

2020 级

《大数据存储系统与管理》课程

实 验 报 告

姓 名 倪志兴

学 号 U202015625

班 号 CS 本硕博 2001 班

日 期 2023.05.23

目 录

一、实验目的.....	2
二、实验背景.....	2
三、实验环境.....	2
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

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

对象存储是面向对象/文件的、海量的互联网存储，它也可以直接被称为“云存储”。对象尽管是文件，它是已被封装的文件（编程中的对象就有封装性的特点），也就是说，在对象存储系统里，你不能直接打开/修改文件，但可以像 `ftp` 一样上传文件，下载文件等。另外对象存储没有像文件系统那样有一个很多层级的文件结构，而是只有一个“桶”(bucket)的概念(也就是存储空间)，“桶”里面全部都是对象，是一种很扁平化的存储方式。其最大的特点就是它的对象名称就是一个域名地址，一旦对象被设置为“公开”，所有网民都可以访问到它；它的拥有者还可以通过 REST API 的方式访问其中的对象。因此，对象存储最主流的使用场景，就是存储网站、移动 app 等互联网/移动互联网应用的静态内容（视频、图片、文件、软件安装包等等）。

对象存储在很多重要方面与 SAN 和 NAS 迥然不同，对存储管理员而言最显著的区别在于对象存储没有 LUNs，卷以及 RAID 等要素。对象数据不是存储在固定的块，而是在大小可变的“容器”里。鉴于元数据 (metadata) 和数据本身可通过传统数据访问方法进行访问，对象存储允许数据被直接访问。此外，支持对象级和命令级的安全策略设置。

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，而一个对象文件可以是任意大小，从几 kb 到最大 5T 不等。

s3bench 是使用了 AWS Go SDK 实现的对象存储服务器测试程序，可以通过设置请求对象大小和数量、并行客户端数量等参数，根据吞吐率、延迟等结果评测对象存储系统的性能。

三、实验环境

本次对象存储测试实验通过虚拟机VMware在Linux系统中完成，具体的实验环境以及所用到的工具如下表所示。

实验环境操作系统	ubuntu-20.04.6
虚拟机软件	VMware Workstation 17Pro
CPU	Intel® Core™ i5-9300H CPU @ 2.40GHz
分配内存	10.6GB
服务端	minio
客户端	minio client
测试程序	s3bench

四、实验内容

搭建总体实验环境，包括 git、go 以及软件环境 minio、minio client、s3 bench。使用 s3bench 对系统性能进行分析，通过调配命令行定制参数，批量测试并对结果进行分析。

4.1 对象存储技术实践

1. 在 linux 虚拟机中配置实验基础环境，如 go 等。
2. 通过 minio 官网的教程和指导手册安装 minio sever 和 minio client。
3. 通过指导手册配置 minio 服务器，在 minio 服务器网页创建 bucket,并创建 minio client 向服务器发出请求，通过服务器网页进行观察验证。

4.2 对象存储性能分析

1. 使用 go get 命令获取 s3bench 测试环境。
2. 使用范例程序，修改参数与本机相对应，测试 minio 服务器性能
3. 再循环测试中不断修改-ObjectSize 参数的值，获取测试结果
4. 对测试结果进行分析。

五、实验过程

（编号说明实验过程及观测效果、数据、图表）

1. 搭建服务器环境

使用官网命令

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

```
chmod +x minio
```

```
sudo mv minio /usr/local/bin/
```

