

**2018** 级

《物联网数据存储与管理》课程

**实 验 报 告**

**姓 名 张骞**

**学 号 U201814573**

**班 号 物联网1801班**

**日 期 2021.06.27**

**目 录**

[一、实验目的 1](#_Toc75717958)

[二、实验背景 1](#_Toc75717959)

[三、实验环境 2](#_Toc75717960)

[四、实验内容 3](#_Toc75717961)

[4.1 对象存储技术实践 3](#_Toc75717962)

[4.2 对象存储性能分析 8](#_Toc75717963)

[五、实验总结 9](#_Toc75717964)

[参考文献 10](#_Toc75717965)

# 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Ubuntu20.04.2 LTS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MC进行测试

对象存储服务器端：采用Minio

对象存储测试工具：采用cosbench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。

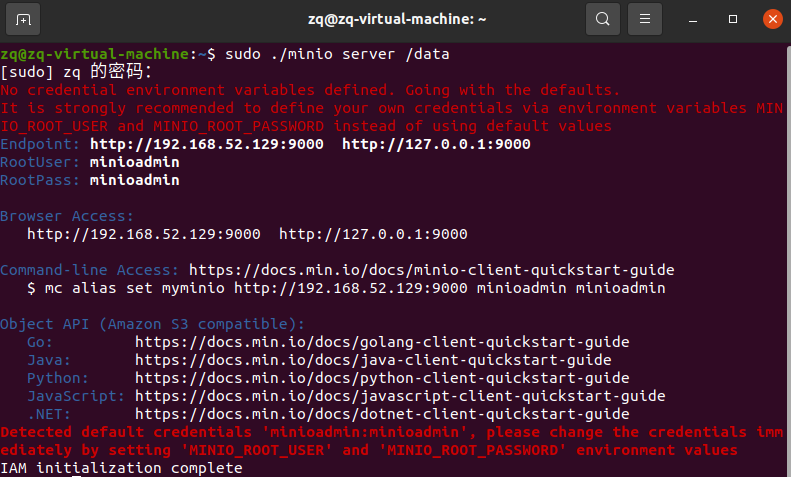


图4-1 运行Minio

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。

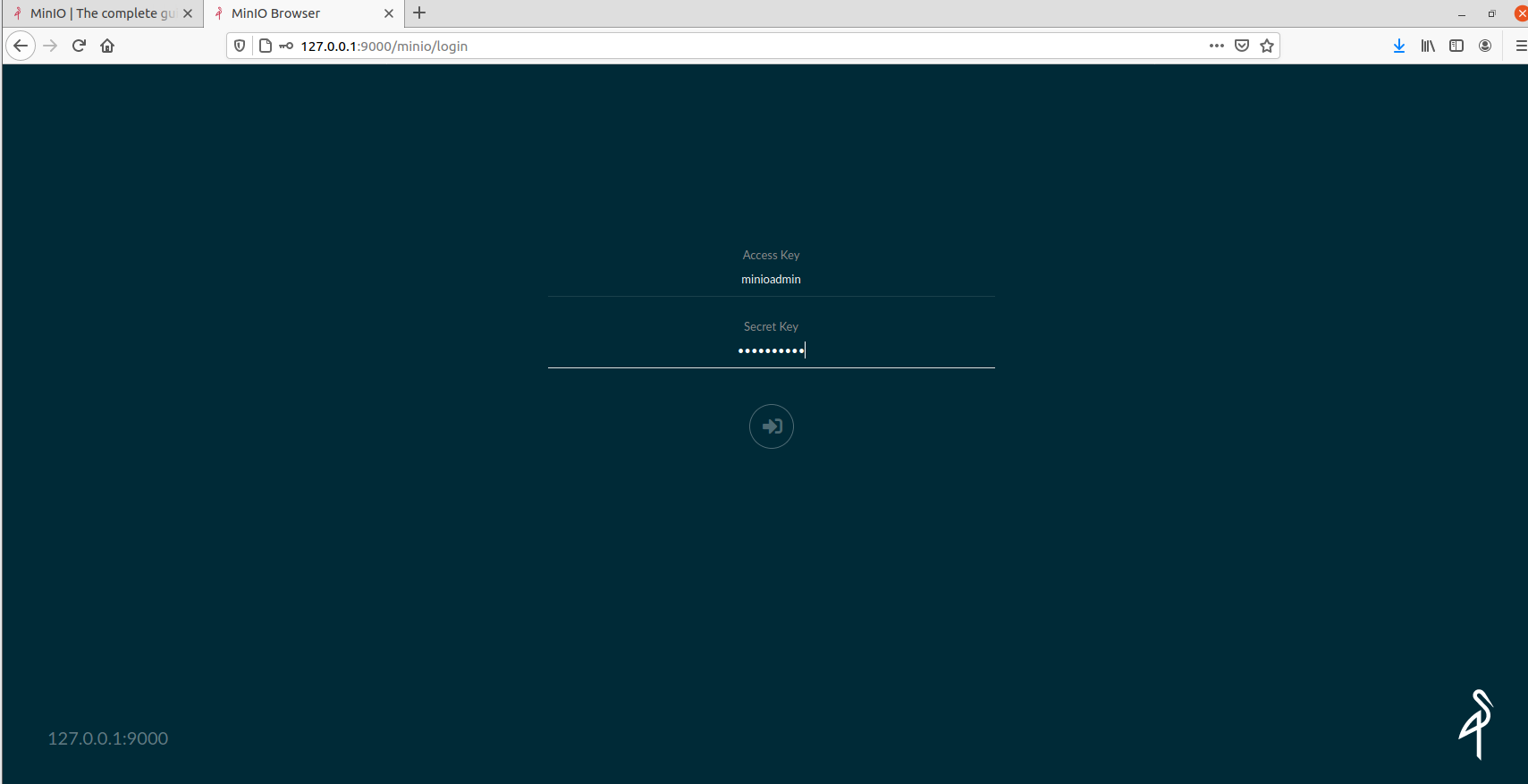


