|  |  |
| --- | --- |
| **分 数：** |  |
| **评卷人：** |  |

****

**数据中心技术课程实验报告**

**学 号 M202173842**

**姓 名 赵子龙**

**专 业 电子信息**

**课程指导教师 施展 童薇**

**院（系、所） 计算机科学与技术学院**

**2021年 1月7日**

## **目 录**

[1 系统搭建 2](#_Toc30299)

[1.1 实验环境 2](#_Toc14521)

[1.2 服务器搭建 2](#_Toc9343)

[2 性能观测 2](#_Toc9914)

[2.1 对象大小对性能的影响 2](#_Toc29913)

[2.2 并发数对性能的影响 7](#_Toc23481)

[3 尾延迟挑战 16](#_Toc1100)

[3.1 原尾延迟 16](#_Toc16431)

[3.2 对冲请求 Hedged Request 17](#_Toc17442)

[3.3 关联请求Tied Request 18](#_Toc18463)

# 1 系统搭建

## 实验环境

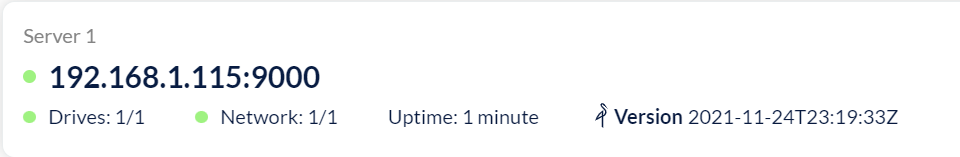
操作系统：Windows 10

处理器：Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz

测试工具：Minio，S3Bench

## 服务器搭建

1. 参照课程实验指导（<https://github.com/cs-course/obs-tutorial>）安装配置软件。
2. 在run-minio.cmd中设置账号（set MINIO\_ROOT\_USER）密码（set MINIO\_ROOT\_PASSWORD）并执行命令搭建Minio服务器。
3. 根据设定的账号密码和本地服务器地址登录服务器。
4. 在服务器上创建一个名为“loadgen”（与run-s3bench.cmd中的-bucket相同，可自行设置）的Bucket。
5. 运行基准测试。

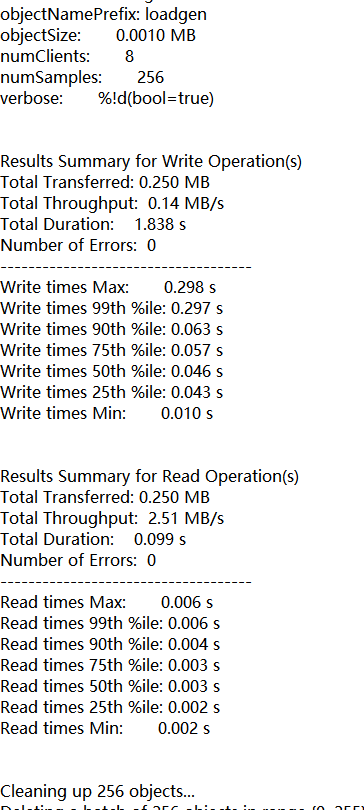


2 性能观测

使用S3Bench进行测试

## 2.1 对象大小对性能的影响

分别设置objectSize为1KB、10KB、100KB、1MB、10MB、100MB，numClients设置为8，numSamples设置为256进行性能观测：

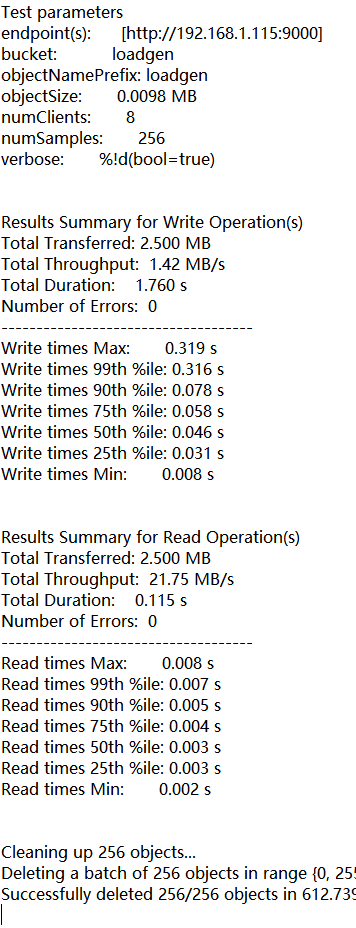


1KB numClients=8 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.14MB/s，总耗时1.838s，99%操作在0.297s内完成，25%操作在0.043s内完成。

