2018 级

《物联网数据存储与管理》课程

实 验 报 告

姓　名　杨清帆

学　号　U201814758

班　号　物联网 1801 班

日　期　2021.06.21

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

随着软硬件和计算机的高速发展，数据存储的几个重要的需求逐渐为高容量，海量，且可扩展；高性能，快速访问、定位；高可用，不怕个别磁盘失效甚至整个中心被毁；低成本：TCO=D(部署)+M(维护)*t(时间)。

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。

本次实验用到的软件包括：minio, MinioClient, S3Proxy, S3cmd, S3benchmark。

minio 是 Apache License v2.0 下发布的对象存储服务器。它与 Amazon S3 云存储服务兼容。它最适合存储非结构化数据，如照片，视频，日志文件，备份和容器/ VM 映像。对象的大小可以从几 KB 到最大 5TB Minio 服务器足够轻，可以与应用程序堆栈捆绑在一起，类似于 NodeJS，Redis 和 MySQL。Minio Cloud Storage 是一款轻量级对的象存储服务，兼容 Amazon S3。通过简单的命令，就可以搭建服务器，能实现可以通过浏览器访问的简易网盘功能。利用 Minio 客户端 MC 可以在命令行中对云中的内容进行管理。

S3Proxy 实现 S3 的 API，还可以实现从 S3 到其他多种云服务的翻译，可以通过使用本地文件系统进行无 Amazon 测试，可嵌入 Java 应用程序等功能。

S3cmd 是一款免费的命令行工具和客户端，用于在 Amazon S3 和其他使用 S3 协议的云存储服务提供商中上传，检索和管理数据。

s3-benchmark 是 Wasabi 为对象执行 S3 操作（PUT，GET 和 DELETE）而提供的性能测试工具。除了桶配置之外，还可以针对不同的测试给出对象大小和线程数。该测试工具基于用于测试性能的 Nasuni 性能基准测试方法。不同的云存储提供商。

三、实验环境

实验环境如下表所示：

Minio/mc 实验及 S3proxy/s3cmd 实验		
硬件环境	操作系统	Windows 10 下 VM 虚拟环境 Ubuntu 18.04
	处理器	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz　1.80 GHz
软件环境	客户端	采用 MC 进行测试
	服务器端	Minio
	测试工具	Cosbench

四、实验内容

本次实验的主要内容如下：

1. 对 git 进行了解和熟悉。
2. 对对象存储系统进行实践，采用目前几种典型的服务，如 minion/mc，s3proy/s3cmd 配置服务器端和客户端，并进行简单的创建或删除 bucket、上传或删除文件等操作。
3. 对对象存储系统进行测试，采用 COSBench 进行负载测试。
4. 对 COSBench 结果进行一定的性能分析。

4.1 对象存储技术实践

1. 在 windows 系统下配置 minio server 端，通过浏览器登陆 127.0.0.1 查看效果，进行简单的创建 bucket，上传文件，删除文件和 bucket 操作。
2. 在 windows 下配置 minio 客户端 mc，在命令行下输入命令实现创建 bucket，上传文件，删除文件和 bucket 操作。
3. 在 windows 下配置 COSBench，提交负载样例到 COSBench，观察运行和结束后的各种状态指标。

4.2 对象存储性能分析

1. 读写性能对比。
2. 测试块大小对运行结果的影响：将负载样例中的每项 workers 数统一改为 4，块大小从 8k 到 1M 逐渐翻倍增大，观察运行结果。
3. 测试 workers 值对运行结果的影响：将负载样例的每项大小都改为 64k，将 workers 值从 1 逐渐翻倍增大到 128，观察运行结果。

五、实验过程

1. 安装配置 minio sever 端
   1) 从 minio 官网：https://min.io/ 下载 minio.exe 文件，启动一个 cmd 窗口，进入 minio.exe 所在文件夹，输入命令：.\minio.exe server D:\html\minio，后面路径是图片上传之后的存储目录，运行成功之后，会看到下图 5.1 所示的界面。