获取 minio 服务器，再使用命令

```
minio server ~/minio --console-address :9090
```

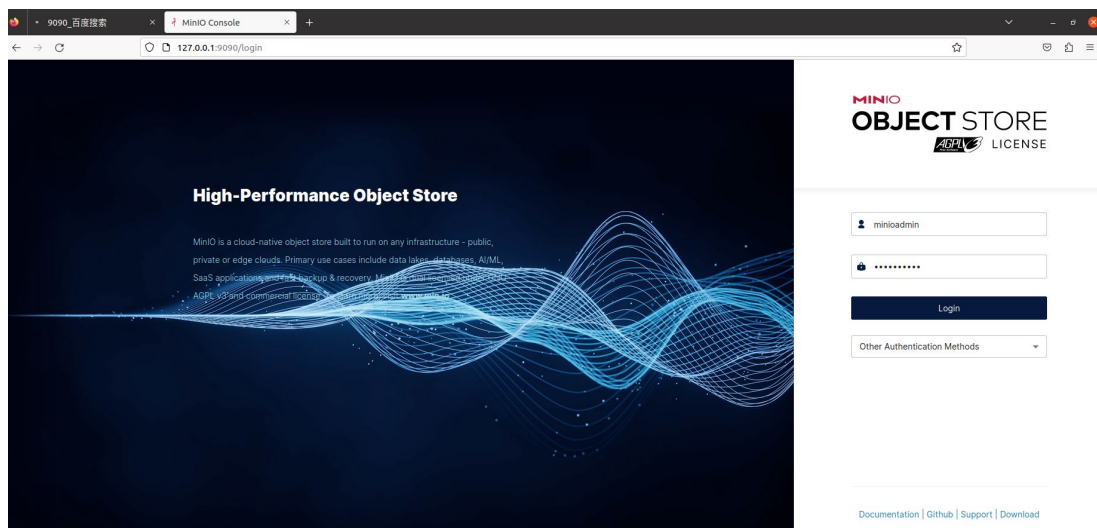
配置服务器端口号。

```
hzxing2@ubuntu:~/Downloads/minio$ mkdir ~/minio
hzxing2@ubuntu:~/Downloads/minio$ minio server ~/minio --console-address :9090
Formatting 1st pool, 1 set(s), 1 drives per set.
WARNING: Host local has more than 0 drives of set. A host failure will result in data becoming unavailable.
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 linux/amd64)

Status:          1 Online, 0 Offline.
S3-API: http://192.168.174.138:9000 http://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin

Console: http://192.168.174.138:9090 http://127.0.0.1:9090
RootUser: minioadmin
RootPass: minioadmin
```

打开 127.0.0.1.9090



输入账户密码，登陆。

2. 配置客户端环境

同样使用官网命令

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
```

```
chmod +x mc
```

```
sudo mv mc /usr/local/bin/mc
```

获取客户端，配置本地参数

```
mc alias set local http://127.0.0.1:9000 minioadmin minioadmin
```

```
mc admin info local
```

```
nzxing2@ubuntu:~/Downloads/client$ mc alias set local http://127.0.0.1:9000 minioadmin minioadmin
Added 'local' successfully.
nzxing2@ubuntu:~/Downloads/client$ mc admin info local
● 127.0.0.1:9000
  Uptime: 47 minutes
  Version: 2023-04-20T17:56:55Z
  Network: 1/1 OK
  Drives: 1/1 OK
  Pool: 1

Pools:
  1st, Erasure sets: 1, Drives per erasure set: 1

100 KiB Used, 2 Buckets, 100 Objects, 200 Versions
1 drive online, 0 drives offline
```

可以使用 `mc --help` 查看可执行命令

```
nzxing2@ubuntu:~/Downloads/client$ mc --help

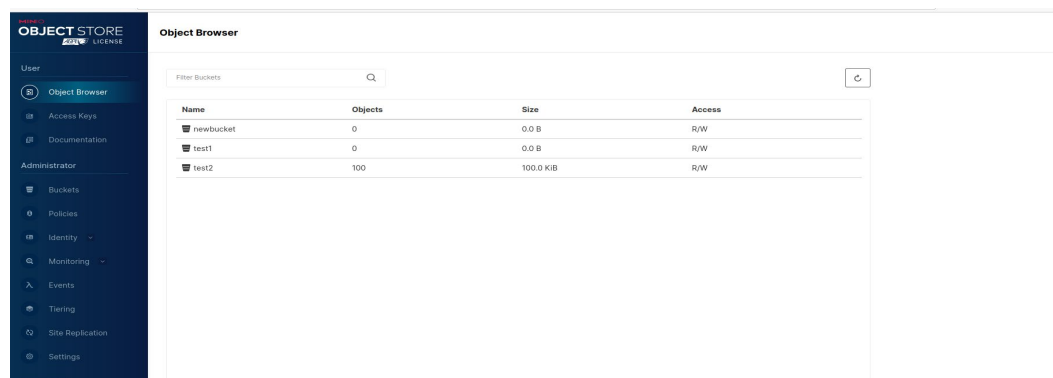
mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]