读操作：吞吐率为2.51MB/s，总耗时0.099s，99%操作在0.006s内完成，25%操作在0.002s内完成。

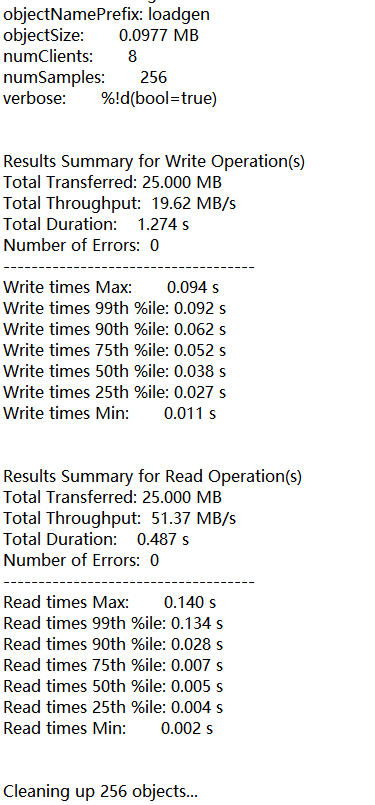


10KB numClients=8 numSamples=256

总文件大小为2.5MB

写操作：吞吐率为1.42MB/s，总耗时1.760s，99%操作在0.316s内完成，25%操作在0.031s内完成。

读操作：吞吐率为21.75MB/s，总耗时0.115s，99%操作在0.007s内完成，25%操作在0.003s内完成。

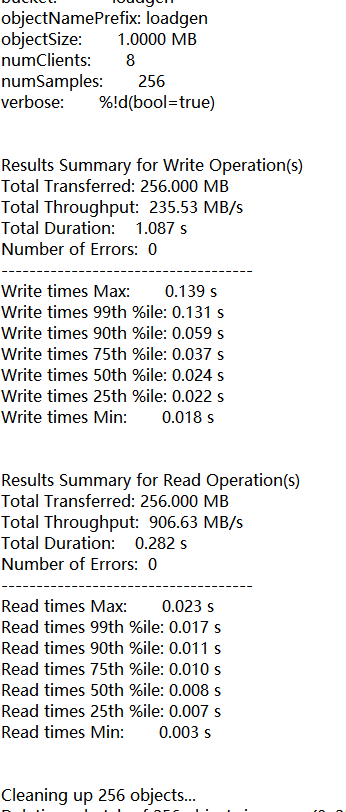


100KB numClients=8 numSamples=256

总文件大小为25MB

写操作：吞吐率为19.62MB/s，总耗时1.274s，99%操作在0.092s内完成，25%操作在0.027s内完成。

读操作：吞吐率为51.37MB/s，总耗时0.487s，99%操作在0.134s内完成，25%操作在0.004s内完成。

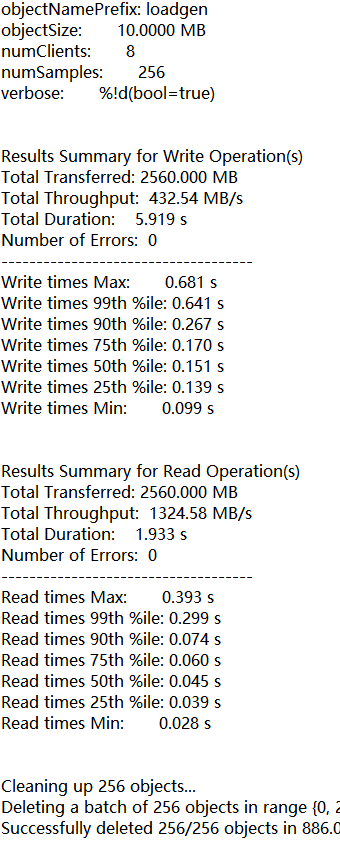


1MB numClients=8 numSamples=256

总文件大小为256MB

写操作：吞吐率为235.53MB/s，总耗时1.087s，99%操作在0.131s内完成，25%操作在0.022s内完成。

读操作：吞吐率为906.63MB/s，总耗时0.282s，99%操作在0.017s内完成，25%操作在0.007s内完成。

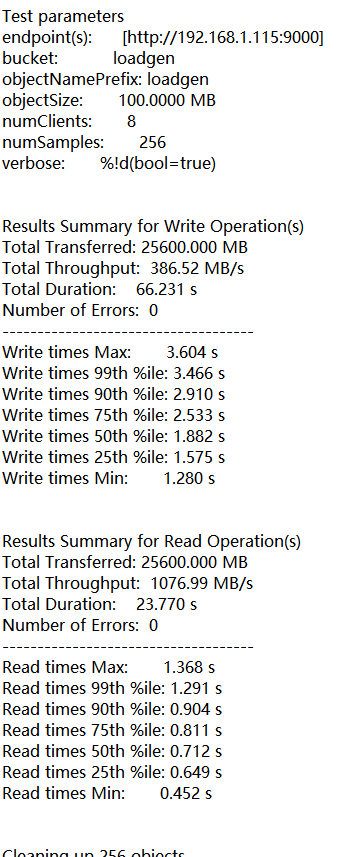


10MB numClients=8 numSamples=256