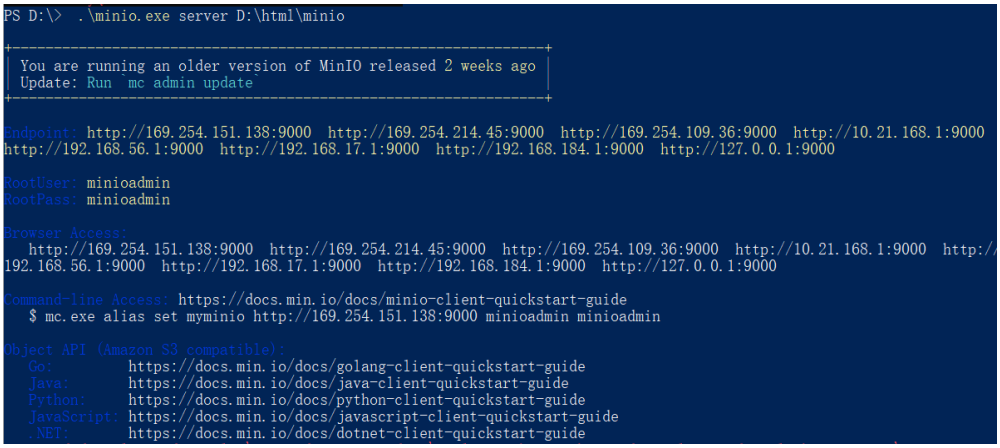

图 5.1 打开 minio server

2) 浏览器打开其中的一个 endpiont 127.0.0.1:9000,用所给的 accesskey 和 secretkey 登录，访问结果如图 5.2 所示。