COMMANDS:
alias      manage server credentials in configuration file
ls         list buckets and objects
mb         make a bucket
rb         remove a bucket
cp         copy objects
mv         move objects
rm         remove object(s)
mirror     synchronize object(s) to a remote site
cat        display object contents
head       display first 'n' lines of an object
pipe       stream STDIN to an object
find       search for objects
sql        run sql queries on objects
stat       show object metadata
tree       list buckets and objects in a tree format
du         summarize disk usage recursively
retention  set retention for object(s)
legalhold  manage legal hold for object(s)
support    support related commands
license    license related commands
share      generate URL for temporary access to an object
version    manage bucket versioning
ilm        manage bucket lifecycle
quota      manage bucket quota
encrypt    manage bucket encryption config
event      manage object notifications
watch      listen for object notification events
undo       undo PUT/DELETE operations
```

使用 mc mb minio /newbucket 创建新 bucket




```
nzxing2@ubuntu:~/Downloads/client$ mc mb local/newbucket
Bucket created successfully `local/newbucket`.
nzxing2@ubuntu:~/Downloads/client$ mc ls local
[2023-05-25 17:02:00 CST]    0B newbucket/
[2023-04-23 15:38:48 CST]    0B test1/
[2023-04-23 16:04:32 CST]    0B test2/
```

同时查看网页



创建一个新的文件 1.txt 测试移入 newbucket 测试

```
nzxing2@ubuntu:~/Downloads/client$ touch 1.txt
nzxing2@ubuntu:~/Downloads/client$ vi 1.txt
nzxing2@ubuntu:~/Downloads/client$
nzxing2@ubuntu:~/Downloads/client$ mc cp 1.txt local/newbucket
...xing2/Downloads/client/1.txt: 230 B / 230 B
```

Name	Objects	Size	Access
 newbucket	1	230.0 B	R/W
 test1	0	0.0 B	R/W
 test2	100	100.0 KiB	R/W

3. 配置使用 s3bench

使用指导手册中的命令

go get -u github.com/igneous-systems/s3bench

获取 s3bench, 在 go/bin 目录下运行

./s3bench

-accessKey=minioadmin

-accessSecret=minioadmin

-endpoint=http://127.0.0.1:9000

-bucket=newbucket

-objectNamePrefix=loadgen

-numClients=10

-numSamples=100

-objectSize=1024

```
nzxing2@ubuntu:~/go/bin$ ./s3bench -accessKey=minioadmin -accessSecret=minioadmin -endpoint=http://127.0.0.1:9000 -bucket=newbucket -objectNamePrefix=loadgen -numClients=10 -n
umSamples=100 -objectSize=1024
Test parameters
endpoint(s): [http://127.0.0.1:9000]
bucket: newbucket
objectNamePrefix: loadgen
objectSize: 0.0010 MB
numClients: 10
numSamples: 100
verbose: %d(bool=false)

Generating in-memory sample data... Done (31.048µs)

Running Write test...

Running Read test...