图4-2 登录服务器

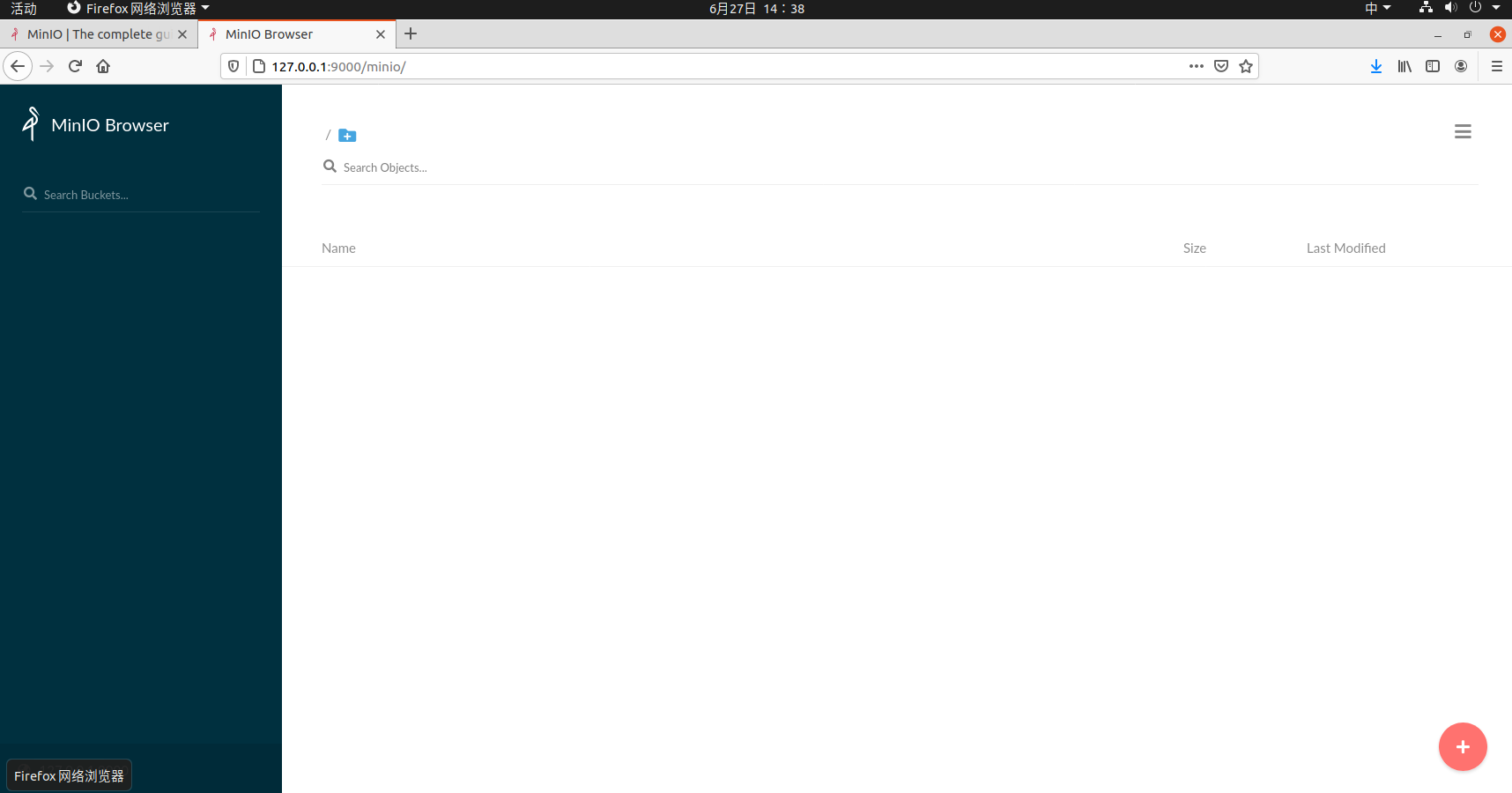


图4-3 在浏览器中访问服务器

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。在这里我们展示上传一个新的存储对象，如图4-4.

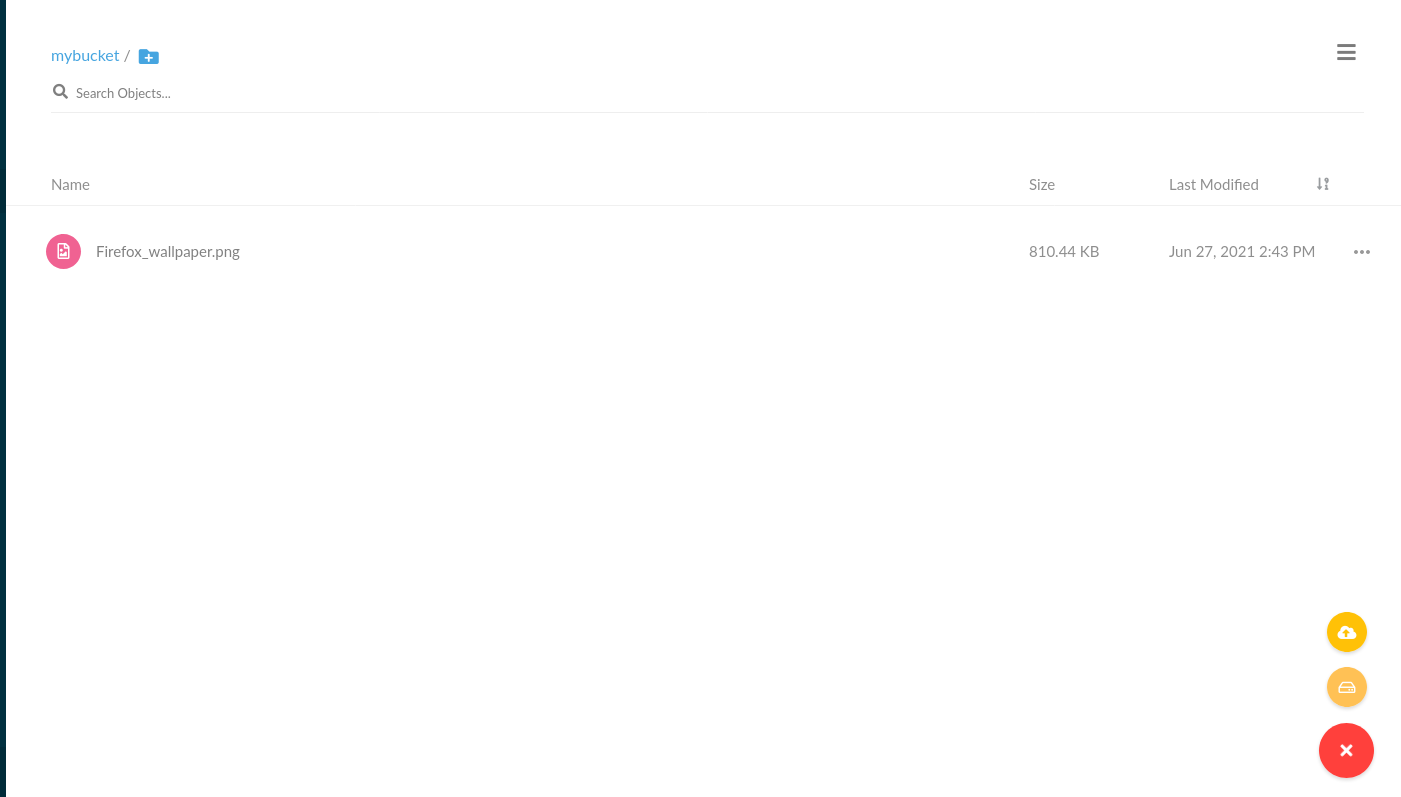


图4-4 添加存储对象

5）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 config host add myminio http://127.0.0.1:9000 minioadmin minioadmin，使得可以通过MC访问服务器，如图4-5。