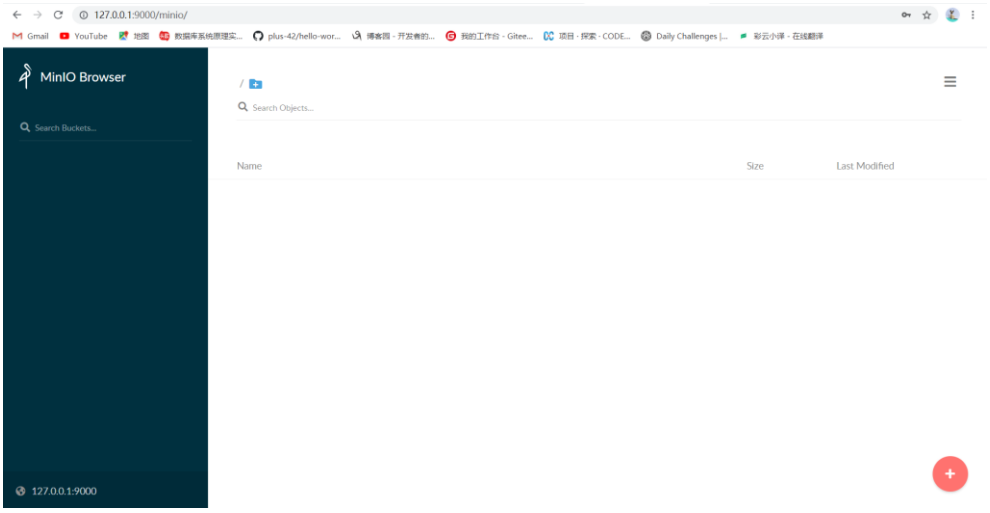

图 5.2 web 端图形化客户端

在 minio 中新建文件夹，如图中的最先的 u201814758，文件夹开头字母不能是大写字母。

2. 安装配置 minio 客户端

1) 首先在 minio 官网上下载 mc.exe 文件，下载的目录下运行此程序进行 client 端的配置，使其连接上客户端，并进行一个简单的 bucket 创建操作来检测其性能，如下图 5.3 所示：

```
PS D:\> ./mc.exe config host add minio http://127.0.0.1:9000 minioadmin minioadmin
Added  minio  successfully.
PS D:\> ./mc.exe mb newbucket
Bucket created successfully `newbucket`.
PS D:\> ./mc.exe ls
[2021-06-08 21:42:31 CST]     0B $RECYCLE.BIN\
[2021-05-14 16:48:41 CST]     0B Android Studio工程文件\
[2021-06-14 14:49:03 CST]     0B BaiduNetdiskDownload\
[2021-06-18 09:01:24 CST] 8.0KiB DumpStack.log.tmp
[2020-05-08 11:13:02 CST]     0B Program Files\
[2020-06-19 10:03:05 CST]     0B ProgramData\
[2019-10-25 18:21:22 CST]     0B Recovery\
[2020-05-15 19:03:30 CST] 7.0KiB Solution1.sqlsuo
[2020-06-19 10:16:57 CST]     0B System Volume Information\
[2021-06-07 20:41:18 CST]     0B VM\
[2021-06-20 09:25:48 CST]     0B applation\
[2021-05-13 08:24:48 CST]     0B book\
[2021-06-20 21:02:43 CST]     0B html\
[2021-06-20 08:43:54 CST]     0B iot-storage-experiment-master\
[2021-05-14 20:55:53 CST]     0B java\
[2021-04-02 20:57:23 CST] 155MiB jdk-8u5-windows-x64.exe
[2021-06-07 20:21:09 CST]  21MiB mc.exe
[2021-06-07 20:21:17 CST]  58MiB minio.exe
[2021-06-20 21:48:31 CST]     0B newbucket\
```

图 5.3 客户端操作测试

3. 在 github 上下载 COSBench 压缩包（0.4.2.c4 版本），解压，运行 start-all.bat，结果如图 1.3 所示。可以看到该批处理文件打开了 driver 和 controller 两个窗口，分别在端口 18089 和 19089 监听。如下图 5.4 所示。

图 5.4 运行 start-all.bat

4. 使用测试工具 cosbench 进行测试

用浏览器访问 127.0.0.1:19088/controller/index.html 进入 cosbench 的测试界面。

将 workload-example.xml 提交给 cosbench 进行测试（更新测试样例中的 endpoint、accesskey、secretkey 值。）测试结果如下图 5.5 和 5.6 所示。

```
<storage type="s3" config="accesskey=mumuzyl;secretkey=
12345654321;endpoint=http://127.0.0.1:9000/" />
```

op1: read	3.6 kops	28.79 MB	4.66 ms	4.59 ms	120.41 op/s	963.25 KB/S	100%
op2: write	834 ops	6.67 MB	266.3 ms	266.26 ms	27.9 op/s	223.21 KB/S	100%
op1: read	2.98 kops	47.62 MB	5.12 ms	4.99 ms	100.36 op/s	1.61 MB/S	100%
op2: write	741 ops	11.86 MB	299.39 ms	299.21 ms	24.99 op/s	399.85 KB/S	100%
op1: read	2.57 kops	82.3 MB	4.06 ms	3.85 ms	86.11 op/s	2.76 MB/S	100%
op2: write	660 ops	21.12 MB	165.07 ms	164.42 ms	22.09 op/s	706.99 KB/S	100%
op1: read	2.97 kops	190.02 MB	4.26 ms	3.73 ms	99.29 op/s	6.35 MB/S	100%
op2: write	734 ops	46.98 MB	145.57 ms	144.17 ms	24.55 op/s	1.57 MB/S	100%
op1: read	1.36 kops	173.57 MB	6.45 ms	4.95 ms	45.2 op/s	5.79 MB/S	100%
op2: write	338 ops	43.26 MB	62.73 ms	58.17 ms	11.27 op/s	1.44 MB/S	100%
op1: read	1.15 kops	293.38 MB	7.88 ms	4.9 ms	38.22 op/s	9.78 MB/S	100%
op2: write	300 ops	76.8 MB	69.64 ms	62.07 ms	10.01 op/s	2.56 MB/S	100%
op1: read	1.03 kops	527.87 MB	9.25 ms	4.59 ms	34.38 op/s	17.6 MB/S	100%
op2: write	254 ops	130.05 MB	80.31 ms	68.43 ms	8.47 op/s	4.34 MB/S	100%
op1: read	713 ops	713 MB	13.89 ms	5.17 ms	23.8 op/s	23.8 MB/S	100%
op2: write	201 ops	201 MB	99.65 ms	80.56 ms	6.71 op/s	6.71 MB/S	100%
op1: cleanup -	128 ops	0 B	4.91 ms	4.91 ms	203.5 op/s	0 B/S	100%

