2020 级

《物联网数据存储与管理》课程

实 验 报 告

姓　名　范亚锋

学　号　U201814580

班　号　物联网 2001 班

日　期　2023.5.16

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

现如今传统的文件系统和数据库 Web 后端不再足以满足要求，只好为管理非结构化数据的存储系统让路。不像文件系统中的文件，对象存储在扁平结构中。只有对象池：没有文件夹，没有目录，也没有层次体系。你只要提供对象 ID，就可以请求某个对象。对象可以是本地的，也可以是远在千里之外的云服务器上，但由于它们是在扁平的地址空间，检索方式一模一样。一个重要的方面是元数据处理。对象存储提供了极高的灵活性，因为对象元数据是任意的。

元数据并不仅限于存储系统认为很重要的对象（想一想文件系统中的固定元数据）。你可以手动添加任何类型或任何数量的元数据。比如说，你可以指定与对象关联起来的应用程序的类型，指定应用程序的重要性，指定赋予对象的数据保护级别，指定是不是想把该对象复制到另一个站点或多个站点，指定何时删除该对象，不一而足。

鉴于互联网上的大部分数据是非结构化数据，加上专家们预测非结构化数据以两位数的幅度增长，因而迎面克服这个挑战很重要。非结构化数据必须以易于访问的方式来存储，兴起的对象存储技术综合了原网络存储技术的高速直接访问和数据共享等优势，同时也提供了具有高性能、高可靠性、跨平台以及安全的数据共享的存储体系结构。

三、实验环境

操作系统规格如图 1：

图 1 操作系统规格

Java 版本如图 2：

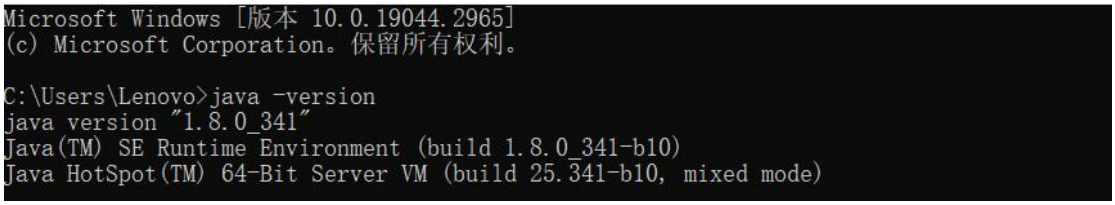

图 2 Java 版本

Python 版本如图 3：

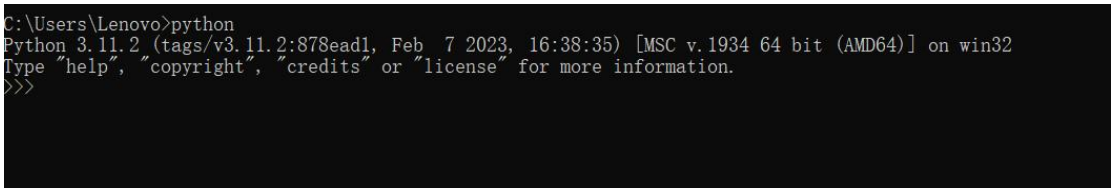

图 3 python 版本

四、实验内容

本实验在 windows 环境下完成相应的环境配置，基础的实验环境安装完毕后部署对象存储的服务端、客户端和测试工具，然后启动 cosbench 测试程序，提交所给的测试文件，观察负载测试的结果，分析吞吐率、延迟等技术指标。

