

**2015** 级

《物联网数据存储与管理》课程

**实 验 报 告**

**姓 名 胡瑞星**

**学 号 U201514879**

**班 号 物联网1501班**

**日 期 2018.04.27**

**目录**

[一、实验目的 3](#_Toc1204)

[二、实验背景 3](#_Toc3552)

[三、实验环境 3](#_Toc30880)

[四、实验内容 4](#_Toc5943)

[五、实验过程 4](#_Toc9699)

[六、实验总结 9](#_Toc26043)

[参考文献 10](#_Toc8460)

# 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

对象存储是一种计算机数据存储架构，它将数据管理为对象，而不是像文件系统那样管理数据作为文件层次结构的其他存储架构，以及将数据作为扇区和路径中的块来管理数据的块存储。每个对象通常包括数据本身、一个变量元数据和一个全局唯一标识符。对象存储可以在多个级别实现，包括设备级别、系统级别和接口级别。在每种情况下，对象存储都试图启用其他存储架构所没有处理的功能，比如可以通过应用程序直接编程的接口，一个可以跨越物理硬件多个实例的名称空间，以及数据管理功能，如数据复制和对象级粒度的数据分布。

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。

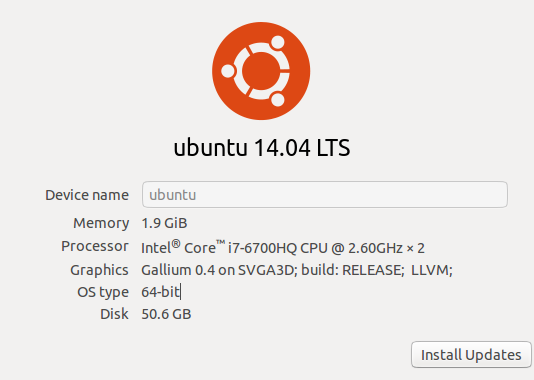
对象存储服务端：Minio server。

对象存储客户端：Minio client。

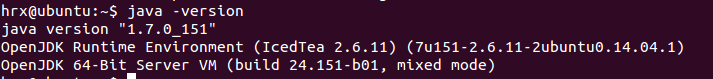
对象存储测评工具：COSBench。

# 三、实验环境

* 操作系统：



* Java：



* Python：

IMG_256

# 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作为第一阶段的尝试，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. 在本机上搭建一个Minio服务器端，将本机作为一个服务器。服务器存储空间即为本机的存储空间。
2. 在本机开启一个Minio客户端。可通过客户端对服务器内的文件进行操作。
3. 运行COSBench来对其进行测评。

## 4.2 对象存储性能分析

对Minio服务器运行COSBench进行测试。观察各项性能数据指标，修改测试代码，反复测试。

# 五、实验过程

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为S73J8ZCFSBUPT0JHLUNH，密码为 zB+pUndhllIeugUbwXV1bctkaRuPjG749uKf1LYa，这是初始的用户名与密码，可以登录修改。

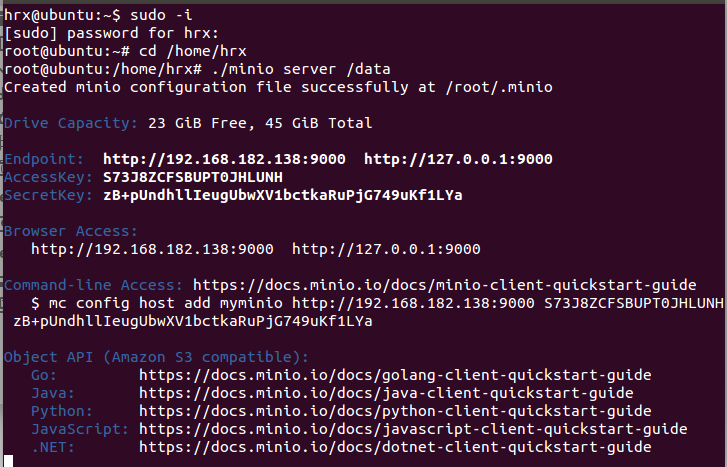


图4-1 运行Minio

1. 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。图中密码已经修改为自设的账号密码。确认登录后，可以看到界面如图4-3。

