

1. 客户端工具和接口

|  |  |  |  |
| --- | --- | --- | --- |
| 客户端 | 存储支持接口 | 优劣点和技术点 | 推荐使用场景 |
| rsync，ftp | s3fs(cephfs) | 客户操作简单自动化 ，性能受限，需要文件系统接口支持，可以利用s3fs挂在s3接口或者cephfs | windows，linux下都有的同步工具，适合小客户，量级不大 |
| s3sync, s3 browser | S3 | 较为通用的S3客户端，对接S3云存储标准接口，对于大文件能够并发上传，提高传输速率 | 类似于rsync，后续可推荐老客户使用此种接口，相比传统工具在性能和容错性上更好 |
| web客户端 | S3 | 适合客户浏览器访问，需要进行web开发 | 适合浏览器访问 |
| 自定义api | S3自定义接口 | 效率和性能高，管理简单 | 直播录制 |

1. s3fs

s3fs(goofs)是利用实现fuse文件系统接口将s3接口挂载成本地目录，对于本地目录的操作转化为对于s3接口的操作

1. proxy

proxy负责接收客户端的操作请求，并转化为对于底层存储的io操作，当io操作成功，生成操作日志保存到kafka中用于集群间同步

proxy起到的作用：大小文件，数据和元数据分离，小文件和元数据保存到hbase中，大文件保存到ceph中



Proxy接口：

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Status** | **Remarks** | **排期** |
| **List Buckets** | Supported | 列出一个用户下所有的bucket | 一期 |
| **Delete Bucket** | Supported |  | 一期 |
| **Create Bucket** | Supported | Different set of canned ACLs | 一期 |
| **Bucket Lifecycle** | Not Supported |  | 二期 |
| **Policy (Buckets, Objects)** | Not Supported | ACLs are supported | 二期 |
| **Bucket Website** | Not Supported |  |  |
| **Bucket ACLs (Get, Put)** | Supported | Different set of canned ACLs |  |
| **Bucket Location** | Supported |  |  |
| **Bucket Notification** | Not Supported |  |  |
| **Bucket Object Versions** | Supported |  |  |
| **Get Bucket Info (HEAD)** | Supported |  | 一期 |
| **Bucket Request Payment** | Supported |  |  |
| **Put Object** | Supported |  | 一期 |
| **Delete Object** | Supported |  | 一期 |
| **Get Object** | Supported |  | 一期 |
| **Object ACLs (Get, Put)** | Supported |  |  |
| **Get Object Info (HEAD)** | Supported |  | 一期 |
| **POST Object** | Supported |  |  |
| **Copy Object** | Supported |  |  |
| **Multipart Uploads** | Supported | (missing Copy Part) | 一期 |

公共请求头部：

| **Request Header** | **Description** |
| --- | --- |
| CONTENT\_LENGTH | Length of the request body. |
| DATE | Request time and date (in UTC). |
| HOST | The name of the host server. |
| AUTHORIZATION | Authorization token. |

响应状态码

|  |  |
| --- | --- |
| **HTTP Status** | **Response Code** |
| 100 | Continue |
| 200 | Success |
| 201 | Created |
| 202 | Accepted |
| 204 | NoContent |
| 206 | Partial content |
| 304 | NotModified |
| 400 | InvalidArgument |
| 400 | InvalidDigest |
| 400 | BadDigest |
| 400 | InvalidBucketName |
| 400 | InvalidObjectName |
| 400 | UnresolvableGrantByEmailAddress |
| 400 | InvalidPart |
| 400 | InvalidPartOrder |
| 400 | RequestTimeout |
| 400 | EntityTooLarge |
| 403 | AccessDenied |
| 403 | UserSuspended |
| 403 | RequestTimeTooSkewed |
| 404 | NoSuchKey |
| 404 | NoSuchBucket |
| 404 | NoSuchUpload |
| 405 | MethodNotAllowed |
| 408 | RequestTimeout |
| 409 | BucketAlreadyExists |
| 409 | BucketNotEmpty |
| 411 | MissingContentLength |
| 412 | PreconditionFailed |
| 416 | InvalidRange |
| 422 | UnprocessableEntity |
| 500 | InternalError |

1. 鉴权

Accessid和secrectkey生成规则：

见ceph rgw RGWAccessKeyPool::generate\_key函数：

创建一个user会创建对应的Accessid和secrectkey：

radosgw-admin user create --uid="dnion" --display-name="Dnion Storage"

目前S3鉴权有V2和V4两个版本，客户端可能都用，两个版本都要支持，可以优先支持V2:

<http://docs.aws.amazon.com/zh_cn/AmazonS3/latest/dev/auth-request-sig-v2.html>

<http://docs.aws.amazon.com/zh_cn/AmazonS3/latest/API/sig-v4-authenticating-requests.html>

鉴权流程：

proxy根据accessid向user表查询对应user记录

response = requests.get('http://stor-server-1:8080/user/6NT1YICUD20NBUY6T9ZE/', headers={"Accept" : "application/json"})