4.1 对象存储技术实践

1.在 windows 下配置 java 环境。

2.在官网上下载安装对象存储服务端 minio 和客户端 minio client

3. 安装后使用浏览器访问 http://127.0.0.1:9000，如果可以访问说明安装成功，否则重新安装

4.下载测试程序 COSBench，配置完成后提交负载测试样例，观察运行结果。

4.2 对象存储性能分析

1.读写性能对比

2.分析块的大小对运行结果的影响

将负载样例中 8kb~1mb work 的 workers 数量改为一致，分析块的大小对存储性能的影响。

3.分析 workers 数量对运行结果的影响

将负载样例中块的大小改为一致，修改 workers 的数量，分析 workers 数量对存储性能的影响。

五、实验过程

Minio：

1.安装 minio 服务器：

从以下 URL 下载 MinIO 可执行文件：

https://dl.min.io/server/minio/release/windows-amd64/minio.exe

下一步包括运行可执行文件的说明。 不能从资源管理器或通过双击文件来运行可执行文件，可以调用可执行文件来启动服务器。

2.启动

将文件的路径添加到系统。使用以下命令在文件夹中启动本地 MinIO 实例。

.\minio.exe server C:\minio --console-address :9090

该过程将其输出打印到系统控制台，如图 4 所示：

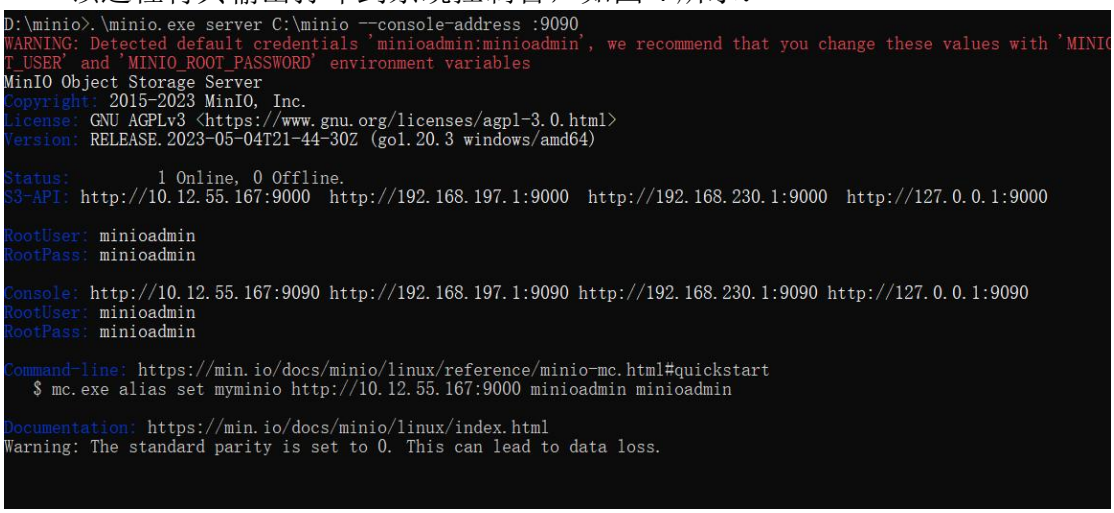

图 4 server

3. 将浏览器连接到 minio 服务器

通过浏览器访问图 4 中显示的地址，然后输入账户密码登录，默认账户密码都是 minioadmin，登录后的界面如图 5 所示：

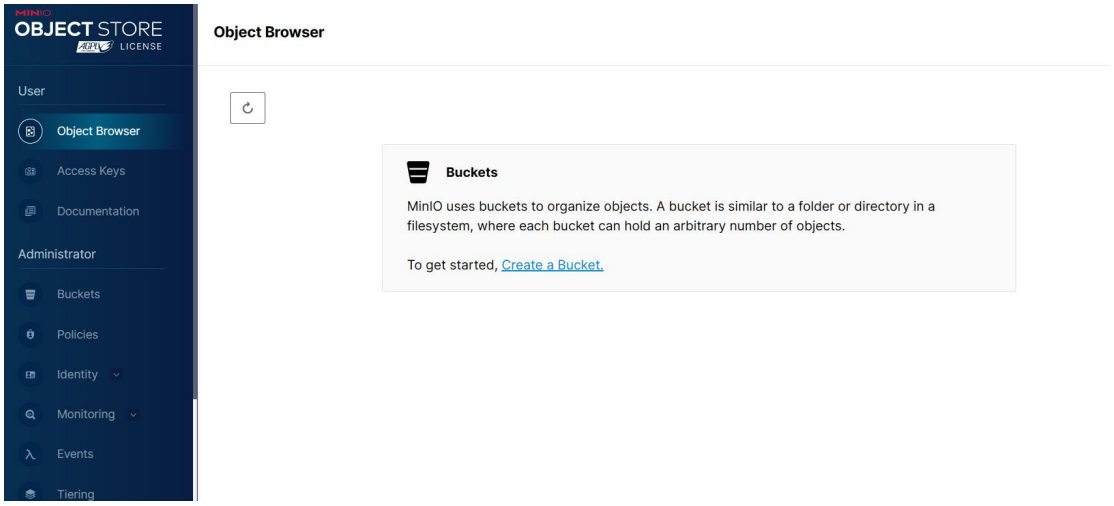

图 5 minio 控制台

4. 安装 minio 客户端

从以下链接下载适用于 Windows 的独立 MinIO 服务器：
https://dl.min.io/client/mc/release/windows-amd64/mc.exe