总文件大小为2560MB

写操作：吞吐率为432.54MB/s，总耗时5.919s，99%操作在0.641s内完成，25%操作在0.139s内完成。

读操作：吞吐率为1324.58MB/s，总耗时1.933s，99%操作在0.299s内完成，25%操作在0.039s内完成。



100MB numClients=8 numSamples=256

总文件大小为25600MB

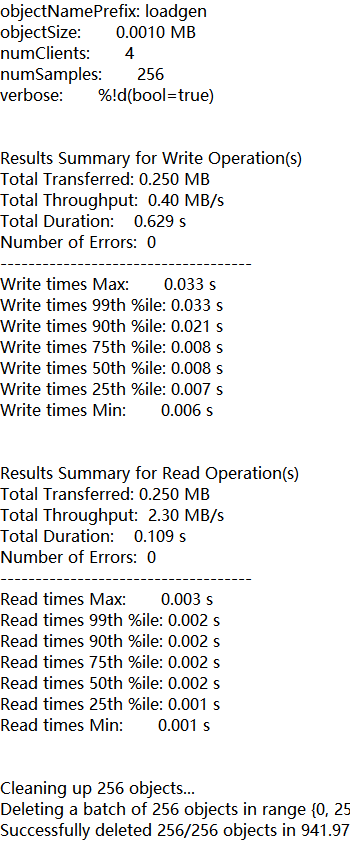
写操作：吞吐率为386.52MB/s，总耗时66.231s，99%操作在3.466s内完成，25%操作在1.575s内完成。

读操作：吞吐率为1076.99MB/s，总耗时23.770s，99%操作在1.291s内完成，25%操作在0.649s内完成。

观察发现：对象越大，读写耗时越多，尾延迟越明显。对象越大，读写吞吐率越大，但在对象大小达到100MB时读写吞吐率呈下降趋势，限制吞吐率的因素变成了并发线程单独处理任务的能力。在对象大小很小时（<100KB），延迟与吞吐率保持稳定。

## 2.2 并发数对性能的影响

分别设置objectSize为1KB、1MB，numClients分别设置为4、8、12、16、20、24、28、32，numSamples设置为256进行性能观测：

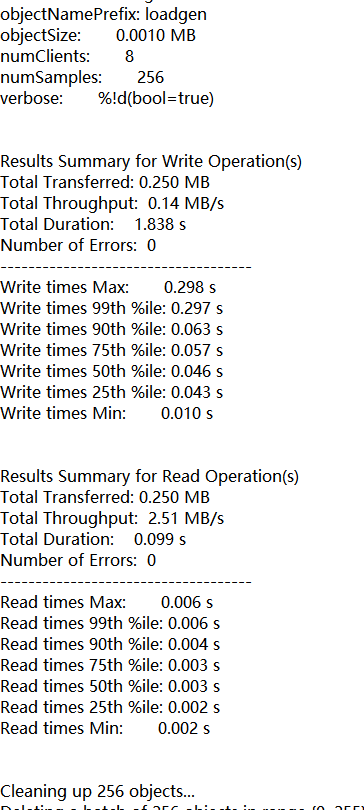


1KB numClients=4 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.4MB/s，总耗时0.629s，99%操作在0.033s内完成，25%操作在0.007s内完成。

