minio调研

1. 功能特性

1）Amazon S3兼容

Minio使用Amazon S3 v2 / v4 API。可以使用Minio SDK，Minio Client，AWS SDK和AWS CLI访问Minio服务器。

2）数据保护

Minio使用Minio Erasure Code来防止硬件故障。损坏一半的driver，仍然可以从中恢复。

3）高度可用

Minio服务器可以容忍分布式设置中高达（N / 2）-1节点故障。而且，您可以配置Minio服务器在Minio与任意Amazon S3兼容服务器之间存储数据。

4）Lambda计算

Minio服务器通过其兼容AWS SNS / SQS的事件通知服务触发Lambda功能。支持的目标是消息队列，如Kafka，NATS，AMQP，MQTT，Webhooks以及Elasticsearch，Redis，Postgres和MySQL等数据库。

5）加密和防篡改

Minio为加密数据提供了机密性，完整性和真实性保证，而且性能开销微乎其微。使用AES-256-GCM，ChaCha20-Poly1305和AES-CBC支持服务器端和客户端加密。加密的对象使用AEAD服务器端加密进行防篡改。

2. minio核心概念

- Drive：即存储数据的磁盘，在 MinIO 启动时，以参数的方式传入；
- Erasure Code: 纠删码，就是可以通过数学计算，把丢失的数据进行还原，它可以将n份原始数据，增加m份数据，并能通过n+m份中的任意n份数据，还原为原始数据。即如果有任意小于等于m份的数据失效，仍然能通过剩下的数据还原出来；
- Erasure Set：即一组 Drive 的集合，分布式部署根据集群规模自动划分一个或多个 Erasure Set（每个Erasure Set包含4到16个Drive），每个 Erasure Set 中的 Drive 分布在不同位置。一个对象存储在一个 Erasure Set 上；
- Bucket Version：MinIO 支持在单个存储桶中保存对象的多个“版本”。 通常会覆盖现有对象的写入操作 导致创建新的版本化对象。 MinIO 版本控制可防止 意外覆盖和删除，同时支持“撤消” 写操作。 存储桶版本控制还支持保留和存档策略；
- Server Pool: `HOSTNAME` 参数传递给 `minio server` 命令代表一个服务器池:

```
minio server https://minio{1...4}.example.net/mnt/disk{1...4}

             |                    Server Pool                |
```

- Cluster: 整个 MinIO 部署由一个或多个服务器池组成。 每个 `HOSTNAME` 参数传递给 `minio server` 命令代表一个服务器池:

```
minio server https://minio{1...4}.example.net/mnt/disk{1...4} \
             https://minio{5...8}.example.net/mnt/disk{1...4}

             |                    Server Pool                |
```

3. 环境说明

节点名	IP	配置	带宽
node1	192.168.251.133	虚拟机 16c16g 2数据盘80G	20Gbit/s
node2	192.168.251.88	虚拟机 16c16g 2数据盘80G	20Gbit/s
node3	192.168.251.98	虚拟机 16c16g 2数据盘80G	20Gbit/s
压力机	192.168.251.153	虚拟机 16c16g	20Gbit/s

4. 部署说明

下载地址：

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

minio版本：

```
[root@localhost minio]# ./minio --version
minio version RELEASE.2022-06-11T19-55-32Z
commit: dd53b287f2eeed9cd3872eeae7d64696bfd7829d
go version: go1.18.3
```

分布式部署架构

分布式部署，默认使用纠删码模式进行数据的保护，最少需要 四块磁盘进行纠删码模式的部署，最高可以允许（N/2）一半的磁盘故障。

测试环境三个节点每个节点两块盘，总共六个 drive（测试只使用了drive1）

```
[root@node1 ~]# ls /data*
/data1:
drive1  drive2

/data2:
drive1  drive2
```

目录结构：

```
[root@node1 opt]# tree /opt/
/opt/
└── minio
    ├── config       // 配置文件目录
    │   └── certs
    │       └── CAs
    ├── minio       // minio二进制
    └── run.sh       //启动脚本
```