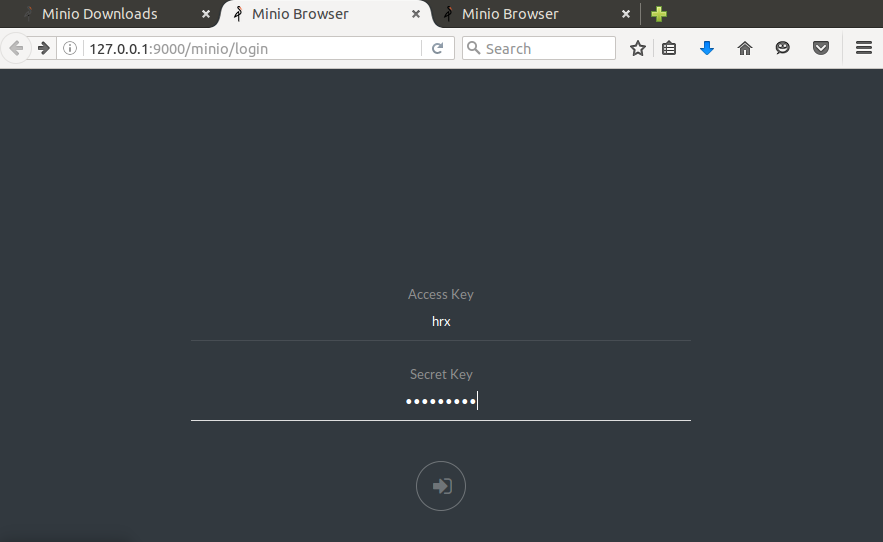


图4-2 登录服务器

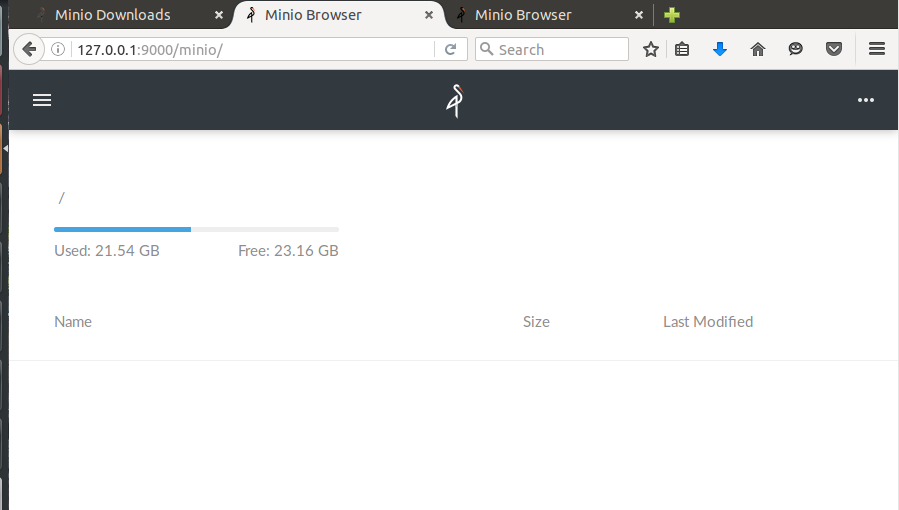


图4-3 在浏览器中访问服务器

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.

