

2019 级

《物联网数据存储与管理》课程

# 实 验 报 告

姓 名 夏棋

学 号 U201914868

班 号 CE1901

日 期 2022.04.14

# 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.....	1
4.2 对象存储性能分析.....	1
五、实验过程.....	1
六、实验总结.....	8
参考文献.....	8

- 1. 总体篇幅不做明确限制；
- 2. 突出说明实验内容、实验过程这两部分；
- 3. 实验背景、实验总结应精炼、客观、准确。

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

本次实验采用 minio 作为实验的存储服务端。

在对象存储里，元数据包括 account（用户）， bucket， bucket index 等信息。Minio 没有独立的元数据服务器，这个和 GlusterFs 的架构设计很类似，在 minio 里都保存在底层的本地文件系统里。

在本地文件系统里，一个 bucket 对应本地文件系统中的 一个目录。一个对象对应 bucket 目录下的一个目录（在 EC 的情况下对应多个 part 文件）。目录下保存者对象相关的数据和元数据。

评测工具采用 S3 Bench，此工具提供了针对 S3 兼容端点运行非常基本的吞吐量和基准测试的能力。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。

三、实验环境

硬件环境	CPU	Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz (8 CPUs), ~2.4GHz
	内存	16.0 GB
软件环境	操作系统	Windows 11
	其它软件	minio mc s3bench

四、实验内容

- 1.了解代码代码管理，运用 git
- 2.运用 minio 和 mc 创建服务端和客户端，访问和管理数据。
- 3.使用 s3bench 进行测试。

4.1 对象存储技术实践

使用 minio 搭建服务端，访问浏览器查看效果。用 mc 配置客户端进行简单操作并查看效果。

4.2 对象存储性能分析

修改脚本参数，分析脚本参数对存储性能的影响。

五、实验过程

按照规范要求下载并安装 minio server 和 minio client。

按照规范运行 minio server，得到如下结果：

```
D:\lab\bigdata-storage-experiment>minio server D:\lab\bigdata-storage-experiment\storage
API: http://10.21.202.109:9000 http://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin

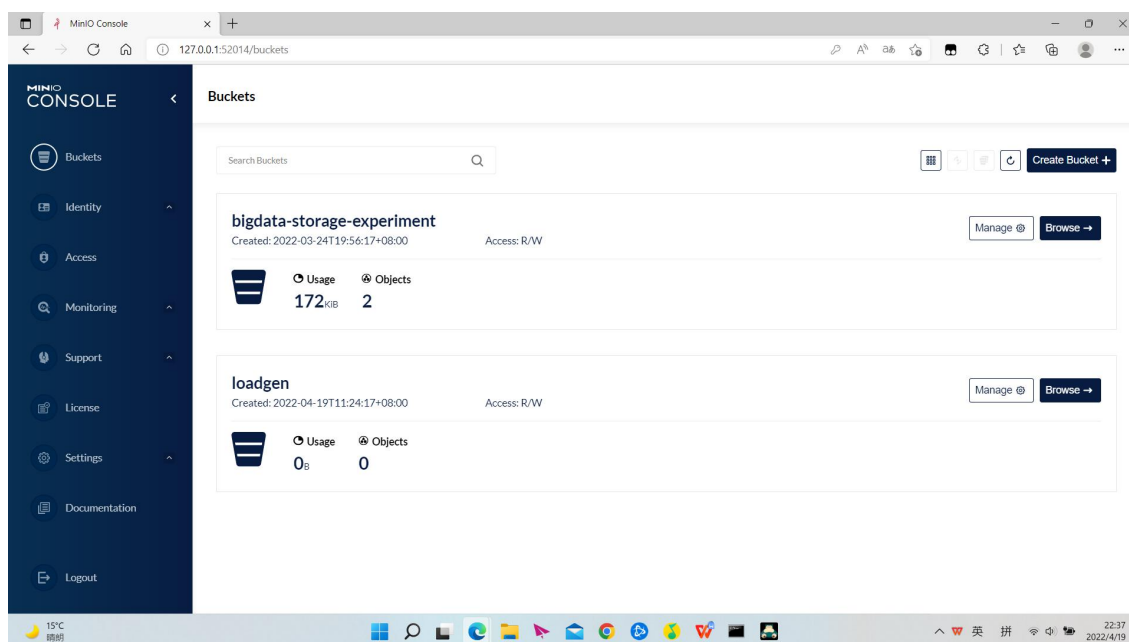
Console: http://10.21.202.109:52014 http://127.0.0.1:52014
RootUser: minioadmin
RootPass: minioadmin

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc.exe alias set myminio http://10.21.202.109:9000 minioadmin minioadmin

Documentation: https://docs.min.io
```

