2019 级

《物联网数据存储与管理》课程

实 验 报 告

姓　名 马世拓

学　号 U201914900

班　号 CS1901 班

日　期 2022.04.19

目　　录

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，是用来描述解决问题和处理离散单元的方法的通用术语，这些离散单元被称作对象。而对象存储系统，提供了高可靠、跨平台以及安全的数据共享的存储体系结构。

目前已经有了大量的基于块和基于文件的存储系统可供选择，基于块的存储系统，磁盘块通过底层存储协议访问，所有高级别的任务，像共享、锁定和安全通常由操作系统负责，即基于块的存储系统关心所有的底层的问题。而文件存储以文件为传输协议，以 TCP/IP 实现网络化存储，可扩展性好、价格便宜、用户易管理。但缺点在于读写速率低，传输速率慢。而对象存储，克服块存储与文件存储各自的缺点，发扬它俩各自的优点。块存储读写快，不利于共享，文件存储读写慢，利于共享。这就是我们在已有基于块和基于文件的存储系统的情况下，还需要对象存储的原因。

在我们的实验中，使用到了 Minio 作为服务端。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，而一个对象文件从数 KB 到 5TB 都能够得到很好的支持。

三、实验环境

本实验的环境如下：
实验所用的操作系统为 CentOS 版 64 位虚拟机环境，如图 3-1.

图 3-2 实验环境

软件环境：

对象存储客户端：采用 Minio 进行测试
对象存储服务器端：采用 mc
对象存储测试工具：采用 s3-bench 进行测试

四、实验内容

本次实验是对象存储实验入门实践，准备工作有 Git 与 GitHub 的学习，linux 虚拟机的安装以及 Java 或 Python 环境的准备。然后是选定对象存储服务端与客户端，选择 Minio 和 MC，在 linux 中运行 Minio 后用测试工具 Cosbench 进行测试。

4.1 对象存储技术实践

1. 采用 Minio 作为服务端

1）下载 Minio 作为服务端。首先在 Minio 官网 https://www.minio.io/downloads.html 下载 Minio 和客户端 MC，然后使用 chmod +x minio 命令行添加权限。

2）运行 Minio。在 linux 打开终端，在 root 权限下运行：./minio server /data。如图 4-1，可以看到服务器已经打开，并且可以通过端口 9000 访问。此时可以看到用户名和密码。

```
[mm@localhost ~]$ sudo MINIO_ROOT_USER=admin MINIO_ROOT_PASSWORD=password ./minio server /mnt/data --console-address ":9001"
[sudo] mm 的密码：
API: http://192.168.253.131:9000  http://127.0.0.1:9000
RootUser: admin
RootPass: password

Console: http://192.168.253.131:9001 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://192.168.253.131:9000 admin password

Documentation: https://docs.min.io
Finished loading IAM sub-system (took 0.0s of 0.1s to load data).

You are running an older version of MinIO released 2 weeks ago
Update: Run `mc admin update`
```