运行该文件，如图 6 所示：

```
D:\minio>mc.exe
                                                                                                                    (q)uit/es
NAME:
  mc - MinIO Client for object storage and filesystems.

USAGE:
  mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]
                                                                                                                    (q)uit/es
C
  update      update mc to latest release
  ready       checks if the cluster is ready or not
  ping        perform liveness check
  od          measure single stream upload and download
  batch       manage batch jobs

GLOBAL FLAGS:
  --autocompletion              install auto-completion for your shell
  --config-dir value, -C value  path to configuration folder (default: "C:\\Users\\Lenovo\\mc")
  --quiet, -q                   disable progress bar display
  --no-color                    disable color theme
  --json                        enable JSON lines formatted output
  --debug                       enable debug output
  --insecure                    disable SSL certificate verification
  --limit-upload value          limits uploads to a maximum rate in KiB/s, MiB/s, GiB/s. (default: unlimited)
  --limit-download value        limits downloads to a maximum rate in KiB/s, MiB/s, GiB/s. (default: unlimited)
  --help, -h                    show help
  --version, -v                 print the version

TIP:
  Use 'mc --autocompletion' to enable shell autocompletion

COPYRIGHT:
  Copyright (c) 2015-2023 MinIO, Inc.

LICENSE:
  GNU AGPLv3 <https://www.gnu.org/licenses/agpl-3.0.html>

                                                                                                                   100%
```

图 6 运行 mc.exe

5. 在 github 上下载 COSBench 压缩包，解压，完成配置后运行 start-all.bat 文件，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。

6.运行 web.bat，会在浏览器中打开如图 7 所示的页面：

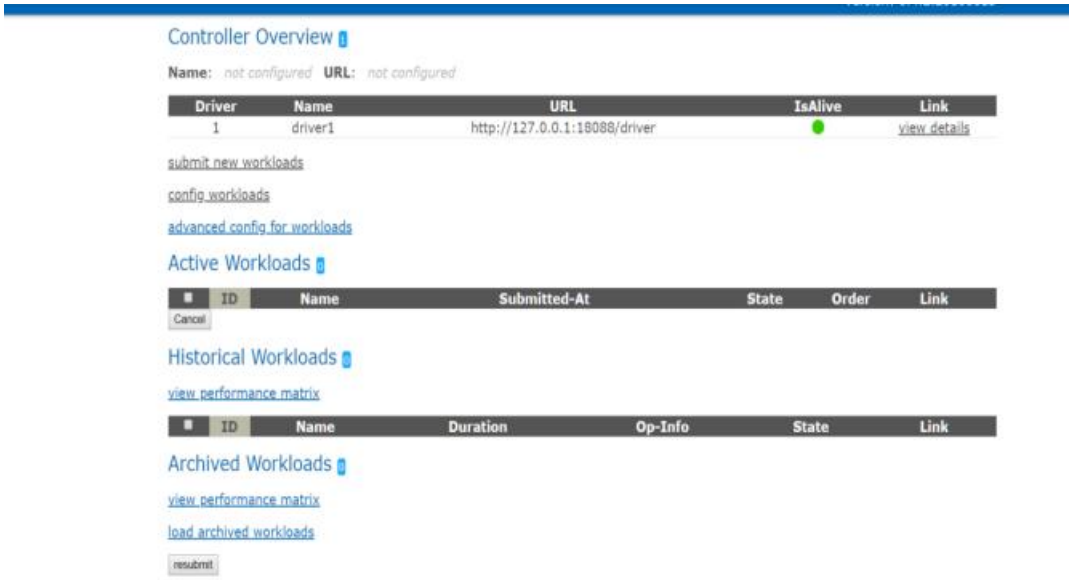

图 7 运行 web.bat

7. 点击 submit new workloads 进入下一界面，选择需要提交的文件，如图 8 所示：

图 8 选择文件

8. 点击 view details 查看刚才提交的负载的详细处理结果，处理结果如图如图 9 和图 10 所示：

Stages

Current Stage	Stages completed	Stages remaining	Start Time	End Time	Time Remaining