读操作：吞吐率为2.3MB/s，总耗时0.109s，99%操作在0.002s内完成，25%操作在0.001s内完成。

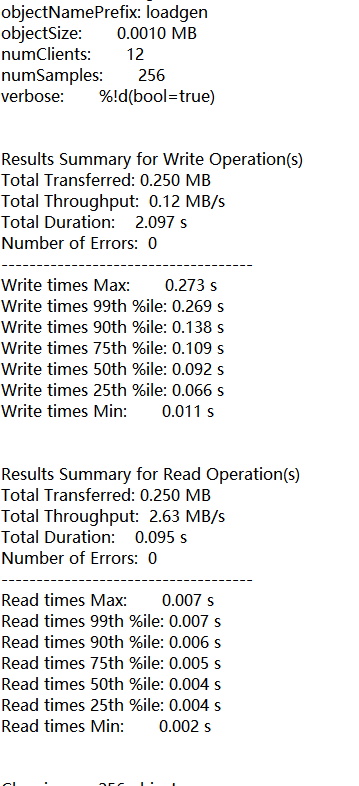


1KB numClients=8 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.14MB/s，总耗时1.838s，99%操作在0.297s内完成，25%操作在0.043s内完成。

读操作：吞吐率为2.51MB/s，总耗时0.099s，99%操作在0.006s内完成，25%操作在0.002s内完成。

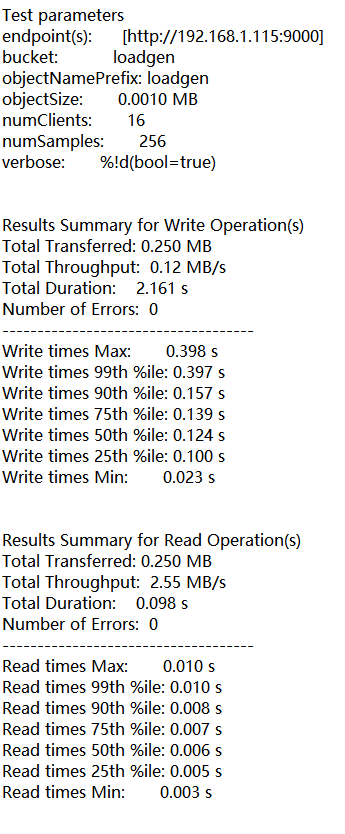


1KB numClients=12 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.12MB/s，总耗时2.097s，99%操作在0.269s内完成，25%操作在0.066s内完成。

读操作：吞吐率为2.63MB/s，总耗时0.095s，99%操作在0.007s内完成，25%操作在0.004s内完成。

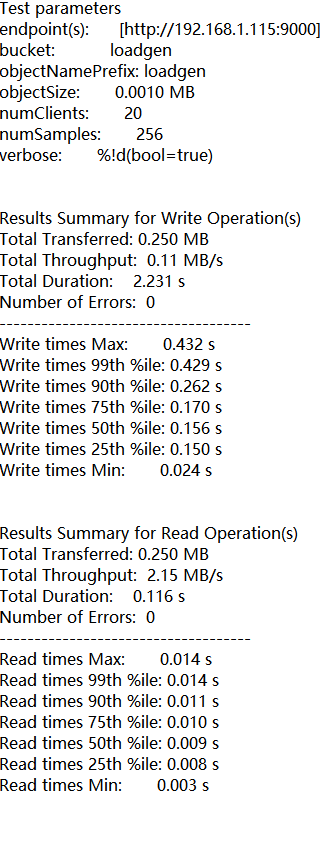


1KB numClients=16 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.12MB/s，总耗时2.161s，99%操作在0.397s内完成，25%操作在0.100s内完成。

读操作：吞吐率为2.55MB/s，总耗时0.098s，99%操作在0.010s内完成，25%操作在0.005s内完成。

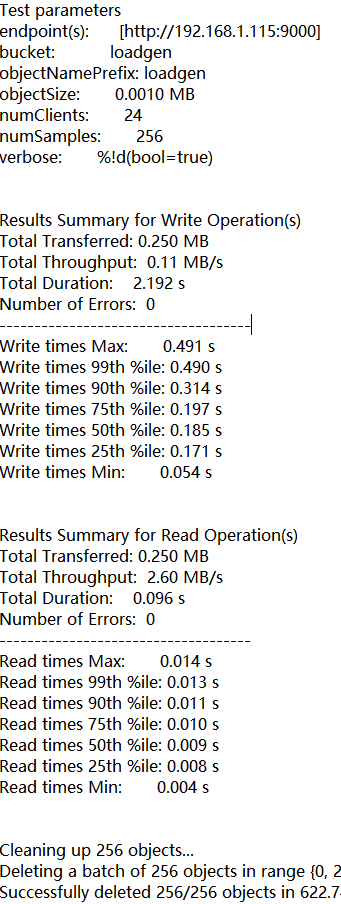


1KB numClients=20 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.11MB/s，总耗时2.231s，99%操作在0.429s内完成，25%操作在0.150s内完成。