图 4-1 运行 Minio

3）在浏览器访问服务器。在浏览器中输入 http://127.0.0.1:9000 可以访问服务器，登录界面如图 4-2。确认登录后，可以看到界面如图 4-3。

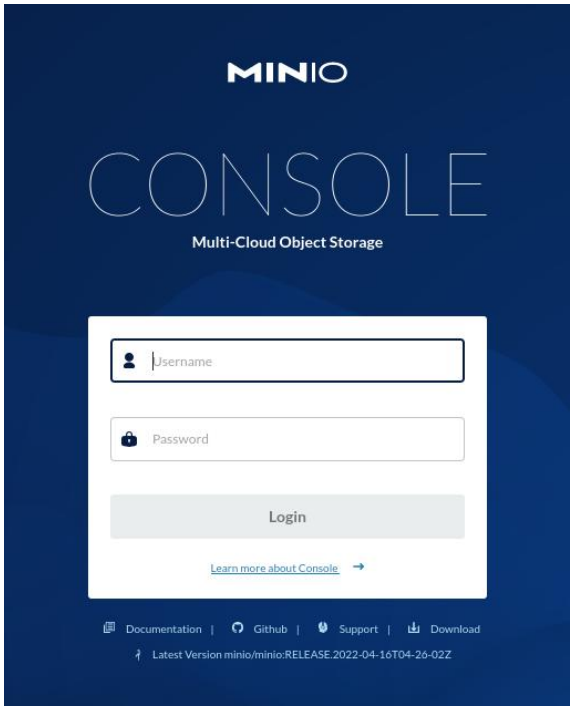

图 4-2 登录服务器

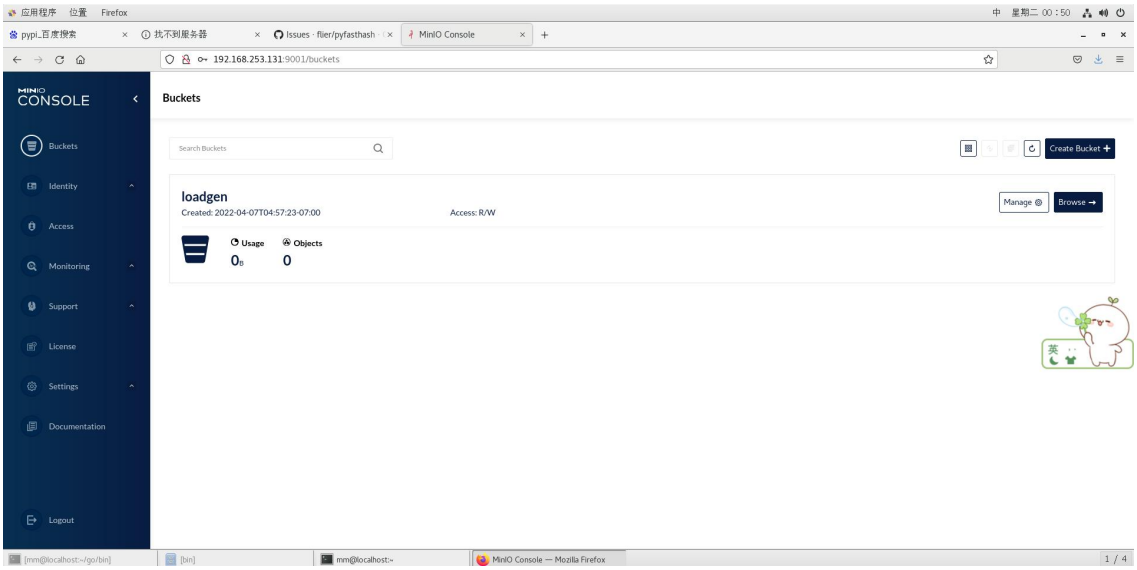

图 4-3 在浏览器中访问服务器

4）在浏览器中可以添加存储对象。点击页面的+号按钮，可以选择新建一个仓库或者上传一个新的存储文件。

5）下载 s3bench 源码和相关项目，GO 语言源码如下：

```
func main() {
	endpoint := flag.String("endpoint", "", "S3 endpoint(s) comma separated - http://IP:PORT,http://IP:PORT")
	region := flag.String("region", "igneous-test", "AWS region to use, eg: us-west-1|us-east-1, etc")
	accessKey := flag.String("accessKey", "", "the S3 access key")
	accessSecret := flag.String("accessSecret", "", "the S3 access secret")
	bucketName := flag.String("bucket", "bucketname", "the bucket for which to run the test")
	objectNamePrefix := flag.String("objectNamePrefix", "loadgen_test_", "prefix of the object name that will be used")
	objectSize := flag.Int64("objectSize", 80*1024*1024, "size of individual requests in bytes (must be smaller than main memory)")
	numClients := flag.Int("numClients", 40, "number of concurrent clients")
	numSamples := flag.Int("numSamples", 200, "total number of requests to send")