得到json格式（也可以是xml格式）的响应，每一项被base64编码过

base64.b64decode解码取得secrectkey对应的值

根据accessid和secretkey得到签名auth\_sign，与请求中的签名比较，相同则验证通过

1. List buckets

GET /bucketname/object HTTP/1.1

User-Agent: S3 Browser 5-9-7 http://s3browser.com

Authorization: AWS 860N2ABT729I1TQIPNVC:iUYxncIumNcgMhEFwWPHpeZ3OSU=

x-amz-date: Wed, 22 Jun 2016 03:18:00 GMT

Host: 192.168.137.128

Connection: Keep-Alive

HTTP/1.1 403 Forbidden

x-amz-request-id: tx00000000000000000000f-00576a724f-d351-default

Content-Length: 193

Accept-Ranges: bytes

Content-Type: application/xml

Date: Wed, 22 Jun 2016 11:11:11 GMT

Connection: Keep-Alive

<?xml version="1.0" encoding="UTF-8"?><Error><Code>RequestTimeTooSkewed</Code><RequestId>tx00000000000000000000f-00576a724f-d351-default</RequestId><HostId>d351-default-default</HostId></Error>

HTTP/1.1 200 OK

x-amz-request-id: tx000000000000000000010-00576a7250-d351-default

Content-Type: application/xml

Content-Length: 401

Date: Wed, 22 Jun 2016 11:11:12 GMT

Connection: Keep-Alive

<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>tc</ID><DisplayName>tc server</DisplayName></Owner><Buckets><Bucket><Name>newbucket</Name><CreationDate>2016-06-22T10:44:39.453Z</CreationDate></Bucket><Bucket><Name>testbucket</Name><CreationDate>2016-06-16T19:49:17.795Z</CreationDate></Bucket></Buckets></ListAllMyBuckets Result>

1. rgw和rados

rgw负责接收s3请求，前期可以考虑直接proxy转发s3请求给rgw，后续proxy对于s3请求开发完善则考虑直接对接rados提升性能

1. hbase

hbase负责存储元数据和小文件，为了加快相应速度，考虑需要加入radis等内存中间件加速

包括三大类信息：

a. 配置元信息，用户，频道的映射关系，用户配额，空间统计，权限信息，过期，是否异地备份等等

b. 文件自身元信息，包括大小文件区分，文件md5，过期时间，文件大小等等

c. 小文件的数据，对于proxy判定小于一定阈值的文件直接将数据存储到hbase中

1. 用户元数据  
   (accesskey,  [uid, secretkey])

(uid,  [username, email, quota(max\_size, max\_files), region(erea1, erea2, ...),])

    (uid,  [user\_stat(used\_size, used\_files)])

2. bucket元数据

     uid\_bucket , bucket\_quota(max\_size, max\_files), bucket\_stat(used\_size, used\_files)

1. 对象元数据

url, length, md5

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Ceph处理流程** | **Proxy流程** | **Hbase存储方案** |
| **List Buckets** | Accessid->uid.buckets->在default.rgw.users.uidpool中列出uid.buckets所有的omap | 根据accessid查询user表，鉴权  根据userid为前缀遍历bucket表，获取匹配的bucket信息  将获取到的bucket概要信息组装成响应返回给客户端 | 一个user table，存储（Accessid，userinfo）映射关系  一个bucket table，以userid为前缀，（userid\_bucketname,bucket\_info）映射关系 |
| **Delete Bucket** | Supported | 根据accessid查询user表，鉴权  以userid为前缀，对bucket表中  （userid\_bucketname,bucket\_info）记录存放到一个删除表中  更新user table中对应user的stat信息 | 一个delete table，以userid为前缀，（userid\_bucketname,bucket\_info）映射关系 |
| **Create Bucket** | Supported | 根据accessid查询user表，鉴权  以userid为前缀，（userid\_bucketname,bucket\_info）映射关系插入一条记录到bucket table | 一个user table，存储（Accessid，user）映射关系  一个bucket table，以userid为前缀，（userid\_bucketname,bucket\_info）映射关系 |
| **Bucket Lifecycle** | Not Supported |  |  |
| **Policy (Buckets, Objects)** | Not Supported |  | ACLs are supported |
| **Bucket Website** | Not Supported |  |  |
| **Bucket ACLs (Get, Put)** | Supported |  | Different set of canned ACLs |
| **Bucket Location** | Supported |  |  |
| **Bucket Notification** | Not Supported |  |  |
| **Bucket Object Versions** | Supported |  |  |
| **Get Bucket Info (HEAD)** | Supported |  |  |
| **Bucket Request Payment** | Supported |  |  |
| **Put Object** | Supported | 根据accessid查询user表，鉴权  根据userid\_bucketname查询bucket表，获取匹配的bucket信息  根据bucket信息判断是否可以写入  写入对象数据  写入成功后将（obj\_uri，object\_info）写入object表  更新bucket统计和user统计信息 | 一个object table，存储（obj\_uri，object\_info）映射关系 |
| **Delete Object** | Supported |  |  |
| **Get Object** | Supported | 根据accessid查询user表，鉴权  根据obj\_uri查询object表，获取元信息  根据元信息确定要读取的对象数据 |  |
| **Object ACLs (Get, Put)** | Supported |  |  |
| **Get Object Info (HEAD)** | Supported |  |  |
| **POST Object** | Supported |  |  |
| **Copy Object** | Supported |  |  |
| **Multipart Uploads** | Supported |  | (missing Copy Part) |

一共有三个表：

|  |  |  |
| --- | --- | --- |
| table | Rowkey | Column |
| user | accessid | uid,secretkey，quota[max\_files,max\_size]，stat[cur\_files,cur\_size] |
| bucket | uid\_bucketname | quota[max\_files,max\_size]，stat[cur\_files,cur\_size] |
| object | object\_uri | length,splitlist,expiretime |

1. kafka

kafka负责存储保存本端集群内的有序操作日志，并由replicater读取后发送到远端进行同步。

1. replicator

replicator负责从kafka中读取操作日志，并且转化为对于远端集群的操作请求，发送给远端集群的proxy