启动脚本：

```
#!/bin/bash
export MINIO_ROOT_USER=admin
export MINIO_ROOT_PASSWORD=edoc2@edoc2

/opt/minio/minio server --config-dir /opt/minio/config/ --console-address
":30000"\
    http://192.168.251.133/data{1..2}/drive1 \
    http://192.168.251.88/data{1..2}/drive1 \
    http://192.168.251.98/data{1..2}/drive1
```

service 文件:

```
[root@node1 opt]# cat /usr/lib/systemd/system/minio.service
[Unit]
Description=Minio service
Documentation=https://docs.minio.io/

[Service]
WorkingDirectory=/opt/minio/
ExecStart=/opt/minio/run.sh
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

服务进程

配置nginx代理（非必须）

```
upstream minio{
    server 192.168.251.133:9000;
    server 192.168.251.88:9000;
    server 192.168.251.98:9000;

}
server {
    listen 9300;
    server_name minio;
    location / {
        proxy_pass http://minio;
        proxy_set_header Host $http_host;
        client_max_body_size 1000m;
    }
}
```

配置minio 客户端 mc

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

添加配置 minio的连接信息

```
mc config host add minio http://192.168.251.189:9300 admin edoc2@edoc2 --api
s3v4
## 这里的9300 是nginx的地址，代理到三个节点的9000
```

单机部署

```
[root@node1 minio]# cat run.sh
#!/bin/bash

export MINIO_ROOT_USER=admin
export MINIO_ROOT_PASSWORD=edoc2@edoc2

# 指定一个存储目录
/opt/minio/minio server --config-dir /opt/minio/config/ --console-address
":30000"\
    /data1/drive1
```

单机部署无法提供数据的安全保障；

测试环境使用192.168.251.133这个机器部署测试。

NAS网关部署

NAS网关在最新版本的minio中此功能已经移除，不再进行支持；

测试使用的之前版本（minio.20210116 version RELEASE.2021-01-16T02-19-44Z）

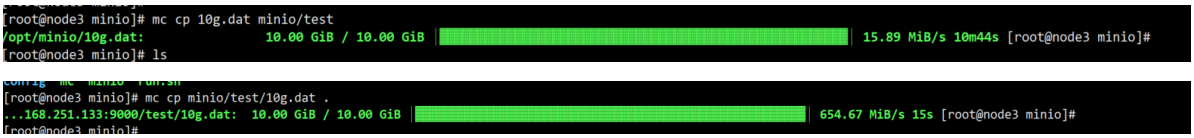
在192.168.251.88 NFS 共享文件目录到 192.168.251.133 进行 NAS网关的测试。