读操作：吞吐率为2.15MB/s，总耗时0.116s，99%操作在0.014s内完成，25%操作在0.008s内完成。

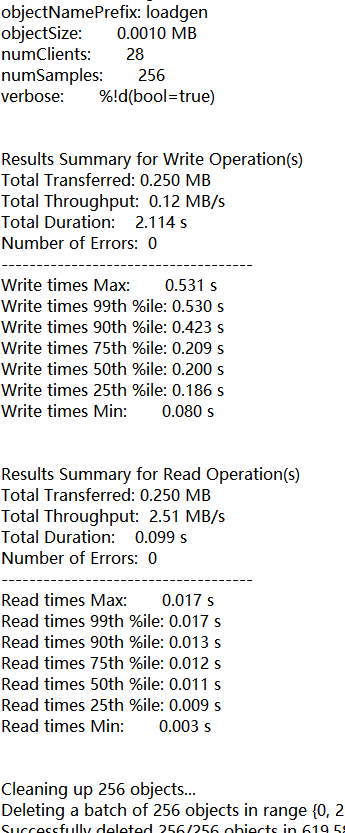


1KB numClients=24 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.11MB/s，总耗时2.192s，99%操作在0.491s内完成，25%操作在0.171s内完成。

读操作：吞吐率为2.60MB/s，总耗时0.096s，99%操作在0.013s内完成，25%操作在0.008s内完成。

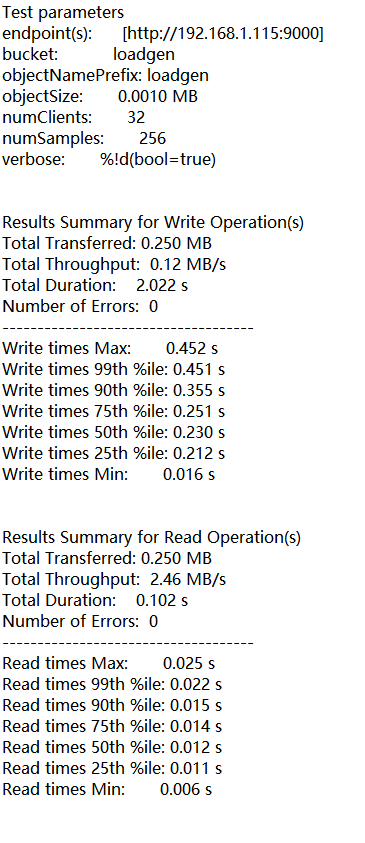


1KB numClients=28 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.12MB/s，总耗时2.114s，99%操作在0.530s内完成，25%操作在0.186s内完成。