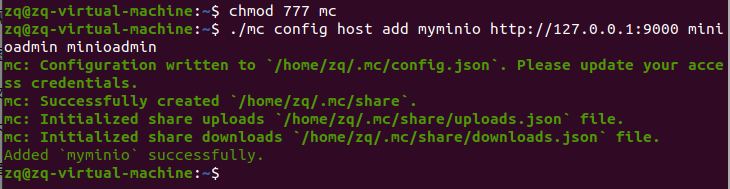


图4-5 通过MC访问服务器

6）使用MC对服务器进行增删。在命令行中输入./mc ls myminio/mybucket可以查看服务器中的文件，如图4-6。其中第一个文件是上面第四步中添加的。

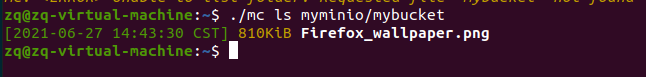


图4-6 MC的ls命令

可以利用MC的rm命令删除服务器中的文件，如图4-7，使用命令删除了这个png文件。如图4-8，还可以使用mb命令来添加仓库或文件。

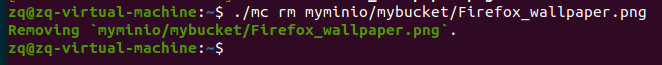


图4-7 MC的rm命令



图4-8 MC的mb命令

我们再通过ls命令查看，看是否真正的对服务器进行了删改。如图4-9，分别对test和test1都使用了ls命令。可以看到，在test目录下的图片文件已经被成功的删除，而在服务器中成功的添加了mybucket2这个包。

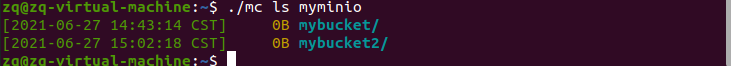


图4-9 通过ls命令验证

最终我们在浏览器中访问服务器，可以看到当前的状态如图4-10，这与我们在命令行中访问的结果是一致的。

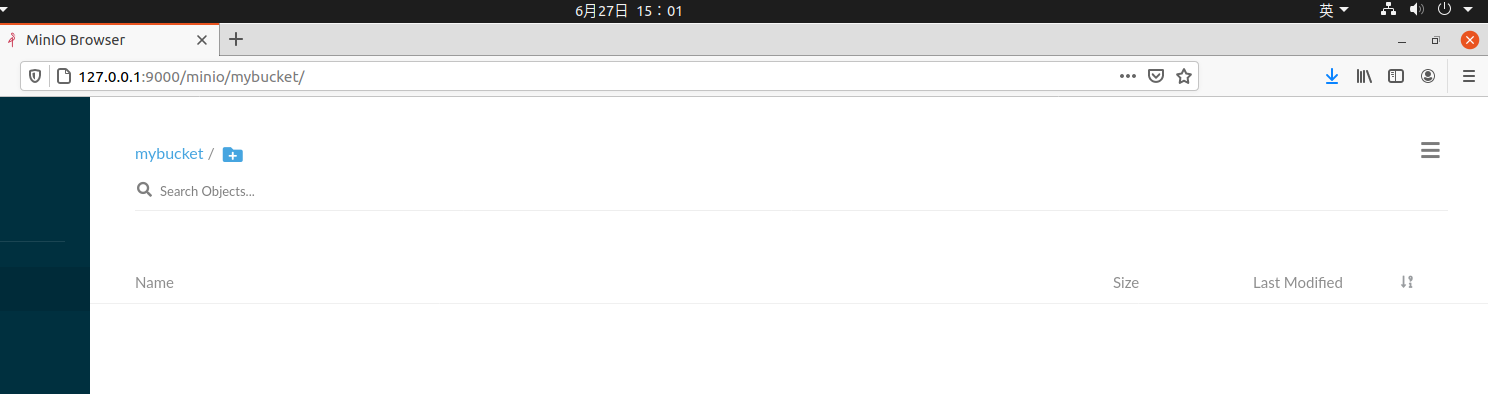


图4-10 通过浏览器访问服务器

7）使用测试工具cosbench进行测试。首先使用命令行unset http\_proxy绕过代理设置，使得控制器和驱动程序可以进行交互；然后运行脚本启动驱动程序和控制器，使用端口19088进行监听，如图4-11。