```
[root@node1 minio]# cat run.sh
#!/bin/bash

export MINIO_ROOT_USER=admin
export MINIO_ROOT_PASSWORD=edoc2@edoc2

# /data1/share 是从 192.168.251.88 共享出来的目录
/opt/minio/minio.20210116 gateway nas /data1/share
```

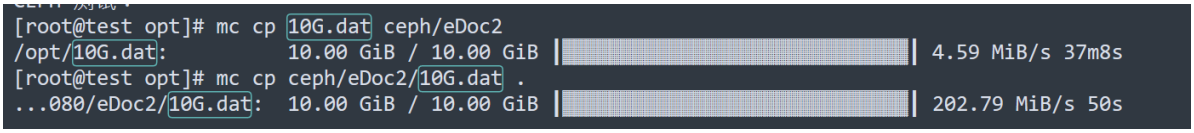
5. 单文件上传测试

使用minio官方提供的s3客户端工具mc在压力机192.168.251.153进行文件上传下载测试

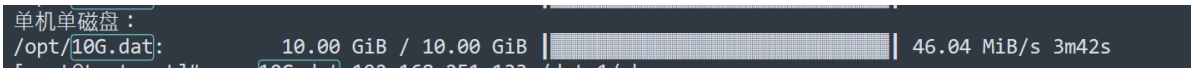
minio分布式集群上传下载



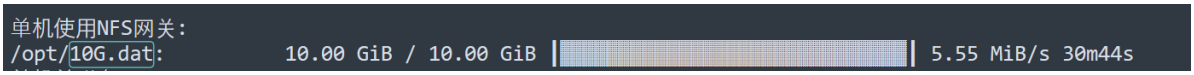
ceph集群



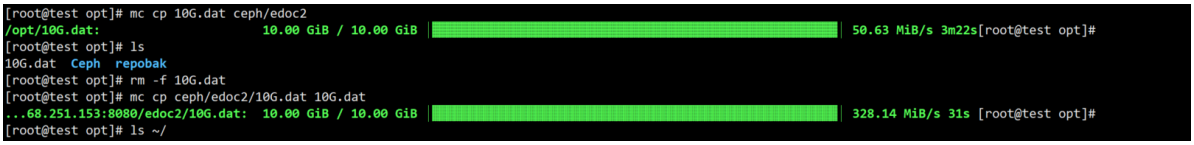
minio单机



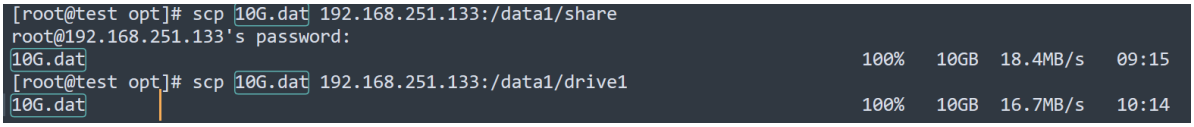
minio单机NAS网关



ceph单机



SCP拷贝



存储部署类别	上传速度	下载速度
minio分布式集群	15.89MiB/s 10m44s	654.67MiB/s 15s
ceph集群	4.59MiB/s 37m8s	202.79MiBs 50s
minio单机单磁盘	46.04 MiB/s 3m42s	-
minio NAS网关	5.55 MiB/s 30m44s	-
ceph 单机	50.63 MiB/s 3m22s	328.14 MiB/s 31s
SCP拷贝（网络盘/本地盘）	（18.4MB/s 09:15/16.7MB/s 10:14）	-

总体上大文件上传，minio的上传下载性能集群上对ceph集群有4倍左右优势，minio的集群上传下载性能基本与本机SCP拷贝持平。使用NAS网关，和minio单机部署方式比较，上传速度衰减明显。

6. 压力测试

压测工具使用是minio的官方提供的压测工具 warp，压力机地址192.168.251.153；

```
# 压力机执行
warp mixed --host=192.168.251.133:9000,192.168.251.88:9000,192.168.251.98:9000 --access-key admin --secret-key edoc2@edoc2 --autoterm  --analyze.v