读操作：吞吐率为2.51MB/s，总耗时0.099s，99%操作在0.017s内完成，25%操作在0.009s内完成。

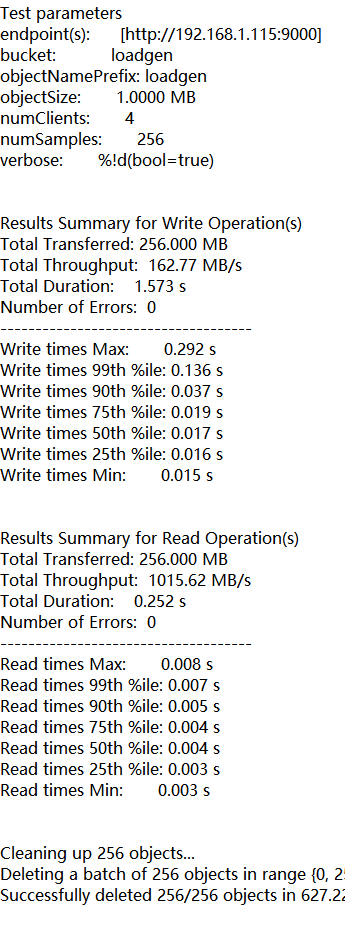


1KB numClients=32 numSamples=256

总文件大小为0.25MB

写操作：吞吐率为0.12MB/s，总耗时2.022s，99%操作在0.451s内完成，25%操作在0.212s内完成。

读操作：吞吐率为2.3MB/s，总耗时0.102s，99%操作在0.022s内完成，25%操作在0.011s内完成。

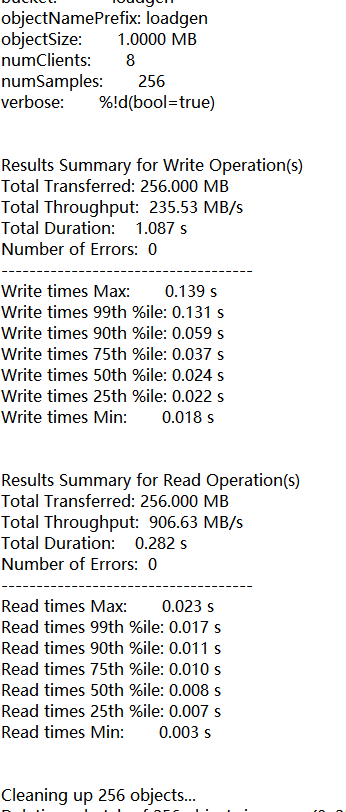


1MB numClients=4 numSamples=256

总文件大小为256MB

写操作：吞吐率为162.77.53MB/s，总耗时1.573s，99%操作在0.136s内完成，25%操作在0.016s内完成。

读操作：吞吐率为1015.62MB/s，总耗时0.252s，99%操作在0.007s内完成，25%操作在0.003s内完成。

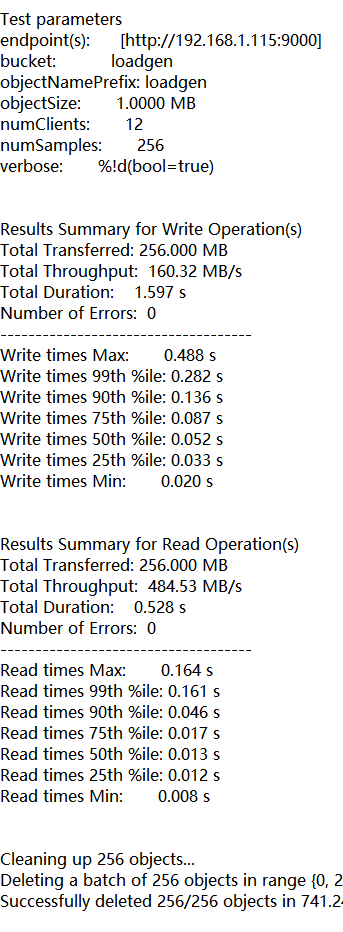


1MB numClients=8 numSamples=256

总文件大小为256MB

写操作：吞吐率为235.53MB/s，总耗时1.087s，99%操作在0.131s内完成，25%操作在0.022s内完成。

读操作：吞吐率为906.63MB/s，总耗时0.282s，99%操作在0.017s内完成，25%操作在0.007s内完成。

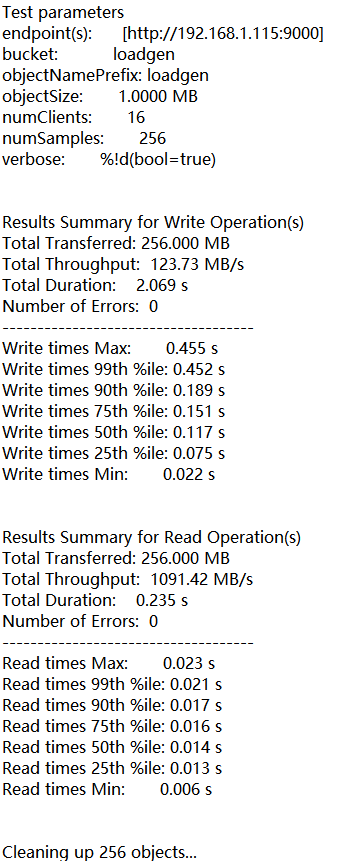


1MB numClients=12 numSamples=256

总文件大小为256MB

写操作：吞吐率为160.32MB/s，总耗时1.597s，99%操作在0.282s内完成，25%操作在0.033s内完成。

读操作：吞吐率为484.53MB/s，总耗时0.528s，99%操作在0.161s内完成，25%操作在0.012s内完成。