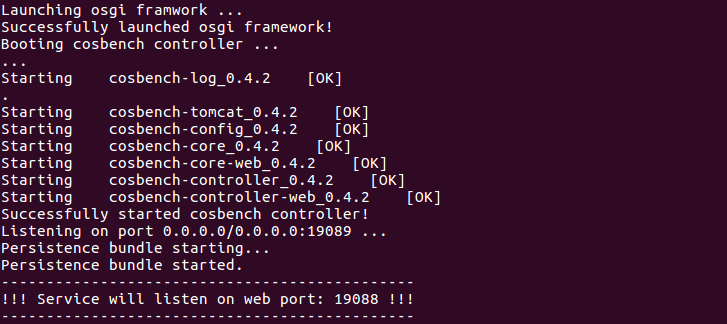


图4-11 启动驱动程序与控制器

然后用浏览器访问<http://127.0.0.1:19088/controller/index.html可以进入cosbench>的测试页面，如图4-12。

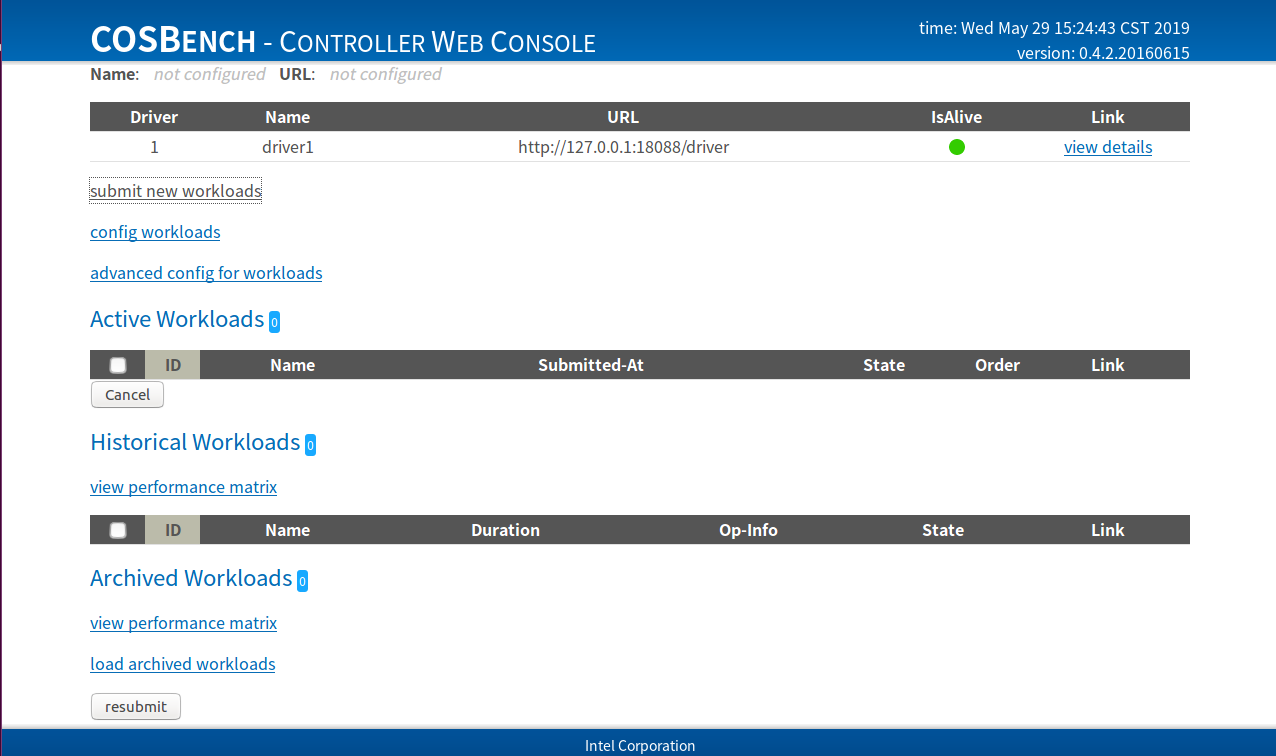


图4-12 cosbench测试页面

然后利用标准的测试样例，进行对服务器的标准测试，测试结果如图4-13，具体的分析在下一节中给出。

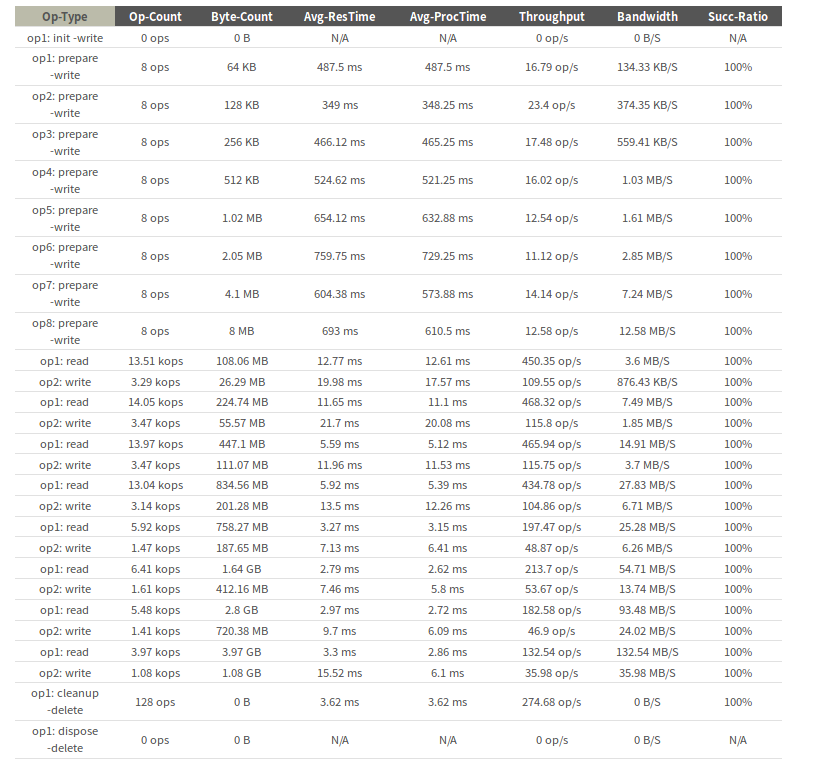


图4-13 cosbench测试结果

## 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cosbench结果如图4-18。可以看到全部12个任务都完成了。

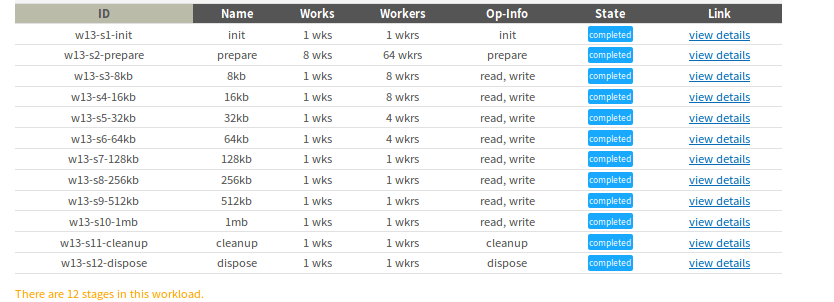