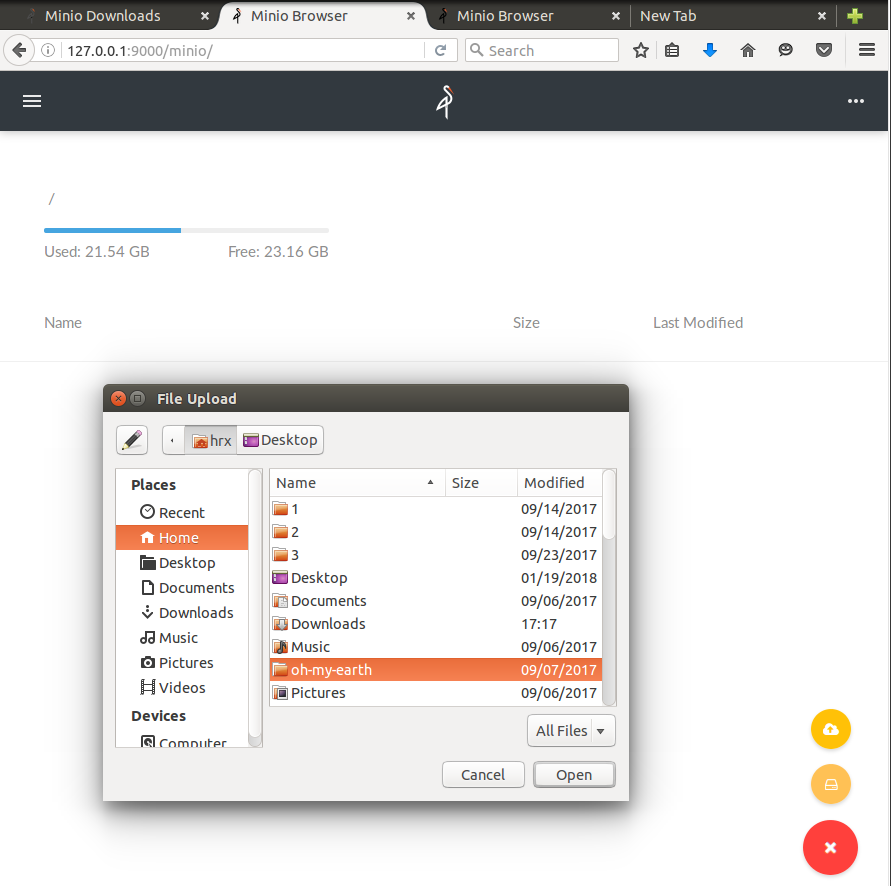


图4-4 添加存储对象

5）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4-5。

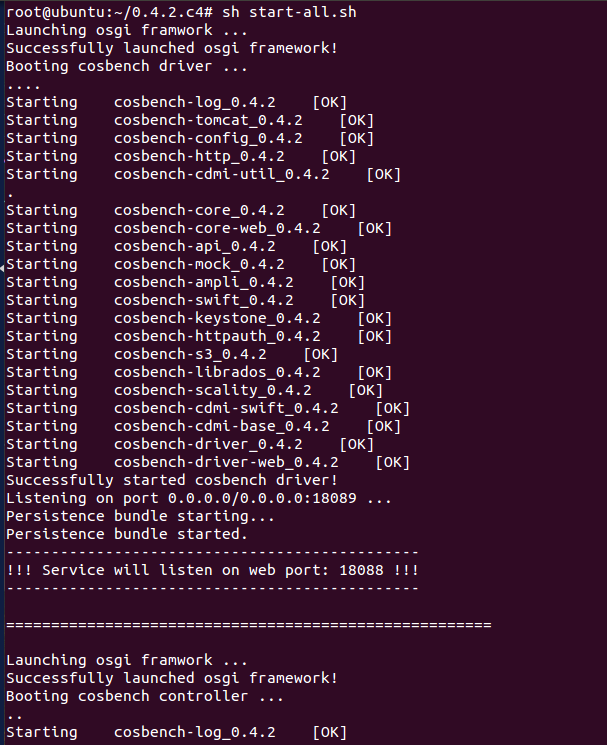


图4-5 启动驱动程序与控制器

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4-6。

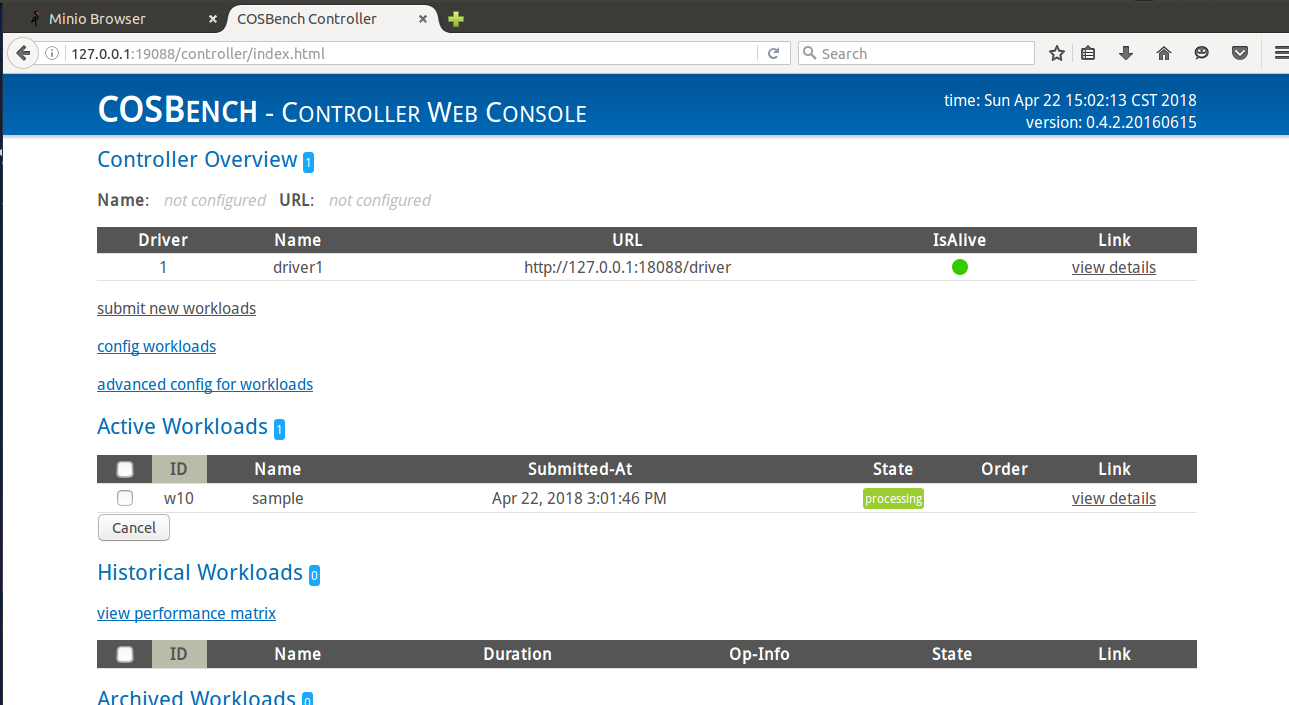


