2 安装 Minio 客户端

1. 在 Minio 官网下载 Minio 客户端，得到名为 mc 的可执行文件。命令行如下：

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
chmod +x mc
./mc --help
```

2. 输入命令行 `./mc -help` 查看 Minio 客户端可用参数。结果如下图 5.3 所示：

```
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 5.3 minio 客户端参数

3. 使用 minio client 配置主机并创建 bucket。创建结果如下图所示：

```
aloha@aloha-hello-world:~/lab
Bucket created successfully
```

4. 在 Minio 服务器上上传文件，结果如下图 5.4 所示：





Used: 31.05 KB	
Name	Size
 LICENSE	1.05 KB
 README.md	4.25 KB
 s3bench	10.08 MB
 s3bench.go	9.54 KB

图 5.4 上传文件信息

5.3 安装 S3Bench

1. 使用 `go get` 命令安装 S3Bench 测试程序，安装结果如下图 5.5 所示：

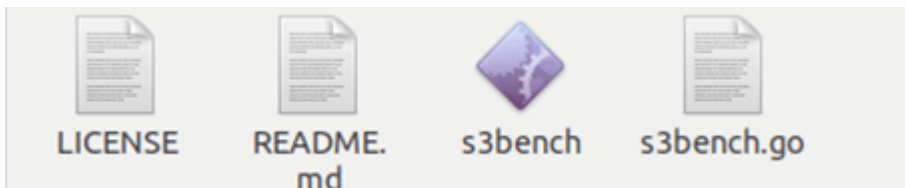


图 5.5 S3Bench 安装结果

2. 运行 S3Bench 程序，查看测试结果。
设置对象存储大小为 0.001M，客户端数量为 2，样本数量为 10。
执行程序，结果如下图 5.6 所示：

```
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 5.6 S3Bench 执行结果

由上图可知，读操作(0.60MB/s)比写操作(0.17MB/s)快。

5.4 性能测试及其结果

1. 增加对象存储大小，设置为 100M。客户端数量为 2，样本数量为 10，保持不变。
执行 S3Bench 测试程序，结果如下图 5.7 所示：


```
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 5.7 对象存储大小为 100M 的测试结果

2. 增加客户端数量，设置为 4。对象存储大小为 100M，样本数量为 10，保持不变。
执行 S3Bench 测试程序，结果如下图 5.8 所示：


```
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 5.8 客户端数量为 4 的测试结果

3. 增加样本数量，设置为 20。对象存储大小为 100M，客户端数量为 4，保持不变。
执行 S3Bench 测试程序，结果如下图 5.9 所示：

```

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 5.9 样本数为 20 的测试结果

4. 结果分析。

为便于观察和总结实验结果间的规律，将测试结果综合到一个表格中，如下表 5.1 所示

表 5.1 测试结果综合

对象存储大小	客户端数量	样本数量	写速度	读速度
0.001M	2	10	0.17M/s	0.60M/
100M	2	10	280.83 MB/s	2359.00 MB/s
100M	4	10	399.65 MB/s	2603.61 MB/s
100M	4	20	322.56 MB/s	3194.51 MB/s
200M	2	10	281.35 MB/s	2669.03 MB/s

注：对象大小 200M 的测试结果是相比较能够综合比较而临时做的测试。

由上表可知，对象存储大小增大，读写速度是呈指数型增长的，即初期随着对象存储大小的增大，读写速度快速增加，而当存储大小超过一定限制之后，读写速度增长速度变慢。客户端数量和样本数量也会影响读写速度，一般情况下，客户端数量或者样本数量增加，会小幅度增加读写速

度。

六、实验总结

本次实验是对象存储的入门实验，由于对对象存储概念及架构不太熟悉，缺乏认识，在实验初期不是很顺手，基本上就是按照老师的操作步骤再结合一点网上的资料，依瓢画葫芦的操作了一遍，没有太深入的了解和实践，但还是受益匪浅。此次实验让我对 go 语言和 git 有了一定的认识，算是打下了基础，虽然也不是很熟悉，但通过阅读一些网上的资料还是学到了很多。

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.