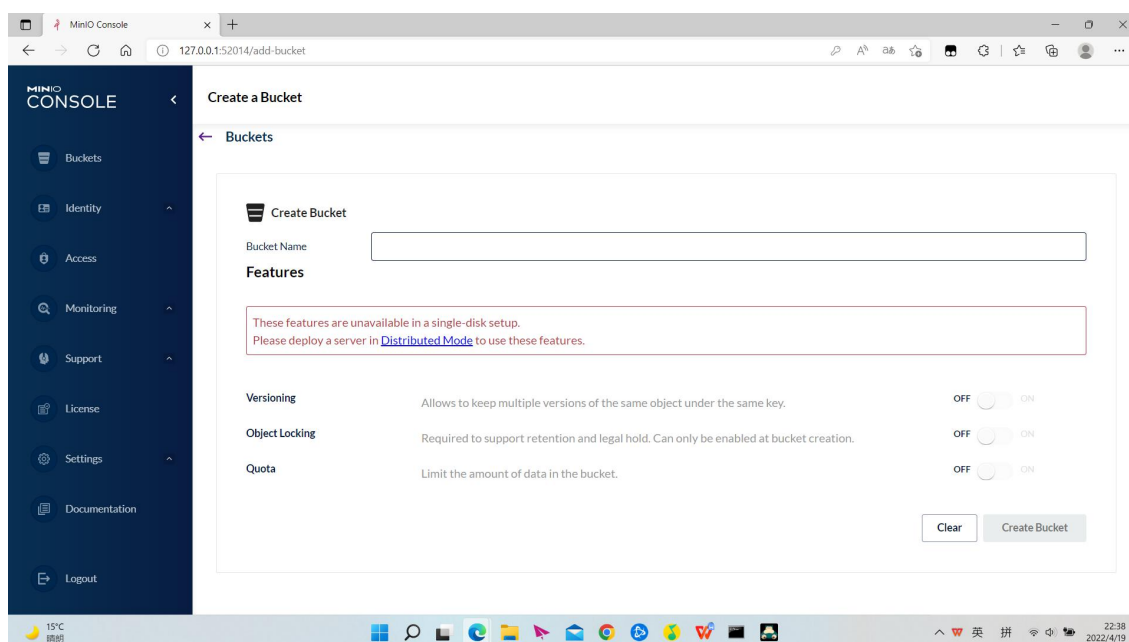
我们可知登录的初始账号名和密码均为默认的 minioadmin。

在浏览器输入 `http://127.0.0.1:52014` 找到对应界面输入用户名和密码进行登录



界面如上图所示。

创建 bucket:



输入创建的 bucket 名，可直接创建。

运行 mc.exe，添加一个存储服务（名字任取）hust。

```
D:\lab\bigdata-storage-experiment>mc config host add hust http://127.0.0.1:9000 minioadmin minioadmin --api s3v4
Added hust successfully.
```

然后展示所有的 bucket:

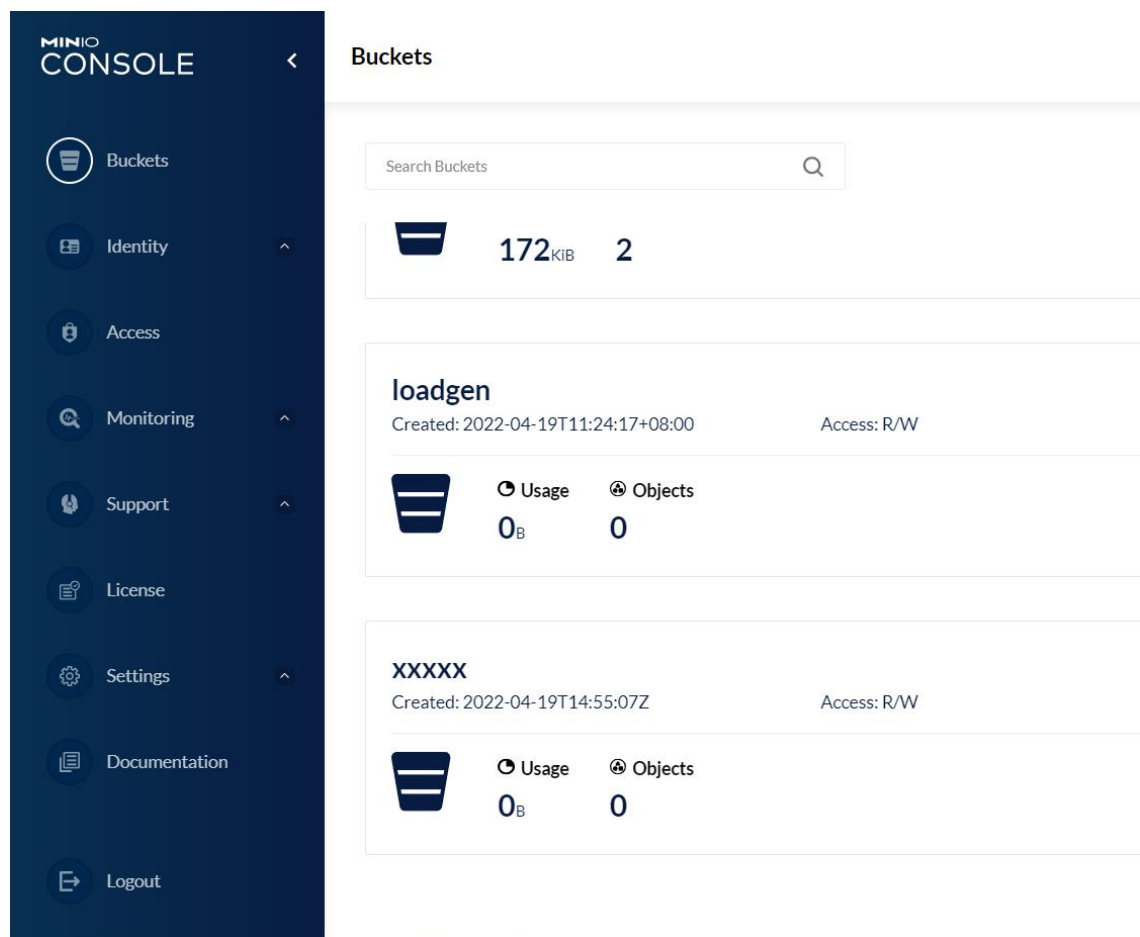
```
D:\lab\bigdata-storage-experiment>mc ls hust
[2022-03-24 19:56:17 CST]    0B bigdata-storage-experiment/
[2022-04-19 11:24:17 CST]    0B loadgen/
```

使用命令 mb，创建新的 bucket:

```
D:\lab\bigdata-storage-experiment>mc mb hust/l
mc: <ERROR> Unable to make bucket `hust/l`. Bucket name cannot be shorter than 3 characters

D:\lab\bigdata-storage-experiment>mc mb hust/xxxxx
Bucket created successfully `hust/xxxxx`.
```