ID	Name	Works	Workers	Op-Info	State	Link
w20-s1-init	init	1 wks	1 wkrs	init	completed	view details
w20-s2-prepare	prepare	8 wks	64 wkrs	prepare	completed	view details
w20-s3-8kb	8kb	1 wks	8 wkrs	read, write	completed	view details
w20-s4-16kb	16kb	1 wks	8 wkrs	read, write	completed	view details
w20-s5-32kb	32kb	1 wks	4 wkrs	read, write	completed	view details
w20-s6-64kb	64kb	1 wks	4 wkrs	read, write	completed	view details
w20-s7-128kb	128kb	1 wks	1 wkrs	read, write	completed	view details
w20-s8-256kb	256kb	1 wks	1 wkrs	read, write	completed	view details
w20-s9-512kb	512kb	1 wks	1 wkrs	read, write	completed	view details
w20-s10-1mb	1mb	1 wks	1 wkrs	read, write	completed	view details
w20-s11-cleanup	cleanup	1 wks	1 wkrs	cleanup	completed	view details
w20-s12-dispose	dispose	1 wks	1 wkrs	dispose	completed	view details

There are 12 stages in this workload.

show error statistics details

Performance Graph

Actions

download-log download-config

图 9 处理结果（1）

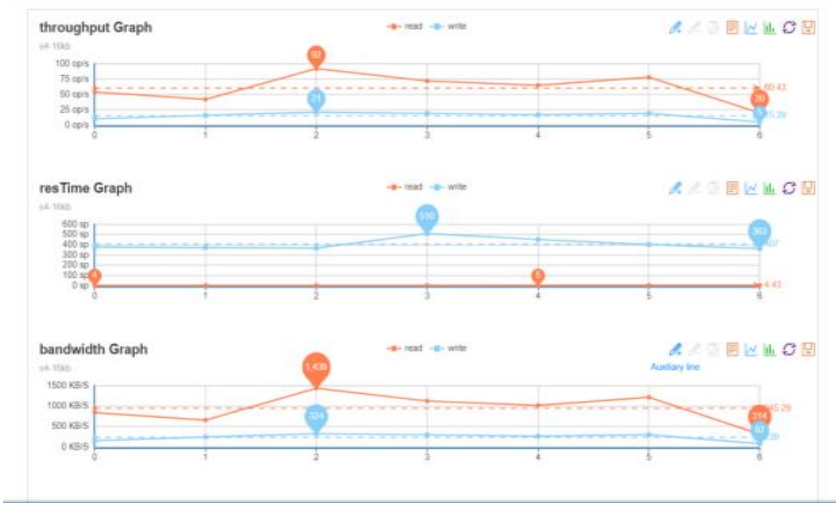

图 10 处理结果（2）

9. 读写性能比较

在 final result 下的 General Report, 中，可以看到 Op-Type、 Op-count、Byte-count 、 Avg-ResTime 、 Avg-ProcTime 、 Throughput 、 Bandwidth 、Succ-Radio 等信息，如图 11。

