



2020 级

《大数据存储与管理》课程

# 实 验 报 告

姓 名 赵昕阳

学 号 U202015481

班 号 物联网 2001 班

日 期 2023.05.29

# 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 .....	1
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 .....	2
参考文献 .....	5

## 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，分析性能问题，架设应用实践。

## 二、实验背景

Minio 是一个基于 Go 语言的对象存储服务。它实现了大部分亚马逊 S3 云存储服务接口，可以看做是 S3 的开源版本，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。区别于分布式存储系统，minio 的特色在于简单、轻量级，对开发者友好，认为存储应该是一个开发问题而不是一个运维问题。

Minio Client 简称 mc，是 minio 服务器的客户端，对 ls, cat, cp, mirror, diff, find 等 UNIX 命令提供了一种替代方案，它支持文件系统和兼容 Amazon S3 的云存储服务（AWS Signature v2 和 v4）

S3bench 提供了针对兼容 S3 的端点运行非常基本的吞吐量基准测试的功能。它执行一系列的 put 操作，然后执行一系列的 get 操作，并显示相应的统计信息。该工具使用 AWS Go SDK。

## 三、实验环境

硬件环境：

操作系统	Windows10
Python	3

### 设备规格

设备名称	DESKTOP-U0UPLP0
处理器	AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz
机带 RAM	16.0 GB (15.4 GB 可用)

图 3.1 设备规格

软件环境：

对象存储客户端：Minio

对象存储服务器端：mc

对象存储测试工具：s3-bench

## 四、实验内容

本次实验是对象存储技术相关实验，使用 minio、minio client、和 s3bench 等工具完成对数据的存储和分析。

## 4.1 对象存储技术实践

- 1.配置 Minio。向配置好的客户端发送文件。
- 2.配置 mc。在远端利用 mc 新建 bucket。
- 3.下载 s3bench 脚本，修改相关参数进行相应的性能测试。

## 4.2 对象存储性能分析

- 1.对 s3bench 脚本运行输出的结果进行相应的记录和处理。
- 2.改变不同参数观察不同的性能指标。
- 3.对相应的实验数据做出记录。

## 五、实验过程

1.在 Windows 系统上下载对应的 Minio 服务器，下载完成后讲 minio.exe 移动至 F:/目录下。

2.不要双击 minio.exe，打开 cmd，进入到 F:/目录下，新建 minioData 文件夹用于存储 minio 上传的文件目录。输入 minio server F:/minioData 启动 minio 服务。

```
F:\>minio.exe server F:/minioData
WARNING: Detected default credentials 'minioadmin:minioadmin', we recommend that you change these values with 'MINIO_ROOT_USER' and 'MINIO_ROOT_PASSWORD' environment variables
MinIO Object Storage Server
Copyright: 2015-2023 MinIO, Inc.
License: GNU AGPLv3 <https://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2023-04-20T17-56-55Z (go1.20.3 windows/amd64)

Status:          1 Online, 0 Offline.
S3-API: http://169.254.95.225:9000 http://10.11.183.44:9000 http://169.254.51.25:9000 http://127.0.0.1:9000

RootUser: minioadmin
RootPass: minioadmin

Console: http://169.254.95.225:53214 http://10.11.183.44:53214 http://169.254.51.25:53214 http://127.0.0.1:53214
RootUser: minioadmin
RootPass: minioadmin

Command-line: https://min.io/docs/minio/linux/reference/minio-mc.html#quickstart
$ mc.exe alias set myminio http://169.254.95.225:9000 minioadmin minioadmin

Documentation: https://min.io/docs/minio/linux/index.html
Warning: The standard parity is set to 0. This can lead to data loss.
```

图 5.1

由图我们可以得到对应的服务端的调用的 URL 为本机网络 IP 的 9000 端口，而我们的服务端控制台为本地网络 IP 的 53214 端口。RootUser 和 RootPass 是用名和密码，可以使用 set MINIO\_ACCESS\_KEY=minioadmin 命令和 set MINIO\_SECRET\_KEY=minioadmin 来更改用户名和密码。