图 5.5 测试结果

ID	Name	Works	Workers	Op-Info	State	Link
w5-s1-init	init	1 wks	1 wkrs	init	completed	view details
w5-s2-prepare	prepare	8 wks	64 wkrs	prepare	completed	view details
w5-s3-8kb	8kb	1 wks	8 wkrs	read, write	completed	view details
w5-s4-16kb	16kb	1 wks	8 wkrs	read, write	completed	view details
w5-s5-32kb	32kb	1 wks	4 wkrs	read, write	completed	view details
w5-s6-64kb	64kb	1 wks	4 wkrs	read, write	completed	view details
w5-s7-128kb	128kb	1 wks	1 wkrs	read, write	completed	view details
w5-s8-256kb	256kb	1 wks	1 wkrs	read, write	completed	view details
w5-s9-512kb	512kb	1 wks	1 wkrs	read, write	completed	view details
w5-s10-1mb	1mb	1 wks	1 wkrs	read, write	completed	view details
w5-s11-cleanup	cleanup	1 wks	1 wkrs	cleanup	completed	view details
w5-s12-dispose	dispose	1 wks	1 wkrs	dispose	completed	view details

图 5.6 测试结果

5. 测试结果分析

w5-s5-32kb	32kb	1 wks	4 wkrs	read, write	completed	view details
w5-s6-64kb	64kb	1 wks	4 wkrs	read, write	completed	view details

32kb 1wks 4wkrs

General Report

Op-Type	Op-Count	Byte-Count	Avg-ResTime	Avg-ProcTime	Throughput	Bandwidth	Succ-Ratio
op1: read	2.57 kops	82.3 MB	4.06 ms	3.85 ms	86.11 op/s	2.76 MB/S	100%
op2: write	660 ops	21.12 MB	165.07 ms	164.42 ms	22.09 op/s	706.99 KB/S	100%

64kb 1wks 4wkrs

Op-Type	Op-Count	Byte-Count	Avg-ResTime	Avg-ProcTime	Throughput	Bandwidth	Succ-Ratio
op1: read	2.97 kops	190.02 MB	4.26 ms	3.73 ms	99.29 op/s	6.35 MB/S	100%
op2: write	734 ops	46.98 MB	145.57 ms	144.17 ms	24.55 op/s	1.57 MB/S	100%

其他条件相同时 bandwidth 与字节数成正比。

32kb 1wks 8wkrs

General Report

Op-Type	Op-Count	Byte-Count	Avg-ResTime	Avg-ProcTime	Throughput	Bandwidth	Succ-Ratio
op1: read	2.97 kops	95.01 MB	4.51 ms	4.28 ms	99.67 op/s	3.19 MB/S	100%
op2: write	761 ops	24.35 MB	295.4 ms	294.75 ms	25.54 op/s	817.28 KB/S	100%

64kb 1wks 8wkrs

Op-Type	Op-Count	Byte-Count	Avg-ResTime	Avg-ProcTime	Throughput	Bandwidth	Succ-Ratio
op1: read	3.07 kops	196.42 MB	5.03 ms	4.35 ms	103.18 op/s	6.6 MB/S	100%
op2: write	741 ops	47.42 MB	300.13 ms	298.52 ms	24.91 op/s	1.59 MB/S	100%

随着 workers 的增多，吞吐率、带宽、平均时间都有着相应的增加，但是没有发现

相对应的关系.

六、实验总结

这次实验的难度并不大，主要问题在于环境的配置和对象存储系统的搭建等方面。在 Linux 上尝试搭建 s3proxy 和 S3 Bench 时，遇到了很多问题，最终也没能解决，最后决定直接在 windows 上使用 minio 服务器和客户端。

环境的配置有之前学长学姐的经验，分析的过程使用控制变量的方法，对比分析 bit 和 worker 的变化对存取结果和带宽的影响。随着 workers 数目的增加，带宽增加，并发性提高带宽，数据吞吐率会减小，所需要的时间增多。

总的来说，这次实验学到了很多有用的新知识，比如 github 的各种操作、对象存储系统的服务等。学会了在本地开启一个 server，然后通过本地的客户端或者终端，来访问这个服务器，上传或者下载文件。

最后也要感谢老师对我们的悉心教导，每节课上课前都嘱咐我们一些需要注意的事项。真的帮助我们很多，给我们在一头露水时点明迷津，感谢老师为我们提供的帮助。

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.
（可以根据实际需要更新调整）