	skipCleanup := flag.Bool("skipCleanup", false, "skip deleting objects created by this tool at the end of the run")
	verbose := flag.Bool("verbose", false, "print verbose per thread status")

	flag.Parse()

	if *numClients > *numSamples || *numSamples < 1 {
		fmt.Printf("numClients(%d) needs to be less than numSamples(%d) and greater than 0\n", *numClients, *numSamples)
		os.Exit(1)
	}

	if *endpoint == "" {
		fmt.Println("You need to specify endpoint(s)")
		flag.PrintDefaults()
		os.Exit(1)
	}

	// Setup and print summary of the accepted parameters
	params := Params{
		requests:       make(chan Req),
		responses:      make(chan Resp),
		numSamples:     *numSamples,
		numClients:     uint(*numClients),
		objectSize:     *objectSize,
		objectNamePrefix: *objectNamePrefix,
		bucketName:     *bucketName,
		endpoints:      strings.Split(*endpoint, ","),
		verbose:        *verbose,
	}
	fmt.Println(params)
	fmt.Println()

	// Generate the data from which we will do the writting
	fmt.Printf("Generating in-memory sample data... ")
	timeGenData := time.Now()
```

图 4-4 s3bench 源码

经过编译可以得到 s3bench 测试程序

6）运行 MC 客户端进行对服务器的访问。重新打开一个终端，输入命令行：mc alias set myminio/ http://MINIO-SERVER MYUSER MYPASSWORD，使得可以通过 MC 访问服务器，如图 4-5。