在浏览器里打开 127.0.0.1:9000，界面如图所示

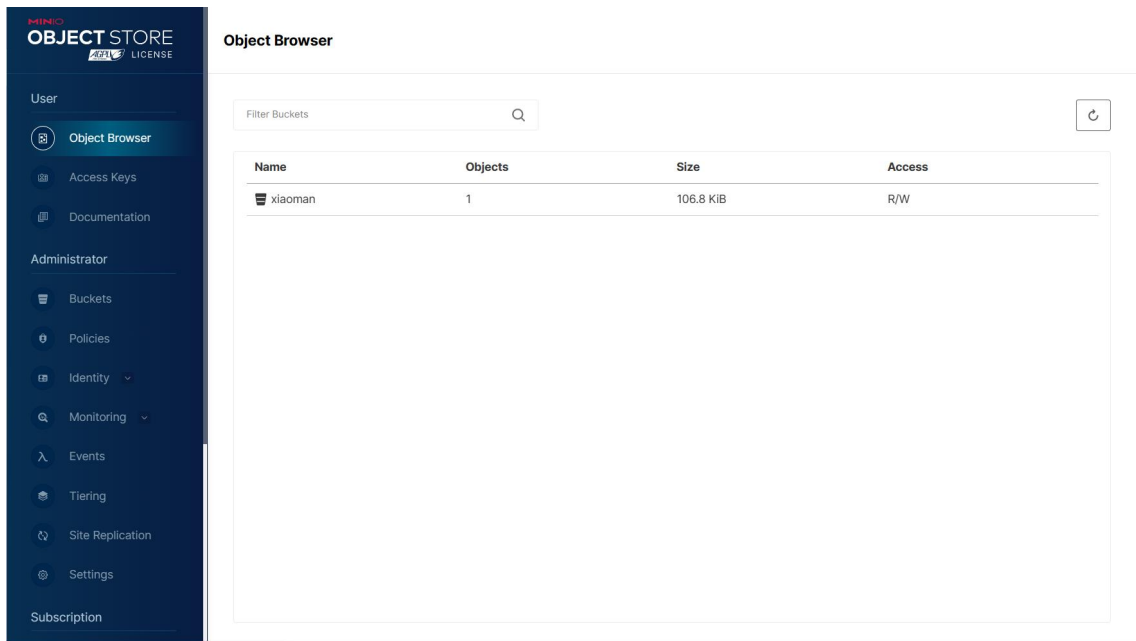


图 5.2

图中已经存在了之前创建的 bucket，在左侧 administrator 栏下点击 buckets 框后，点击右上角的 create bucket 即可创建新的 bucket。如图：

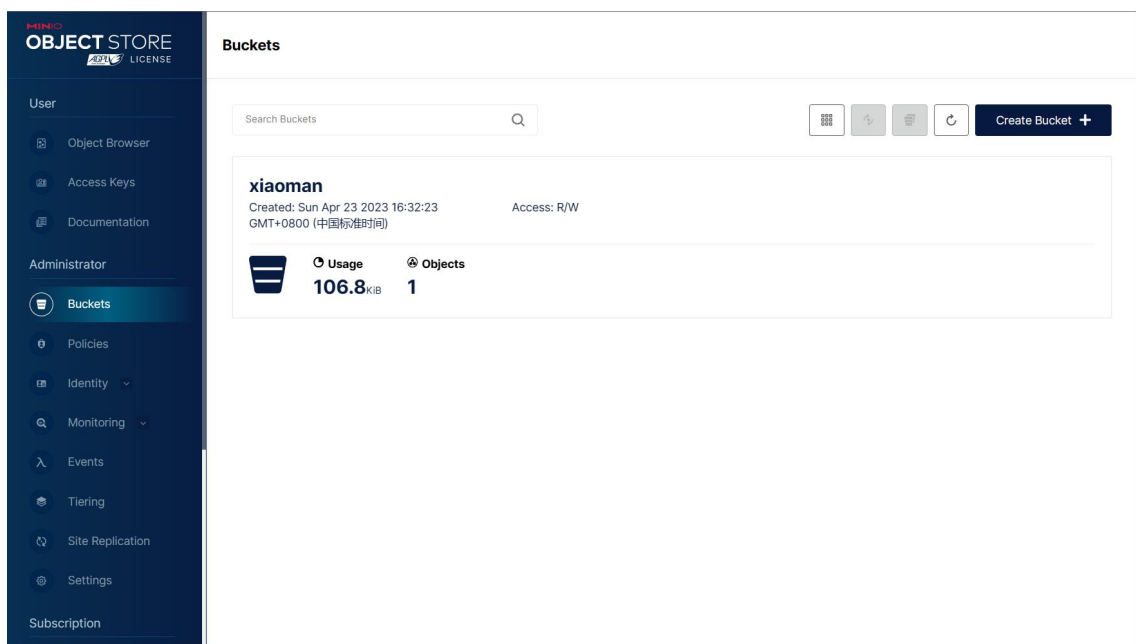


图 5.3

那之后，上传一个文件到 bucket 中用于后续的测试。

3. 在 Windows 系统上下载 mc.exe，进行客户端的相关操作。首先按照上述步骤启动 minio 服务器，再打开一个新的命令提示符 cmd，输入 `mc config host add [name] http://127.0.0.1:9000 [rootuser] [rootpass] --api s3v4`，用于创建名字为[name]的存储服务，通过 `mc mb [name]/[bucketname]` 创建新的名字为[bucketname]的 bucket。使用 `mc rb [name]/[bucketname]` 来删除这个 bucket。