Op-Type	Op-Count	Byte-Count	Avg-ResTime	Avg-ProcTime	Throughput	Bandwidth	Succ-Ratio
op1: init -write	0 ops	0 B	N/A	N/A	0 op/s	0 B/S	N/A
op1: prepare - write	8 ops	64 KB	4076.62 ms	4076.62 ms	4.63 op/s	37.07 KB/S	100%
op2: prepare - write	8 ops	128 KB	6284.75 ms	6282.88 ms	1.3 op/s	20.84 KB/S	100%
op3: prepare - write	8 ops	256 KB	6630.62 ms	6630.25 ms	1.24 op/s	39.57 KB/S	100%
op4: prepare - write	8 ops	512 KB	5765.25 ms	5762.12 ms	2.68 op/s	171.62 KB/S	100%
op5: prepare - write	8 ops	1.02 MB	5727.5 ms	5372 ms	1.88 op/s	241.16 KB/S	100%
op6: prepare - write	8 ops	2.05 MB	5464.75 ms	2495.38 ms	2.36 op/s	603.4 KB/S	100%
op7: prepare - write	8 ops	4.1 MB	6567.75 ms	3020 ms	1.64 op/s	840.65 KB/S	100%
op8: prepare - write	8 ops	8 MB	4968.62 ms	2244.62 ms	2.51 op/s	2.51 MB/S	100%
op1: read	2.21 kops	17.67 MB	6.72 ms	6.62 ms	74.47 op/s	595.79 KB/S	100%
op2: write	553 ops	4.42 MB	401.82 ms	400.83 ms	18.65 op/s	149.23 KB/S	100%
op1: read	2.2 kops	35.22 MB	4.52 ms	4.19 ms	74.28 op/s	1.19 MB/S	100%
op2: write	542 ops	8.67 MB	418.74 ms	416.2 ms	18.3 op/s	292.86 KB/S	100%
op1: read	2.04 kops	65.12 MB	4.54 ms	3.89 ms	68.37 op/s	2.19 MB/S	100%
op2: write	509 ops	16.29 MB	215.6 ms	210.52 ms	17.1 op/s	547.27 KB/S	100%
op1: read	1.7 kops	108.93 MB	5.99 ms	4.4 ms	56.95 op/s	3.64 MB/S	100%
op2: write	415 ops	26.56 MB	263.37 ms	252.59 ms	13.88 op/s	888.56 KB/S	100%
op1: read	689 ops	88.19 MB	7 ms	4.01 ms	22.97 op/s	2.94 MB/S	100%
op2: write	172 ops	22.02 MB	146.24 ms	123.91 ms	5.73 op/s	734.04 KB/S	100%
op1: read	643 ops	164.61 MB	10.71 ms	5.15 ms	21.51 op/s	5.51 MB/S	100%
op2: write	171 ops	43.78 MB	134.42 ms	99.64 ms	5.72 op/s	1.46 MB/S	100%
op1: read	517 ops	264.7 MB	16.73 ms	4.95 ms	17.25 op/s	8.83 MB/S	100%
op2: write	116 ops	59.39 MB	183.74 ms	116.68 ms	3.87 op/s	1.98 MB/S	100%
op1: read	281 ops	281 MB	28.12 ms	4.01 ms	9.4 op/s	9.4 MB/S	100%
op2: write	70 ops	70 MB	313.94 ms	173.87 ms	2.34 op/s	2.34 MB/S	100%

图 11 final result

从图中我们可以看到，读操作的 op 数和字节数比写操作更多，但平均处理时间却比写操作平均处理时间要短的多，吞吐量和带宽也是写操作的数倍，说明读性能远远优于写性能。