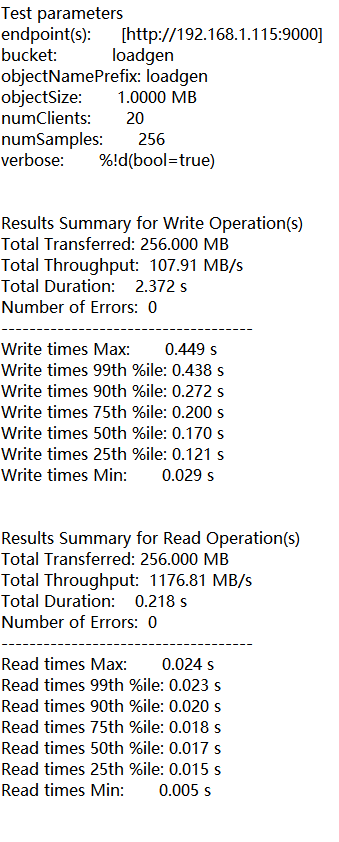


1MB numClients=16 numSamples=256

总文件大小为256MB

写操作：吞吐率为123.73MB/s，总耗时2.069s，99%操作在0.452s内完成，25%操作在0.075s内完成。

读操作：吞吐率为1091.42MB/s，总耗时0.235s，99%操作在0.021s内完成，25%操作在0.013s内完成。

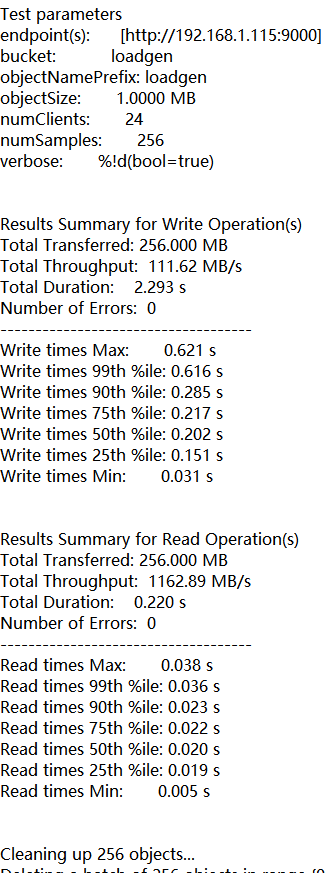


1MB numClients=20 numSamples=256

总文件大小为256MB

写操作：吞吐率为107.91MB/s，总耗时2.372s，99%操作在0.438s内完成，25%操作在0.121s内完成。

读操作：吞吐率为1176.81MB/s，总耗时0.218s，99%操作在0.023s内完成，25%操作在0.015s内完成。

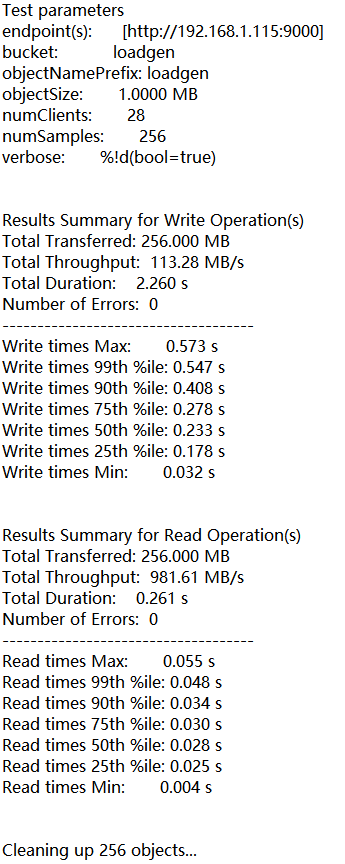


1MB numClients=24 numSamples=256

总文件大小为256MB

写操作：吞吐率为111.62MB/s，总耗时2.293s，99%操作在0.616s内完成，25%操作在0.151s内完成。

读操作：吞吐率为1162.89MB/s，总耗时0.220s，99%操作在0.036s内完成，25%操作在0.019s内完成。

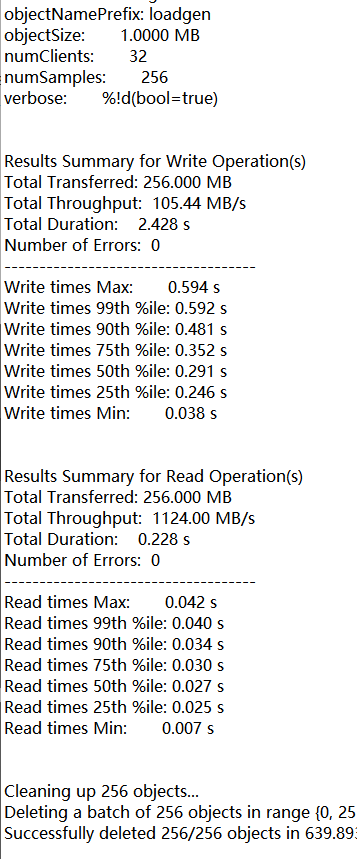


1MB numClients=28 numSamples=256

总文件大小为256MB

写操作：吞吐率为113.28MB/s，总耗时2.260s，99%操作在0.547s内完成，25%操作在0.178s内完成。

读操作：吞吐率为981.61MB/s，总耗时0.261s，99%操作在0.048s内完成，25%操作在0.025s内完成。



1MB numClients=32 numSamples=25