图4-18 cosbench测试结果（minio作为服务端）

更加详细的情况如图4-19，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。

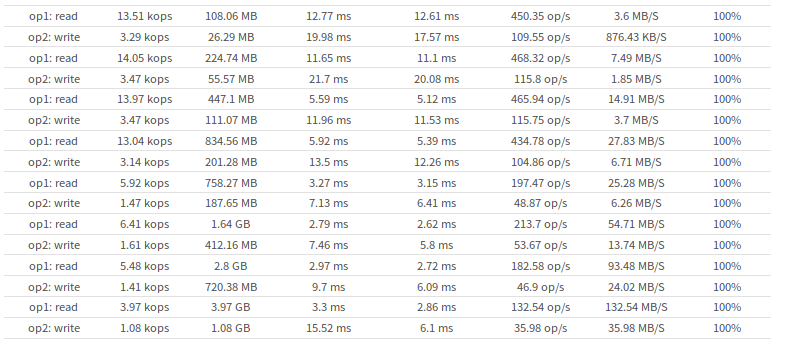


图4-19 详细测试结果（minio作为服务端）

# 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cosbench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。

环境的配置有之前学长学姐的经验，照葫芦画瓢其实也还好，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。最后采用cosbench测试是因为cosbench有图形界面，可视化程度较高

# 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.