这里不小心犯了一个错误，bucket 的名字不能小于三个字符。

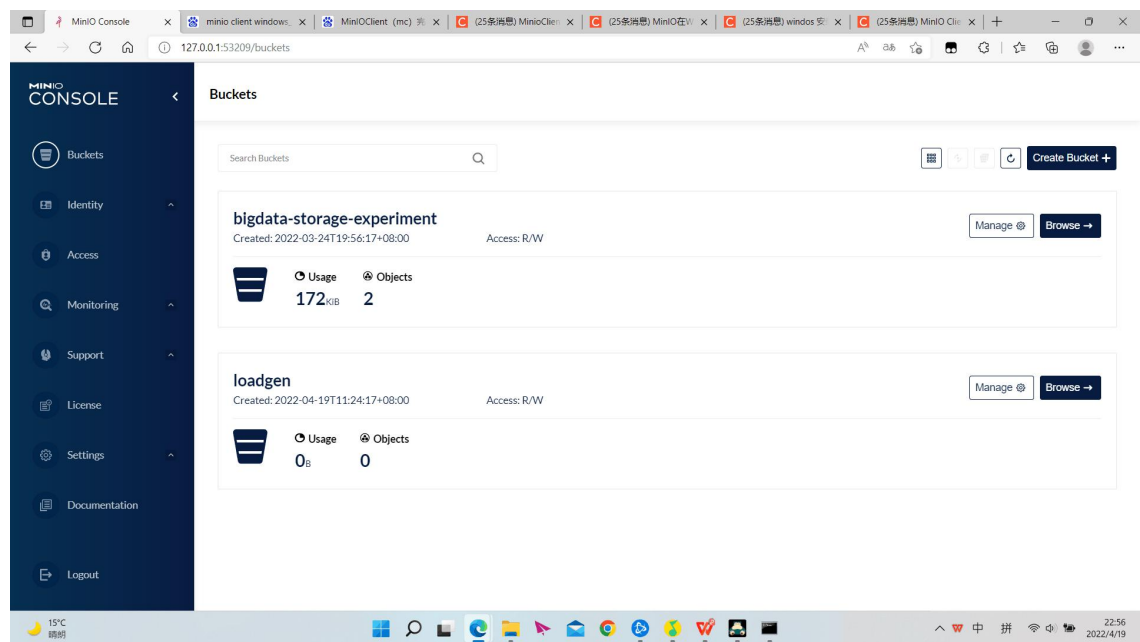


The screenshot displays the MinIO Console interface. On the left is a dark blue sidebar with the 'MINIO CONSOLE' header and a list of navigation items: Buckets, Identity, Access, Monitoring, Support, License, Settings, Documentation, and Logout. The main area is titled 'Buckets' and features a search bar. Below the search bar, there are two bucket entries. The first entry is for a bucket named 'loadgen', showing it was created on 2022-04-19T11:24:17+08:00 with R/W access, and contains 172 KiB of data and 2 objects. The second entry is for a bucket named 'XXXXXX', created on 2022-04-19T14:55:07Z with R/W access, and is currently empty (0 B, 0 objects).

回到浏览器中发现，多出一个名为“xxxxx”的 bucket，再使用 lb 进行删除:


```
D:\lab\bigdata-storage-experiment>mc rb hust/xxxxx
Removed hust/xxxxx successfully.
```

结果如下：



接下来分析存储性能：

这里才用 s3bench 进行测试，首先下载 s3bench 及其对应脚本

 run-s3bench.cmd	2022/4/19 11:28	Windows 命令脚本	1 KB
 s3bench.exe	2022/4/19 10:27	应用程序	11,596 KB

对脚本进行简单的修改，使其如下：

```
11 s3bench.exe ^
12     -accessKey=minioadmin ^
13     -accessSecret=minioadmin ^
14     -bucket=loadgen ^
15     -endpoint=http://127.0.0.1:9000 ^
16     -numClients=8 ^
17     -numSamples=256 ^
18     -objectNamePrefix=loadgen ^
19     -objectSize=1024
20 pause
```