图4-6 cosbench测试页面

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4-7

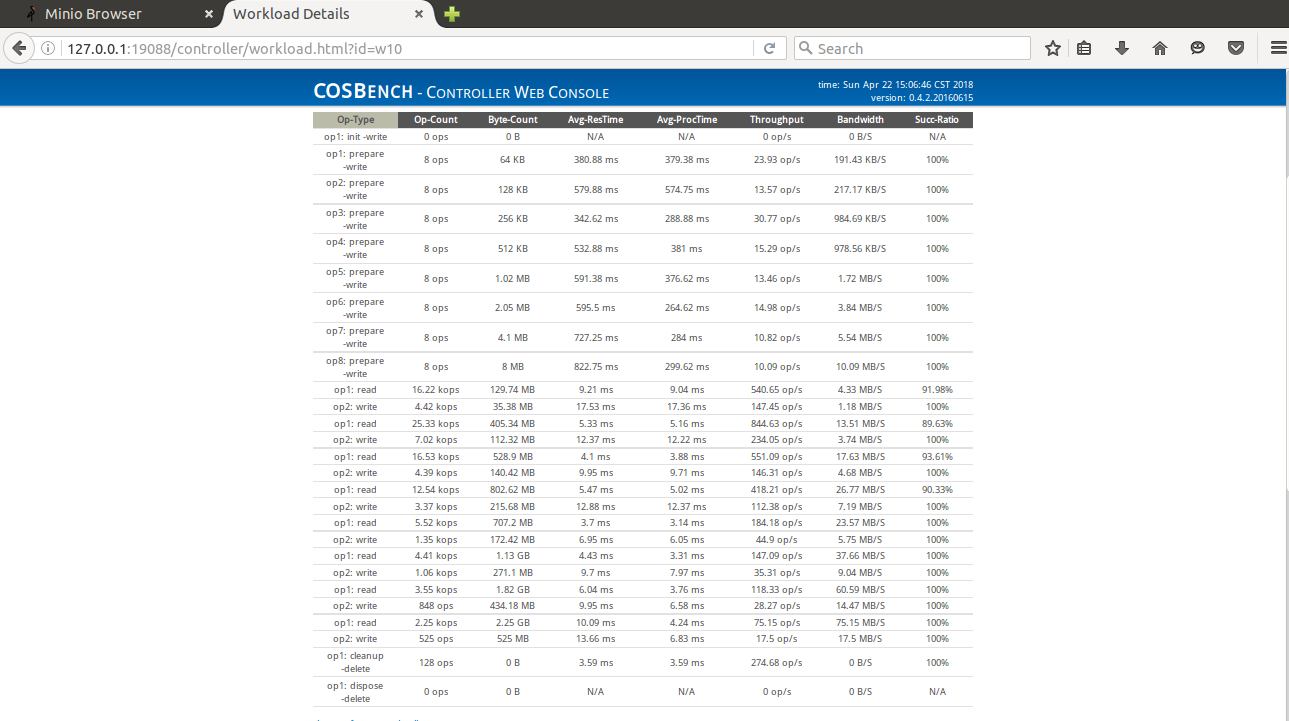


图4-7 cosbench测试结果

测试标准的cosbench结果如图4-8。可以看到全部12个任务都完成了。

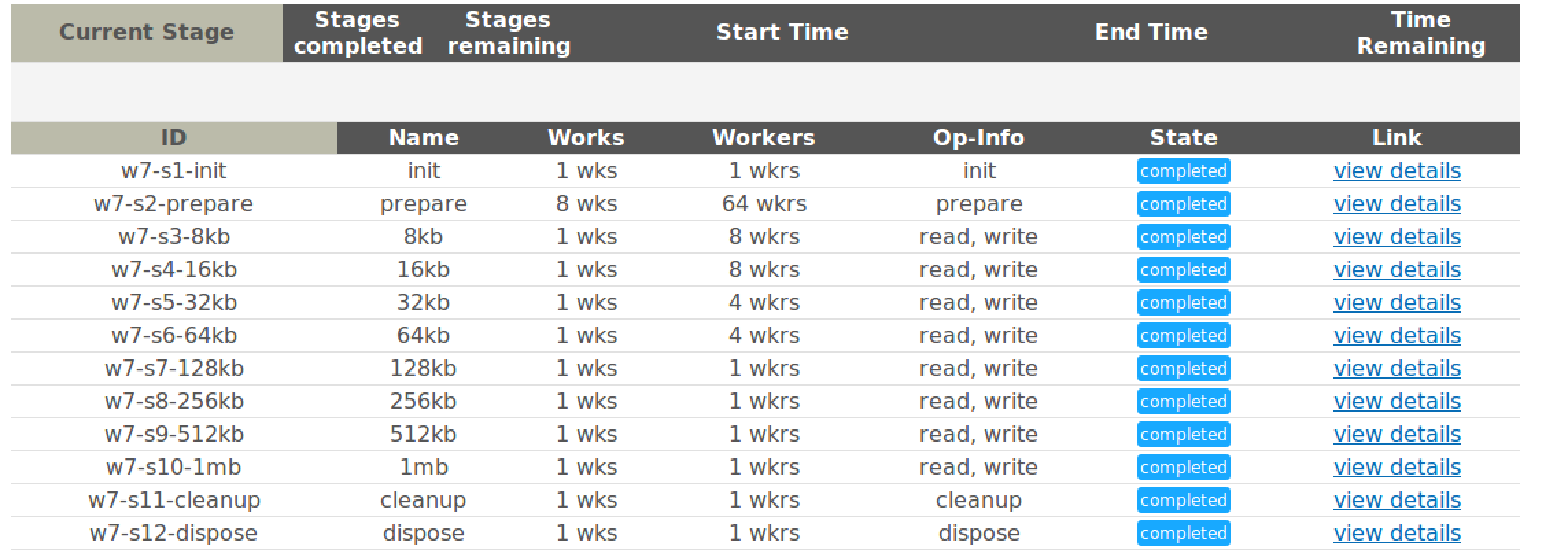


图4-8 cosbench测试结果

我们可以看到：

写入的成功率一直都是100%，而读取的成功率基本在90%-100%之间，在size较小的时候，成功率基本在90%左右，而随着size的增加，读取成功率一直保持在100%；

# 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。

这一次co's'bench测试的时候发现了当读写数据较小的时候，比较容易出错，但当测试的数据变大的时候，正确率反而提高了，但是我们测试的时候使用的数据量太少，不足以模拟当读写数据超过100m甚至tb级之上的情况，但也很好的揭示了当数据量偏小的时候的试验情况。

# 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）