Test parameters
endpoint(s): [http://127.0.0.1:9000]
bucket: newbucket
objectNamePrefix: loadgen
objectSize: 0.0010 MB
numClients: 10
numSamples: 100
verbose: %d(bool=false)

Results Summary for Write Operation(s)
Total Transferred: 0.098 MB
Total Throughput: 1.42 MB/s
Total Duration: 0.069 s
Number of Errors: 0
.....
```

```

-----
Write times Max:      0.022 s
Write times 99th %ile: 0.022 s
Write times 90th %ile: 0.012 s
Write times 75th %ile: 0.008 s
Write times 50th %ile: 0.005 s
Write times 25th %ile: 0.004 s
Write times Min:      0.002 s

Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  2.13 MB/s
Total Duration:    0.046 s
Number of Errors:  0
-----
Read times Max:      0.018 s
Read times 99th %ile: 0.018 s
Read times 90th %ile: 0.008 s
Read times 75th %ile: 0.006 s
Read times 50th %ile: 0.003 s
Read times 25th %ile: 0.002 s
Read times Min:      0.001 s

Cleaning up 100 objects...
Deleting a batch of 100 objects in range {0, 99}... Succeeded
Successfully deleted 100/100 objects in 40.12999ms

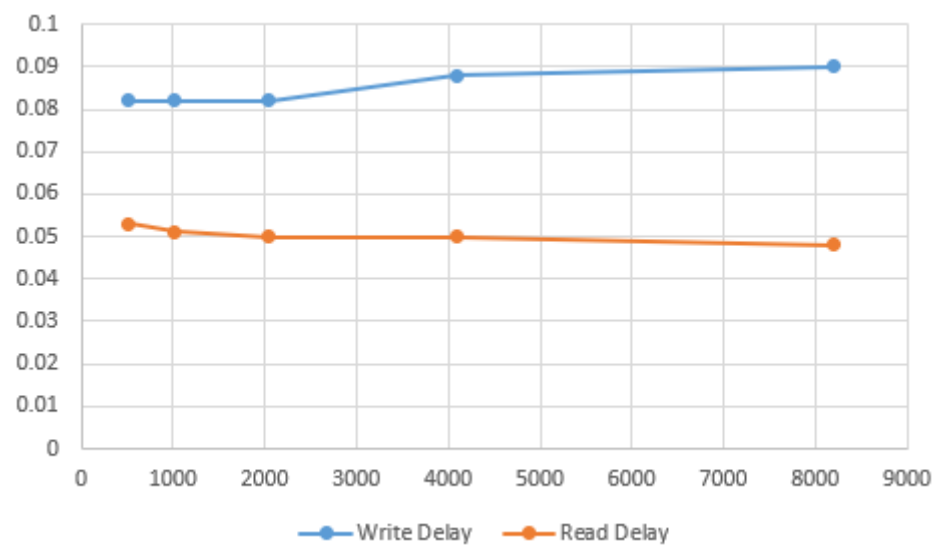
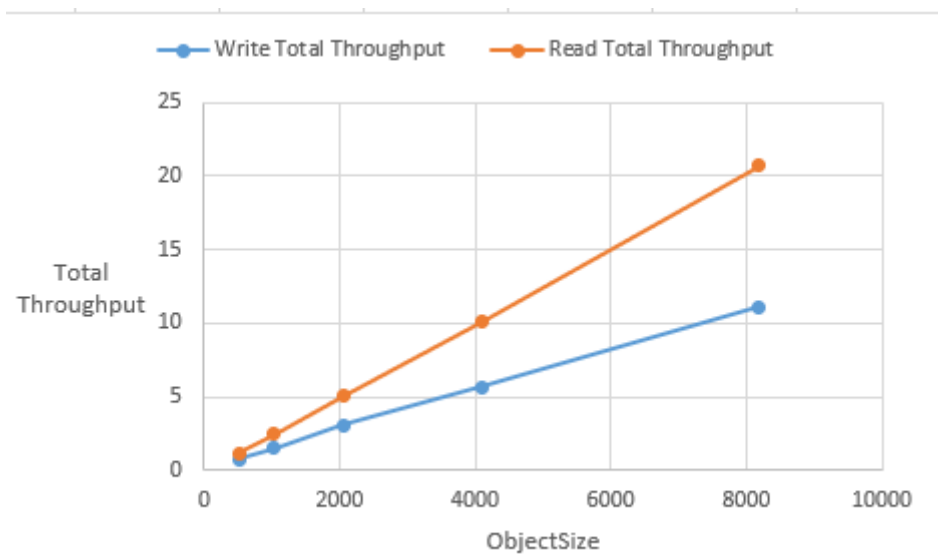
```

4. 测试分析

1) 对象大小对性能的影响

设置 numClients 为 10 和 numSamples 为 128 不变, 对 ObjectSize 从 512 增加至 8192 共 5 组, 分析速度和延迟

ObjectSize	Write Total Throughput	Write Delay	Read Total Throughput	Read Delay
512	0.77	0.082	1.17	0.053
1024	1.53	0.082	2.44	0.051
2048	3.07	0.082	5.05	0.05
4096	5.67	0.088	10.09	0.05
8192	11.09	0.09	20.72	0.048

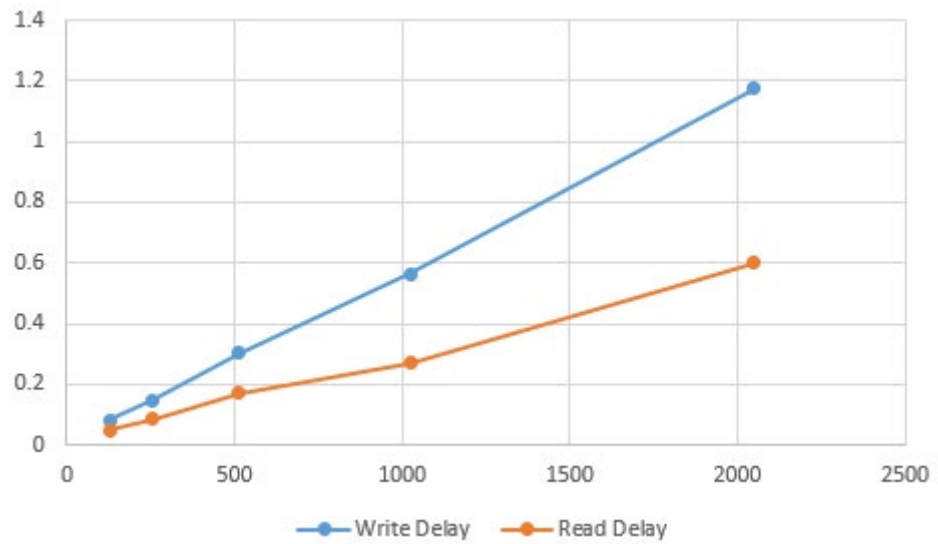


随着 ObjectSize 的增大，读写速度都呈线性增长，且读的速度接近些速度的两倍，但是读写延迟几乎不变。

2) 对象数量对性能的影响

设置 ObjectSize 为 2048 和 numClients 为 10 不变，对 numSamples 从 128 增加至 2048 共 5 组，分析速度和延迟

NumSamples	Write Total Throughput	Write Delay	Read Total Throughput	Read Delay
128	3.13	0.08	5.09	0.049