## 默认的 混合压力测试，默认生成2500个对象，每个对象大小10MB，并发20，进行上传、下载、删除、查看状态操作
```

压测方法：分别部署minio分布式集群，minio单机，minio NAS 网关，ceph集群这几种部署方式，分别进行对象大小100Kib，1Mib，5Mib，10Mib的总存储量大概20G左右的 20并发的mixed混合压力测试

压测记录： https://v5.edoc2.com/preview.html?fileid=3684760

压测数据整理： https://v5.edoc2.com/preview.html?fileid=3684739

minio集群与ceph集群

minio分布式集群使用的三个节点6个磁盘作为一个set进行测试；

ceph集群使用的三副本，每个节点两块HDD磁盘作为osd进行部署测试。

存储部署类别	对象数量	对象大小	操作类型	操作数量（比例）	Throughput by host	Requests time				Cluster Total
						Avg	50%	90%	99%	
minio分布式集群	200000	100Kib	DELETE	2359, 10%	• http://192.168.251.98:9000: Avg: 3.17 obj/s. • http://192.168.251.133:9000: Avg: 3.07 obj/s. • http://192.168.251.88:9000: Avg: 3.14 obj/s.	302ms	85ms	261ms	3.094s	5.46 MiB/s, 93.19 obj/s over 4m14s.
			GET	10654, 45.0%	• http://192.168.251.98:9000: Avg: 1.42 MiB/s, 14.51 obj/s. • http://192.168.251.88:9000: Avg: 1.36 MiB/s, 13.91 obj/s. • http://192.168.251.133:9000: Avg: 1.36 MiB/s, 13.90 obj/s.	184ms	40ms	117ms	1.96s	
			PUT	3537, 14.9%	• http://192.168.251.88:9000: Avg: 0.46 MiB/s, 4.71 obj/s. • http://192.168.251.98:9000: Avg: 0.46 MiB/s, 4.68 obj/s. • http://192.168.251.133:9000: Avg: 0.46 MiB/s, 4.70 obj/s.	473ms	112ms	385ms	5.655s	
			STAT	7105, 30.0%	• http://192.168.251.98:9000: Avg: 9.54 obj/s. • http://192.168.251.133:9000: Avg: 9.34 obj/s. • http://192.168.251.88:9000: Avg: 9.34 obj/s.	96ms	24ms	79ms	1.142s	
	20000	1Mib	DELETE	1399, 10.0%	• http://192.168.251.88:9000: Avg: 1.88 obj/s. • http://192.168.251.98:9000: Avg: 1.63 obj/s. • http://192.168.251.133:9000: Avg: 1.66 obj/s.	241ms	41ms	578ms	2.946s	31.06 MiB/s, 51.77 obj/s over 4m31s.
			GET	6307, 45.0%	• http://192.168.251.133:9000: Avg: 7.41 MiB/s, 7.41 obj/s. • http://192.168.251.88:9000: Avg: 7.94 MiB/s, 7.94 obj/s. • http://192.168.251.98:9000: Avg: 7.93 MiB/s, 7.93 obj/s.	334ms	43ms	613ms	4.804s	
			PUT	2084, 14.9%	• http://192.168.251.133:9000: Avg: 2.70 MiB/s, 2.70 obj/s. • http://192.168.251.88:9000: Avg: 2.47 MiB/s, 2.47 obj/s. • http://192.168.251.98:9000: Avg: 2.56 MiB/s, 2.56 obj/s.	1.266s	267ms	2.521s	19.674s	
			STAT	4209, 30.0%	• http://192.168.251.88:9000: Avg: 5.11 obj/s. • http://192.168.251.133:9000: Avg: 5.06 obj/s. • http://192.168.251.98:9000: Avg: 5.36 obj/s.	75ms	8ms	52ms	1.389s	
	4000	5Mib	DELETE	446, 9.9%	• http://192.168.251.133:9000: Avg: 0.56 obj/s. • http://192.168.251.88:9000: Avg: 0.50 obj/s. • http://192.168.251.98:9000: Avg: 0.53 obj/s.	566ms	256ms	1.198s	6.822s	46.50 MiB/s, 15.49 obj/s over 4m51s.
			GET	2024, 45.0%	• http://192.168.251.133:9000: Avg: 12.36 MiB/s, 2.47 obj/s. • http://192.168.251.88:9000: Avg: 11.88 MiB/s, 2.38 obj/s. • http://192.168.251.98:9000: Avg: 11.15 MiB/s, 2.23 obj/s.	1.387s	722ms	2.573s	13.637s	