总文件大小为256MB

写操作：吞吐率为105.44MB/s，总耗时2.428s，99%操作在0.592s内完成，25%操作在0.246s内完成。

读操作：吞吐率为1124.00MB/s，总耗时0.228s，99%操作在0.040s内完成，25%操作在0.025s内完成。

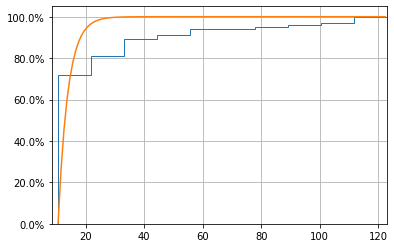
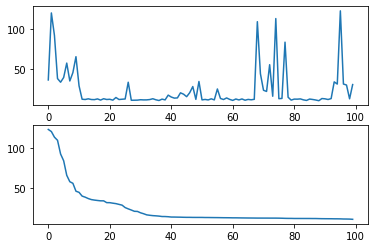
对于对象大小为1KB的实验，并发数在4时出现最好性能，其他并发数延迟较高并区域稳定，可能时由于并发数过多而样本太小造成过多的开销浪费。由图可看出并发数为20的实验发生尾延迟。

对于对象大小为1MB的实验，写性能在并发数为8时表现最好，而后随并发数增加而降低，读性能在并发数为12时表现最好，而后随并发数增加而降低。

观察可得，并发数越大，尾延迟越大。

# 3 尾延迟挑战

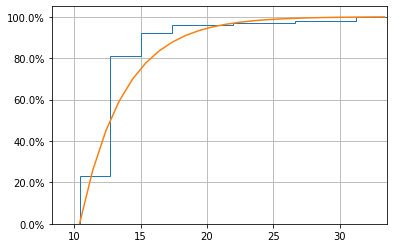
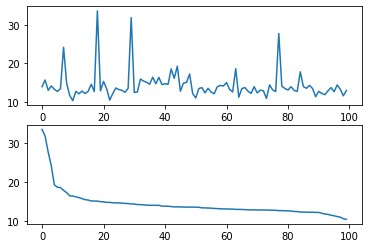
## 3.1 原尾延迟



通过运行latency-collect.ipynb得到原始尾延迟数据，并用latency-plot.ipynb作图。由图可知99%的延迟在110ms内。

## 3.2 对冲请求 **Hedged Request**

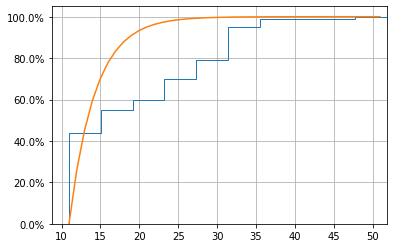
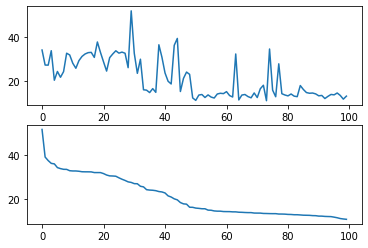
对冲请求对同一资源并发地发送多个请求，以完成最快的请求作为请求的结束。本实验采取两个请求对冲，选取延迟较小的作为结果作图。



与原数据相比尾延迟明显降低，99%的延迟在35ms内.

## 3.3 关联请求Tied Request

关联请求一次只对资源同时发送一个请求，但在请求耗时超过阈值（通常选取95%分位数）时，中断当前请求，重新发送新请求。



与原数据相比尾延迟有较大的改善，99%的延迟在36ms以内。