账号密码改为默认密码，同时由脚本可知，需创建一个叫做 loadgen 的 bucket，我提前准备好了。

运行脚本：

```

Results Summary for Write Operation(s)
Total Transferred: 0.250 MB
Total Throughput: 0.15 MB/s
Total Duration: 1.641 s
Number of Errors: 0
-----
Write times Max: 0.108 s
Write times 99th %ile: 0.108 s
Write times 90th %ile: 0.079 s
Write times 75th %ile: 0.065 s
Write times 50th %ile: 0.051 s
Write times 25th %ile: 0.035 s
Write times Min: 0.012 s

Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput: 2.36 MB/s
Total Duration: 0.106 s
Number of Errors: 0
-----
Read times Max: 0.014 s
Read times 99th %ile: 0.014 s
Read times 90th %ile: 0.007 s
Read times 75th %ile: 0.004 s
Read times 50th %ile: 0.002 s
Read times 25th %ile: 0.002 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 324.934ms

D:\lab\bigdata-storage-experiment>pause
请按任意键继续. . .

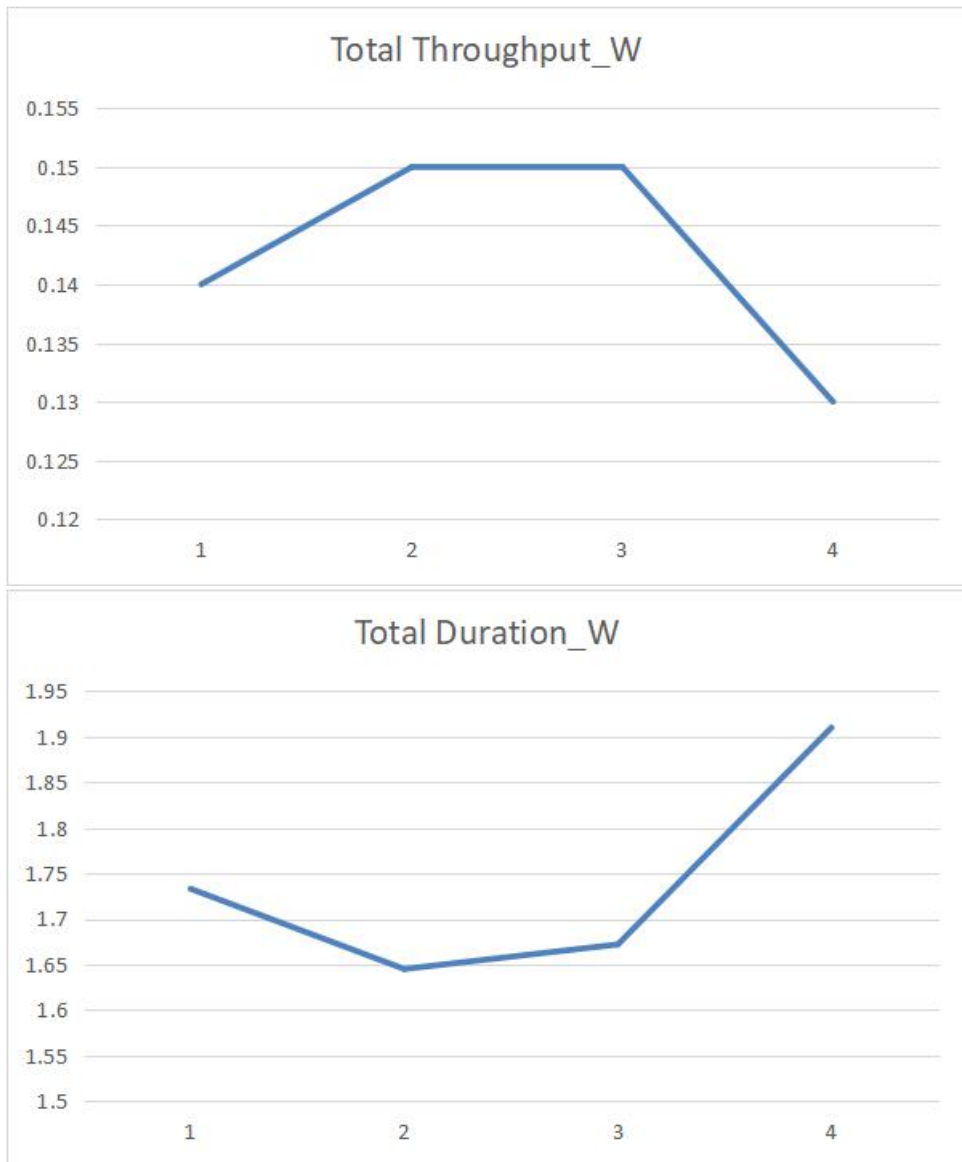
```