			PUT	661, 14.7%	• http://192.168.251.98:9000: Avg: 4.03 MiB/s, 0.81 obj/s. • http://192.168.251.133:9000: Avg: 3.80 MiB/s, 0.76 obj/s. • http://192.168.251.88:9000: Avg: 3.93 MiB/s, 0.79 obj/s.	3.567s	2.462s	7.371s	19.893s	
			STAT	1347, 30.0%	• http://192.168.251.88:9000: Avg: 1.50 obj/s. • http://192.168.251.133:9000: Avg: 1.58 obj/s. • http://192.168.251.98:9000: Avg: 1.61 obj/s.	176ms	25ms	349ms	2.84s	
	2500	10Mib	DELETE	128, 10.2%	• http://192.168.251.88:9000: Avg: 0.14 obj/s. • http://192.168.251.98:9000: Avg: 0.19 obj/s. • http://192.168.251.133:9000: Avg: 0.21 obj/s.	1.364s	147ms	2.681s	29.592s	29.04 MiB/s, 4.98 obj/s over 4m13s
			GET	546, 43.4%	• http://192.168.251.133:9000: Avg: 6.73 MiB/s, 0.67 obj/s. • http://192.168.251.98:9000: Avg: 8.15 MiB/s, 0.81 obj/s. • http://192.168.251.88:9000: Avg: 7.39 MiB/s, 0.74 obj/s.	3.508s	979ms	9.913s	37.978s	
			PUT	170, 13.5%	• http://192.168.251.133:9000: Avg: 2.83 MiB/s, 0.28 obj/s. • http://192.168.251.98:9000: Avg: 2.15 MiB/s, 0.21 obj/s. • http://192.168.251.88:9000: Avg: 2.35 MiB/s, 0.23 obj/s.	15.788s	4.96s	48.486s	1m34.202s	
			STAT	392, 31.2%	• http://192.168.251.133:9000: Avg: 0.53 obj/s. • http://192.168.251.98:9000: Avg: 0.49 obj/s. • http://192.168.251.88:9000: Avg: 0.54 obj/s.	245ms	8ms	199ms	7.835s	

ceph集群	200000	100Kib	DELETE	3746, 10.0%	* http://192.168.251.133:8080: Avg: 4.97 obj/s. * http://192.168.251.88:8080: Avg: 5.27 obj/s. * http://192.168.251.98:8080: Avg: 5.13 obj/s.	302ms	117ms	621ms	3.669s	9.00 MiB/s, 153.54 obj/s over 4m5s.
			GET	16906, 45.0%	* http://192.168.251.133:8080: Avg: 2.23 MiB/s, 22.80 obj/s. * http://192.168.251.98:8080: Avg: 2.26 MiB/s, 23.15 obj/s. * http://192.168.251.88:8080: Avg: 2.27 MiB/s, 23.21 obj/s.	85ms	14ms	105ms	1.569s	
			PUT	5625, 15.0%	* http://192.168.251.133:8080: Avg: 0.76 MiB/s, 7.83 obj/s. * http://192.168.251.98:8080: Avg: 0.74 MiB/s, 7.59 obj/s. * http://192.168.251.88:8080: Avg: 0.74 MiB/s, 7.63 obj/s.	377ms	141ms	904ms	4.25s	
			STAT	11274, 30.0%	* http://192.168.251.88:8080: Avg: 15.17 obj/s. * http://192.168.251.133:8080: Avg: 15.54 obj/s. * http://192.168.251.98:8080: Avg: 15.39 obj/s.	17ms	2ms	12ms	284ms	
	20000	1Mib	DELETE	1576, 10.0%	* http://192.168.251.88:8080: Avg: 1.88 obj/s. * http://192.168.251.133:8080: Avg: 1.81 obj/s. * http://192.168.251.98:8080: Avg: 1.97 obj/s.	856ms	242ms	2.014s	9.124s	32.87 MiB/s, 54.78 obj/s over 4m49s.
			GET	7134, 45.0%	* http://192.168.251.88:8080: Avg: 8.21 MiB/s, 8.21 obj/s. * http://192.168.251.98:8080: Avg: 8.17 MiB/s, 8.17 obj/s. * http://192.168.251.133:8080: Avg: 8.29 MiB/s, 8.29 obj/s.	150ms	16ms	232ms	3.072s	