256	3.37	0.148	5.95	0.084
512	3.29	0.304	5.87	0.17
1024	3.54	0.565	7.39	0.271
2048	3.4	1.176	6.66	0.6

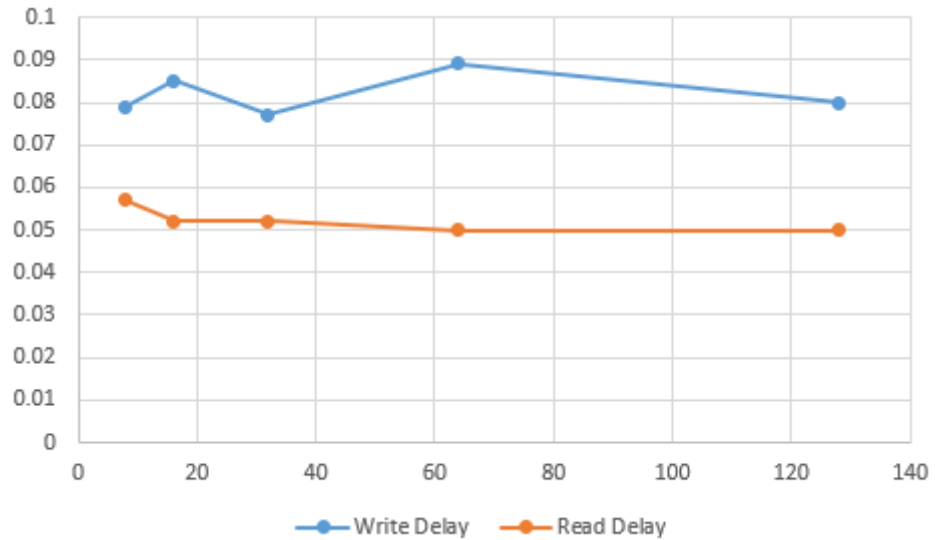
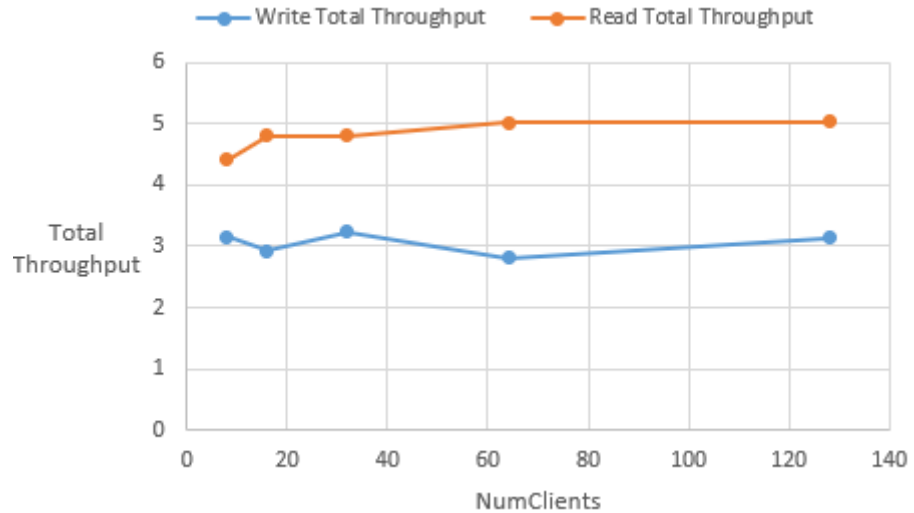


可以看到，随着对象数量的增加，写速度变化不大，读的速度先增大后减小，而读写延迟都明显增大。

3) 客户端数量对性能的影响

设置 ObjectSize 为 2048 和 numSamples 为 128 不变，对 numClients 从 8 增加至 128 共 5 组，分析速度和延迟

NumClients	Write Total Throughput	Write Delay	Read Total Throughput	Read Delay
8	3.15	0.079	4.41	0.057
16	2.93	0.085	4.8	0.052
32	3.24	0.077	4.81	0.052
64	2.81	0.089	5.01	0.05
128	3.14	0.08	5.04	0.05



由图像可知，读写速度与读写延迟会随客户端数量波动变化。

六、实验总结

此次实验主题是面向对象的存储系统，我采用的是比较简单的 minio+s3bench 的组合，在 linux 虚拟机下非常顺利地完成了安装与配置，mc 语句与 linux 命令十分相似易懂，很快就能正常测试并获取数据，但还是收获了许多，修改配置参数得到的数据也很多，能够发现一些明显的变化关联关系。算是对这个存储系统有了一定的认识，未来也许能够更深入地了解。

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 Sympos

- ium 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.