4. 利用 s3bench 脚本对对象的存储性能进行相应的分析。首先下载老师提供的相关测试工具包，之后修改 run-s3bench.cmd 这个文件。把 accesskey 改为自己的账号名，accessSecret 改为学号，bucket 改为相应的 bucketname（）。之后运行脚本就可以进行相应的性能测试操作。

```
s3bench.exe ^
-accessKey=ZXY ^
-accessSecret=U202015481 ^
-bucket=xiaoman ^
-endpoint=http://127.0.0.1:9000 ^
-numClients=24 ^
-numSamples=256 ^
-objectNamePrefix=loadgen ^
-objectSize=1024
pause
```

图 5.4

那之后，运行脚本，得到的结果如下图所示：

```
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           xiaoman
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       24
numSamples:       256
verbose:          %!d(bool=false)
tracing:          %!d(bool=false)

Results Summary for Write Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  0.79 MB/s
Total Duration:    0.316 s
Number of Errors:  0
-----
Write times Max:    0.081 s
Write times 99th %ile: 0.071 s
Write times 90th %ile: 0.041 s
Write times 75th %ile: 0.035 s
Write times 50th %ile: 0.027 s
Write times 25th %ile: 0.019 s
Write times Min:    0.008 s

Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  1.47 MB/s
Total Duration:    0.170 s
Number of Errors:  0
-----
Read times Max:     0.058 s
Read times 99th %ile: 0.035 s
Read times 90th %ile: 0.025 s
Read times 75th %ile: 0.017 s
Read times 50th %ile: 0.009 s
Read times 25th %ile: 0.005 s
Read times Min:     0.001 s

Cleaning up 256 objects...
Deleting a batch of 256 objects in range {0, 255}... Succeeded
Successfully deleted 256/256 objects in 319.5885ms
```

图 5.5

发现读取速率远大于写入速率。我们将 objectSize、numClients 和 numSamples 分别修改，得到以下表格：

Object size	Num clients	Num samples	写速度	写时间	读速度	读时间
0.001MB	24	256	1.15MB/s	0.218s	2.72MB/s	0.092s
0.002MB	24	256	2.20MB/s	0.227s	4.17MB/s	0.120s
0.003MB	24	256	2.95MB/s	0.254s	6.57MB/s	0.114s
0.001MB	16	256	1.09MB/s	0.230s	2.12MB/s	0.118s
0.001MB	8	256	1.08MB/s	0.232s	2.33MB/s	0.107s
0.001MB	4	256	1.10MB/s	0.228s	2.79MB/s	0.089s
0.001MB	24	128	0.89MB/s	0.141s	1.99MB/s	0.063s
0.001MB	24	64	0.64MB/s	0.097s	2.04MB/s	0.031s

在实验过程中，可以看到：

- (1) 写入读写未曾出错
- (2) 读取速率大于写入速率
- (3) 随着 object size 增大，读写速率也会增大
- (4) 随着 client 增大，速率先减后增。但理论上应该先升高后降低，比较疑惑。
- (5) 随着 sample 增大，逐渐增大。但理论上应该先降低后升高，未观测到最低水平。

## 六、实验总结

在实验过程中，参考了老师和同学们提供的资料，了解了对象存储技术的基本概念和优缺点和应用场景。熟悉了 minio、mc、s3bench 的使用方法，顺利进行了小范围内的测试，虽然数据量很小，但进行了较基本的观察。

感谢老师和同学的帮助和指导，我才能明了实验的基本过程，解决了实验过程中的问题，最终完成实验。

## 参考文献

- [1] 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.
- [2] ARNOLD J. OpenStack Swift[M]. O' Reilly Media, 2014.
- [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.
- [4] Dean J, Barroso L A. Association for Computing Machinery, 2013. The Tail at Scale[J]. Commun. ACM, 2013, 56(2): 74 - 80.
- [5] Delimitrou C, Kozyrakis C. Association for Computing Machinery, 2018. Amdahl's Law for Tail Latency[J]. Commun. ACM, 2018, 61(8): 65 - 72.
- [6] [https://blog.csdn.net/LG\\_15011399296/article/details/124859515](https://blog.csdn.net/LG_15011399296/article/details/124859515)