			PUT	2354, 14.9%	* http://192.168.251.133:8080: Avg: 2.83 MiB/s, 2.83 obj/s. * http://192.168.251.88:8080: Avg: 2.83 MiB/s, 2.83 obj/s. * http://192.168.251.98:8080: Avg: 2.80 MiB/s, 2.80 obj/s.	1.298s	401ms	3.971s	11.567s	
			STAT	4751, 30.0%	* http://192.168.251.88:8080: Avg: 5.87 obj/s. * http://192.168.251.98:8080: Avg: 5.32 obj/s. * http://192.168.251.133:8080: Avg: 5.33 obj/s.	30ms	3ms	8ms	524ms	
	4000	5Mib	DELETE	135, 10.3%	* http://192.168.251.88:8080: Avg: 0.19 obj/s. * http://192.168.251.98:8080: Avg: 0.26 obj/s. * http://192.168.251.133:8080: Avg: 0.24 obj/s.	4.809s	1.401s	15.709s	28.761s	17.36 MiB/s, 5.91 obj/s over 3m43s.
			GET	576, 44.0%	* http://192.168.251.133:8080: Avg: 4.46 MiB/s, 0.89 obj/s. * http://192.168.251.98:8080: Avg: 4.82 MiB/s, 0.96 obj/s. * http://192.168.251.88:8080: Avg: 4.48 MiB/s, 0.90 obj/s.	2.359s	449ms	8.037s	24.129s	
			PUT	184, 14.1%	* http://192.168.251.133:8080: Avg: 1.59 MiB/s, 0.32 obj/s. * http://192.168.251.98:8080: Avg: 1.48 MiB/s, 0.30 obj/s. * http://192.168.251.88:8080: Avg: 1.67 MiB/s, 0.33 obj/s.	9.922s	4.136s	27.981s	37.37s	
			STAT	397, 30.4%	* http://192.168.251.133:8080: Avg: 0.63 obj/s. * http://192.168.251.88:8080: Avg: 0.56 obj/s. * http://192.168.251.98:8080: Avg: 0.68 obj/s.	352ms	3ms	470ms	9.883s	
	2500	10Mib	DELETE	56, 8.8%	* http://192.168.251.133:8080: Avg: 0.11 obj/s. * http://192.168.251.88:8080: Avg: 0.09 obj/s. * http://192.168.251.98:8080: Avg: 0.08 obj/s.	10.826s	5.125s	28.682s	56.312s	16.66 MiB/s, 2.83 obj/s over 3m48s.
			GET	284, 44.4%	* http://192.168.251.88:8080: Avg: 3.65 MiB/s, 0.37 obj/s. * http://192.168.251.133:8080: Avg: 4.21 MiB/s, 0.42 obj/s. * http://192.168.251.98:8080: Avg: 4.62 MiB/s, 0.46 obj/s.	7.258s	1.704s	23.272s	47.531s	
			PUT	81, 12.7%	* http://192.168.251.98:8080: Avg: 1.33 MiB/s, 0.13 obj/s. * http://192.168.251.88:8080: Avg: 1.22 MiB/s, 0.12 obj/s. * http://192.168.251.133:8080: Avg: 1.28 MiB/s, 0.13 obj/s.	17.684s	3.881s	53.462s	1m30.088s	
			STAT	202, 31.6%	* http://192.168.251.88:8080: Avg: 0.30 obj/s. * http://192.168.251.133:8080: Avg: 0.33 obj/s. * http://192.168.251.98:8080: Avg: 0.26 obj/s.	1.186s	6ms	1.768s	25.915s	

从上面测试数据可以看出，minio集群在压测100Kib小文件的时候，吞吐量弱于ceph集群100Kib的压测；在大文件对象测试中好于ceph集群。

从压测过程中系统资源占用情况，minio的三个节点的CPU负载一直持续在30左右浮动，而ceph集群的CPU负载在2左右，由于minio使用的是纠删码的方式保证数据安全，需要大量计算，CPU成为minio的性能瓶颈点。

minio单机与minio NAS 网关

minio单机	200000	100Kib	DELETE	20774, 10.0%	69.56 obj/s	25ms	3ms	11ms	272ms	40.60 MiB/s, 692.97 obj/s over 5m0s.
			GET	93480, 45.0%	30.56 MiB/s, 312.96 obj/s	15ms	3ms	8ms	192ms	
			PUT	31141, 15.0%	10.19 MiB/s, 104.30 obj/s	115ms	8ms	84ms	1.706s	