接下来所需要做的仅仅只需要改变参数测得对应数据的变化即可。为方便测量，采用 **throughput** 和 **duration** 作为性能观测指标。

1) 改变 Clients（并发数）（numsamples=256,objectsize=1024）测量性能：

Clients（并发数）	Total Throughput_W	Total Duration_W	Total Throughput_R	Total Duration_R
8	0.14	1.733	2.7	0.093
16	0.15	1.645	3.43	0.073
24	0.15	1.672	3.77	0.066
32	0.13	1.91	3.23	0.077





可知随着 clients 的增长，吞吐量先上升后下降，总用时先下降再上升。分析可知，当 clients 较小时，随着其增长，吞吐量上升，但并发客户端较多时，请求过多，会导致部分请求失败，使得吞吐量下降，用时和吞吐量呈反比。

2) 修改 numSamples(numclients=8, objectsize=1024) 进行性能测试：

Samples	Total Throughput_W	Total Duration_W	Total Throughput_R	Total Duration_R
256	0.19	1.308	3.4	0.073
512	0.17	2.876	3.91	0.1287
768	0.18	4.148	4.08	0.184
1024	0.15	6.602	3.78	0.265

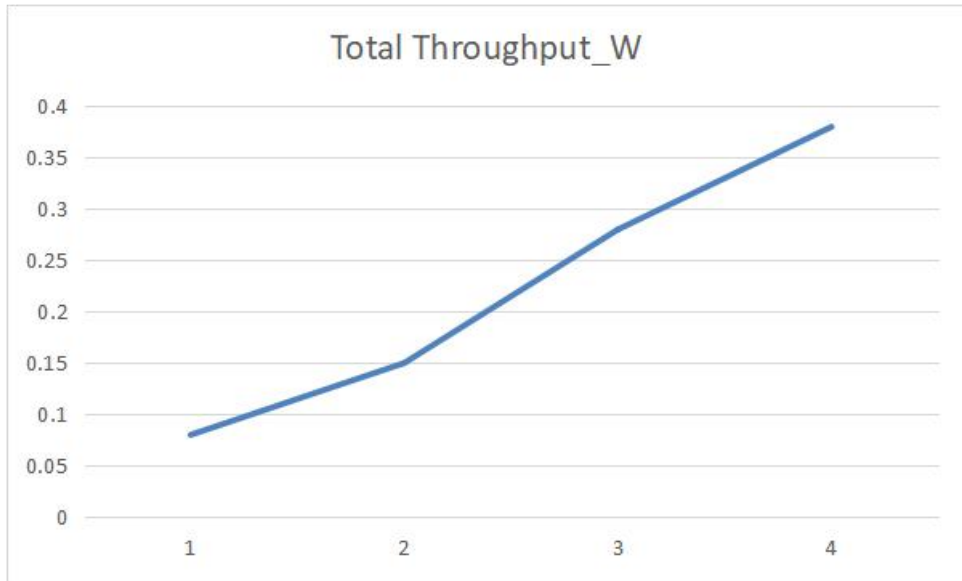




随着 samples 的增长，吞吐率基本不变，但是总耗时上升。

3) 修改 objectSize (numclients=8,numsamples=256)进行性能测试:

objectSize	Total Throughput_W	Total Duration_W	Total Throughput_R	Total Duration_R
512	0.08	1.604	1.82	0.069
1024	0.15	1.628	3.25	0.077
1536	0.28	1.339	5.27	0.071
2048	0.38	1.323	7.24	0.069



随着文件大小的提升，吞吐量不断增大。

## 六、实验总结

本次实验由于我选择了难度较为低的 minio 进行实验，所以较为顺利。其中最大的麻烦来自于，我一开始打算再 ubuntu 上完成实验，但是由于我对于 linux 系统的不熟练，导致使用时出现很多问题，最后转换回 Windows 平台进行实验。

同时在数据填表时，经常会出现同样的参数导致测量结果不同的现象。

## 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.

（可以根据实际需要更新调整）