10. 块大小对处理结果的影响

将负载样例中 8kb~1mb work 的 workers 数量改为 8，块的大小保持不变，提交修改后的负载样例，运行的最终结果如图 12。

Op-Type	Op-Count	Byte-Count	Avg-ResTime	Avg-ProcTime	Throughput	Bandwidth	Succ-Ratio
op1: init -write	0 ops	0 B	N/A	N/A	0 op/s	0 B/S	N/A
op1: prepare - write	8 ops	64 KB	4082 ms	4081.62 ms	2.17 op/s	17.36 KB/S	100%
op2: prepare - write	8 ops	128 KB	4460.75 ms	4459.75 ms	1.81 op/s	28.93 KB/S	100%
op3: prepare - write	8 ops	256 KB	4532.5 ms	4531 ms	1.77 op/s	56.5 KB/S	100%
op4: prepare - write	8 ops	512 KB	4164.62 ms	4162 ms	1.94 op/s	124.13 KB/S	100%
op5: prepare - write	8 ops	1.02 MB	1937.38 ms	1681.5 ms	5.03 op/s	643.54 KB/S	100%
op6: prepare - write	8 ops	2.05 MB	1514.75 ms	643.5 ms	5.84 op/s	1.5 MB/S	100%
op7: prepare - write	8 ops	4.1 MB	3623.62 ms	1776.25 ms	2.74 op/s	1.4 MB/S	100%
op8: prepare - write	8 ops	8 MB	4930.62 ms	1889.5 ms	1.62 op/s	1.62 MB/S	100%
op1: read	2.2 kops	17.59 MB	7.18 ms	7.03 ms	74.12 op/s	593 KB/S	100%
op2: write	535 ops	4.28 MB	413.23 ms	411.65 ms	18.06 op/s	144.48 KB/S	100%
op1: read	2 kops	32.08 MB	4.93 ms	4.61 ms	68.08 op/s	1.09 MB/S	100%
op2: write	518 ops	8.29 MB	435.45 ms	431.63 ms	17.59 op/s	281.45 KB/S	100%
op1: read	1.98 kops	63.49 MB	4.81 ms	4.1 ms	66.67 op/s	2.13 MB/S	100%
op2: write	498 ops	15.94 MB	458.7 ms	455.59 ms	16.73 op/s	535.51 KB/S	100%
op1: read	2.11 kops	134.91 MB	6.29 ms	4.67 ms	70.91 op/s	4.54 MB/S	100%
op2: write	551 ops	35.26 MB	407.45 ms	399.19 ms	18.52 op/s	1.19 MB/S	100%
op1: read	2.14 kops	274.3 MB	8.52 ms	4.73 ms	72.1 op/s	9.23 MB/S	100%
op2: write	498 ops	63.74 MB	440.7 ms	425.76 ms	16.75 op/s	2.14 MB/S	100%
op1: read	1.74 kops	445.44 MB	13.36 ms	5.25 ms	58.8 op/s	15.05 MB/S	100%
op2: write	429 ops	109.82 MB	497.69 ms	473.88 ms	14.49 op/s	3.71 MB/S	100%
op1: read	1.3 kops	665.6 MB	25.2 ms	6.1 ms	43.58 op/s	22.31 MB/S	100%
op2: write	311 ops	159.23 MB	661.72 ms	606.89 ms	10.43 op/s	5.34 MB/S	100%
op1: read	856 ops	856 MB	51.68 ms	7.84 ms	28.71 op/s	28.71 MB/S	100%
op2: write	223 ops	223 MB	871.21 ms	780.91 ms	7.48 op/s	7.48 MB/S	100%

图 12 块性能比较

通过与图 11 的对比可以发现，调整 workers 数量后，出现了如下变化：

1）平均处理时间相比之前的结果有所增加，且块越大，增加的越快；

2）吞吐量先在原来吞吐量的附近波动，之后迅速减小；

3）带宽先成倍增加，然后增加速度逐渐减小。

分析：当块较小时，服务器的带宽和处理能力足够，吞吐量能基本保持原来的水平。当块的大小成倍增加时，带宽的增长速度跟不上块的大小的增长速度，服务器的处理能力逐渐达到饱和，无法保持之前的处理速度，导致吞吐量开始下降。且块的大小越大，下降程度越大。