			STAT	62325, 30.0%	208.66 obj/s	7ms	2ms	3ms	69ms	
	20000	1Mib	DELETE	8904, 10.0%	29.77 obj/s	19ms	4ms	31ms	247ms	178.62 MiB/s, 297.72 obj/s over 4m59s.
			GET	40090, 45.0%	134.00 MiB/s, 134.00 obj/s	29ms	9ms	53ms	317ms	
			PUT	13340, 15.0%	44.63 MiB/s, 44.63 obj/s	334ms	152ms	540ms	3.893s	
			STAT	26722, 30.0%	89.33 obj/s	7ms	2ms	6ms	83ms	
	4000	5Mib	DELETE	417, 10.0%	1.53 obj/s	236ms	5ms	416ms	4.96s	44.08 MiB/s, 14.70 obj/s over 4m44s.
			GET	1871, 44.9%	34.09 MiB/s, 6.82 obj/s	342ms	37ms	817ms	5.442s	
			PUT	612, 14.7%	11.39 MiB/s, 2.28 obj/s	6.731s	2.069s	21.728s	29.822s	
			STAT	1249, 30.0%	4.45 obj/s	51ms	2ms	6ms	1.47s	
	2500	10Mib	DELETE	289, 10.0%	1.21 obj/s	144ms	9ms	161ms	3.175s	70.42 MiB/s, 11.83 obj/s over 4m4s.
			GET	1282, 44.5%	54.15 MiB/s, 5.41 obj/s	680ms	163ms	922ms	11.054s	
			PUT	403, 14.0%	17.39 MiB/s, 1.74 obj/s	9.401s	1.206s	42.211s	58.202s	
			STAT	872, 30.3%	3.65 obj/s	25ms	3ms	15ms	199ms	
minio NAS 网关	200000	100Kib	DELETE	1464, 9.9%	19.88 obj/s	144ms	134ms	195ms	267ms	11.59 MiB/s, 198.01 obj/s over 1m14s.
			GET	6627, 45.0%	8.71 MiB/s, 89.14 obj/s	90ms	86ms	118ms	148ms	
			PUT	2196, 14.9%	2.90 MiB/s, 29.67 obj/s	213ms	205ms	266ms	344ms	
			STAT	4425, 30.0%	59.53 obj/s	46ms	44ms	61ms	77ms	
	20000	1Mib	DELETE	1749, 10.0%	6.10 obj/s	390ms	185ms	621ms	4.882s	35.95 MiB/s, 59.89 obj/s over 4m53s.
			GET	7892, 45.0%	27.01 MiB/s, 27.01 obj/s	217ms	112ms	206ms	2.619s	
			PUT	2620, 14.9%	9.17 MiB/s, 9.17 obj/s	1.066s	354ms	3.094s	10.113s	
			STAT	5255, 30.0%	17.96 obj/s	109ms	57ms	87ms	1.342s	
	4000	5Mib	DELETE	563, 10.0%	2.13 obj/s	971ms	286ms	2.518s	9.759s	63.31 MiB/s, 21.16 obj/s over 4m27s.
			GET	2539, 45.0%	47.66 MiB/s, 9.53 obj/s	506ms	139ms	1.171s	6.044s	
			PUT	824, 14.6%	16.60 MiB/s, 3.32 obj/s	3.505s	1.381s	12.039s	28.721s	
			STAT	1696, 30.1%	6.37 obj/s	250ms	65ms	481ms	3.287s	
	2500	10Mib	DELETE	126, 9.6%	0.67 obj/s	2.056s	348ms	8.932s	13.503s	37.22 MiB/s, 6.34 obj/s over 3m28s.
			GET	590, 44.9%	28.58 MiB/s, 2.86 obj/s	1.344s	210ms	5.616s	9.532s	
			PUT	169, 12.9%	8.58 MiB/s, 0.86 obj/s	14.997s	2.362s	47.132s	1m5.6s	
			STAT	412, 31.4%	2.00 obj/s	688ms	102ms	2.692s	6.097s	

minio 通过网关对共享存储对外提供s3接口的性能相较于minio单机有性能衰减。

7. 总结

1. minio对外只提供s3的标准接口，只有一个单一的二进制文件，部署起来极其简单，很容易进行容器化；没有ceph那么多的概念和组件，对实施和运维十分便捷；
2. 由于minio是通过纠删码来进行数据的安全保障的，每个对象都要进行纠删码的计算，对集群的CPU要求会比较高，这也决定了minio最好单独部署为存储集群，不适合与应用进行混布；minio默认在一半磁盘故障的情况下依然数据可用，数据存储大小是实际数据的两倍，减少磁盘空间占用；

3. 在当前测试环境的配置看，minio集群相对于ceph集群看，在大对象上有性能优势，由于小文件压测的数量较多，瓶颈在于测试环境CPU的性能，小文件对于ceph集群没有较大优势；
4. minio分布式集群，官方建议最大部署节点为32个节点，受限于其内部实现的分布式锁Dsync的性能，官方给的扩容方案是建立新的集群，进行联邦扩容；
5. ceph集群在出现问题的时候，有比较强的自愈恢复能力，minio在故障情况下如何处理还需要验证。