图 4-5 通过 MC 访问服务器

6）使用测试工具 s3-bench 进行测试。运行脚本启动驱动程序和控制器，如图 4-6。

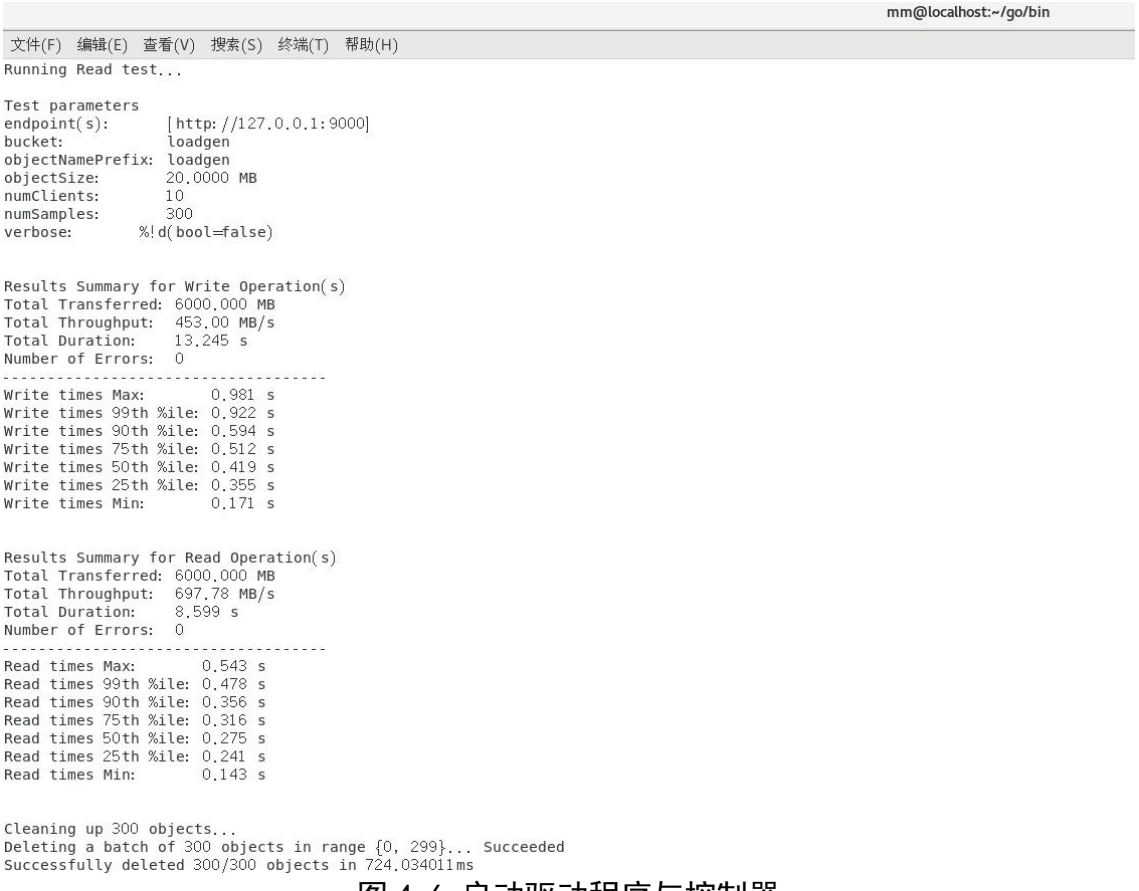

图 4-6 启动驱动程序与控制器

4.2 对象存储性能分析

1.采用 Minio 作为服务端

首先，测试标准的 s3-bench 结果如表 4-1。可以看到我们改变 object size 等参数时运行时间的变化。

Object size	Num clients	Num samples	写速度	写时间	读速度	读时间
0.001MB	10	100	0.91 MB/s	0.108 s	1.59 MB/s	0.062 s
20MB	10	100	128.40 MB/s	0.152 s	233.52 MB/s	0.084 s
200MB	10	100	392.80 MB/s	50.916 s	659.84 MB/s	30.310 s
20MB	20	100	471.09 MB/s	4.245 s	613.13 MB/s	3.262 s
20MB	30	100	453.42 MB/s	4.411 s	561.41 MB/s	3.562 s
20MB	10	200	420.58 MB/s	9.511 s	736.45 MB/s	5.431 s
20MB	10	300	453.00 MB/s	13.245 s	697.78 MB/s	8.599 s

表 4-1 s3bench 测试结果（minio 作为服务端）

我们可以看到：

1） 写入和读取的成功率一直都是 100%；

2） 读取的 Bandwidth 比写入的 Bandwidth 大，这和我们平时了解的读取速度大于写入速度是一致的；

3） 随着每次读取与写入 size 的增大，Bandwidth 渐渐增大，而 Throughput 渐渐减小，相应的平均的休息时间与工作时间也减小。

4） 随着 object size 增大，读写速度也会逐渐增大

5） 在 object size 相等的情况下，若 client 数量更多则读写速率先升高后降低，即存在一个最优水平

6） 在 object size 相等的情况下，若 sample 数量更多则读写速率先降低后升高，即存在一个最低水平

五、实验总结

此次试验是面向对象存储的入门实验，在实验中我了解了对象存储技术，明白了在已有基于块和基于文件的存储系统的情况下，我们仍然需要面向对象存储系统的原因。

实验中，我在 linux 环境中使用了 minio 作为服务端，使用测试工具 s3-bench 进行了测试，受限于实验条件，进行操作的容量都比较小，在实际应用中肯定会有更大的存储吞吐量，不过总体来看速度和成功率都十分可观，而且通过简单的 minio 搭建便可以达成类似网盘的效果，是一个不小的惊喜。

实验总体而言更偏向于熟悉了解这门技术，难度主要在于实验环境的配置，过程中碰到了不少问题，一开始我因为系统版本和多数人不一致而导致 go 语言的安装包怎么都弄不好，后来查了很多资料改了下配置，问题马上就解决了，还有 s3-bench 的权限问题，一开始忘记分配权限，出现一大堆奇怪的错误，后来再 root 下调用脚本，结果因为之前失败的进程没有杀死造成失败，饶了一大圈才解决问题。更让我体会到了 linux 系统下权限的重要性。总体而言获益匪浅，希望以后还能有这样的实验！

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.