11. 吞吐量与带宽比较

如图 13、图 14 分别为吞吐量变化图和带块变化图：

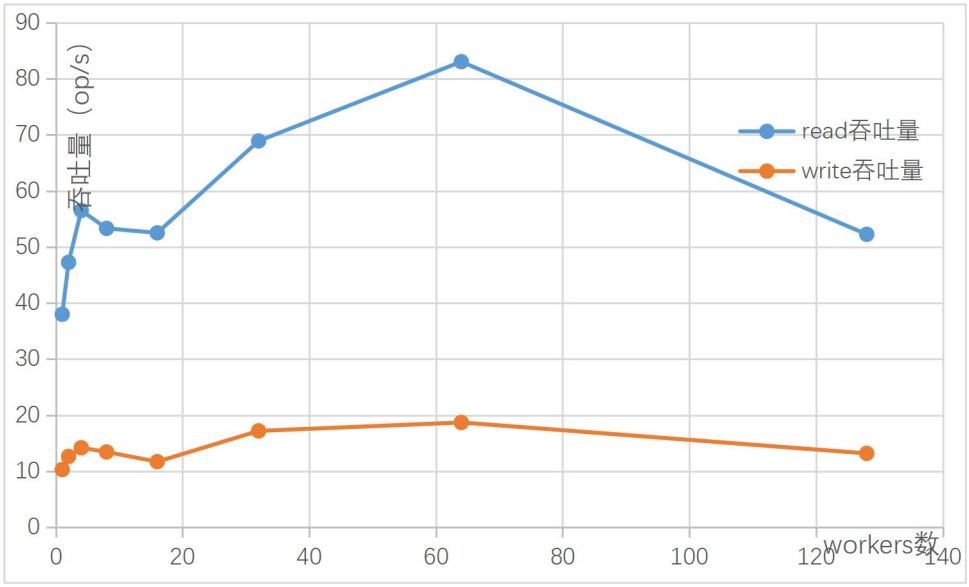

图 13 吞吐量变化图

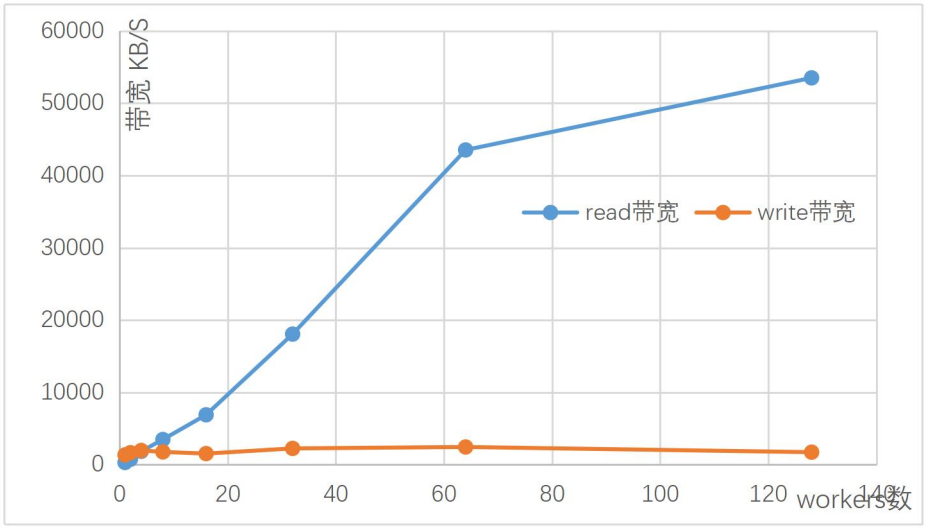

图 14 带宽变化图

从图中可以看到，相比于写操作的吞吐量变化，读操作的吞吐量变化更大，随着 workers 数的增加，基本上呈现出先增加后减小的趋势，同时，随着 workers 数的增加，读操作带宽逐渐增加，且增加速度逐渐减小；写操作带宽基本在原值附近波动，变化较小。综合比较，可以发现当 workers 数为 64 时，读写性能最

好。

六、实验总结

这次实验遇到的困难主要集中在环境的配置和对象存储系统的搭建等方面。在尝试了分别在 Windows 和 Linux 环境下下搭建对象存储系统后，我选择了 Windows 系统，不过网上关于在 Windows 系统下配置 cosbench 的教程比较少，同时由于版本问题也踩了很多坑，好在最终解决了问题。

此次主要进行了以下实验：

1. 块大小对性能的影响实验：我们通过修改块的大小，比较了不同块大小下的性能表现。实验结果表明，块大小对性能有一定的影响，较小的块大小可以提高吞吐量，但同时也会增加带宽的消耗。

2. Workers 数量对性能的影响实验：我们通过修改 workers 数量，比较了不同 workers 数量下的性能表现。实验结果表明，workers 数量对性能有明显的影响，增加 workers 数量可以提高吞吐量和带宽，但同时也会增加 CPU 的消耗。

3. 带宽和吞吐量的变化实验：我们通过修改数据的大小，比较了不同数据大小下的带宽和吞吐量的变化。实验结果表明，数据大小对带宽和吞吐量都有明显的影响，数据越大，带宽和吞吐量越高。

通过以上实验，我们可以得到以下结论：

1. 块大小、workers 数量和数据大小都会对存储性能产生影响，需要根据实际需求进行调整。

2. 对于需要高吞吐量的场景，可以适当增加 workers 数量和数据大小。

3. 对于需要高带宽的场景，可以适当减小块大小和数据大小。 综上所述，本次实验通过实践对象存储技术，深入探究了大数据存储与管理的相关技术，并对存储性能进行了分析，对于今后的大数据存储与管理工作具有一定的指导意义。

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.