

***Ceph存储接口概要设计说明书***

v01.00

本手册内容会不定期进行更新修正

请在HTTP://WWW.GHTCHINA.COM及工程BBS相关平台获取最新版本

版权所有：广州广哈通信股份有限公司

责任编辑：研发部软件部 2019-2022

目录

[1 引言 4](#_Toc169177645)

[1.1 设计目标 4](#_Toc169177646)

[1.2 适用范围 4](#_Toc169177647)

[1.3 相关术语 4](#_Toc169177648)

[1.4 参考资料 5](#_Toc169177649)

[2 Ceph接口概述 6](#_Toc169177650)

[2.1 RADOS接口结构图 6](#_Toc169177651)

[2.2 RADOS接口说明 6](#_Toc169177652)

[3 总体设计 7](#_Toc169177653)

[3.1 设计思想 7](#_Toc169177654)

[3.2 块存储 8](#_Toc169177655)

[3.2.1 块存储接口框架图 8](#_Toc169177656)

[3.2.1 ceph-csi使用方法 9](#_Toc169177657)

[3.3 对象存储 9](#_Toc169177658)

[3.3.1 对象存储接口框架图 9](#_Toc169177659)

[3.3.2 调用AWS -S3接口流程 10](#_Toc169177660)

[3.3.3 AWS S3接口说明 11](#_Toc169177661)

[3.3.4 REST接口说明 15](#_Toc169177662)

[3.3.5 调用AWS boto3接口 18](#_Toc169177663)

[3.3.6 调用AWS Java接口 18](#_Toc169177664)

[4 接口安全 18](#_Toc169177665)

# 引言

Ceph是一种分布式对象存储系统，通过ceph对象网关提供对象存储接口，也称为RADOS网关（RGW）接口，它构建在ceph RADOS层之上。RGW使用librgw（RADOS Gateway library）和librados，允许应用程序与ceph对象存储建立连接。RGW为应用程序提供了一个RESTful S3/swift兼容的接口，用于在ceph集群中以对象的形式存储数据。ceph还支持多租户对象存储，可以通过RESTful API访问。此外，RGW还支持ceph管理API，可以使用本机API调用来管理ceph存储集群。

librados软件库非常灵活，允许用户应用程序通过C、C++、java、python和php绑定直接访问ceph存储集群。ceph对象存储还具有多站点功能，即灾难恢复提供解决方案。

## 设计目标

本文档目标是Ceph接口librados基础上，提供多种Ceph集群访问的接口方法，供不同的应用使用，减少对Ceph知识的依赖，提供灵活的统一的数据存储接口。

## 适用范围

本文档适用于使用Ceph存储数据的开发人员和测试人员。

## 相关术语

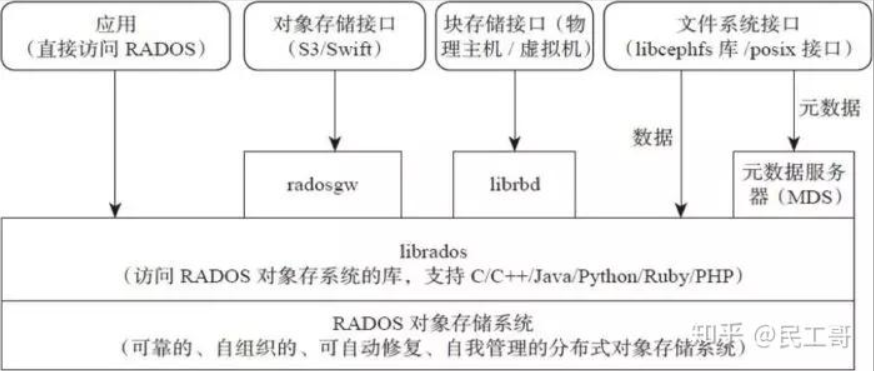
|  |  |
| --- | --- |
| PG | Placement Group，归置组 |
| MON | Ceph Monitor, 监视器(管理服务) |
| MGR | Ceph Manager |
| OSD | Object Storage Daemon (对象存储守护程序) |
| OSS | Object Storage Service(对象存储) |
| MDS | Ceph元数据服务器(Metadata Server) |
| RBD | RADOS Block Device, 块存储 |
| RADOS | Reliable Autonomic Distributed Object Store(存储集群,可靠的、自动化的、分布式的对象存储) |
| RGW | RADOS gateway/RADOSGW网关接口，提供对象存储服务，提供对象存储服务  RADOS网关=Ceph对象网关=RADOSGW=RGW |
| CRUSH | Controlled Replication Under Scalable Hashing，是一种类似于一致性hash的算法，用于为RADOS存储集群控制数据分布 |
| Librados | 简化访问RADOS的一种接口，目前支持PHP、Ruby、Java、Python、C和C++语言 |
| S3 | Simple Storage Service |
| Object | Ceph最底层的存储单元是Object对象，每个Object包含元数据和原始数据。 |
| PGP | Placement Group for Placement purpose)，相当于是PG存放的一种OSD排列组合。PGP是为实现定位而设置的PG，它的值应该与PG的总数保持一致。 |
| POOL | 存储池，也称：数据池/资源池/存储池。是一个用来存储对象的逻辑分区，方便管理对象。可理解成一个存放PG的命名空间，一个Pool可以多个PG，不同Pool的PG可以同名 |
| [Ceph CSI](https://github.com/ceph/ceph-csi) | 基于容器存储接口（Container Storage Interface） |

## 参考资料

1. 《Ceph入门学习交流.docx》
2. 《rbd块设备部署》
3. 《对象存储部署》
4. 《Amazon S3 参考》

# Ceph接口概述

## RADOS接口结构图



RADOS是Ceph最底层的功能模块，是一个无限可扩容的对象存储服务，能将文件拆解成无数个对象（碎片）存放在硬盘中，大大提高了数据的稳定性。

它主要由OSD和Monitor两个组件组成，OSD和Monitor都可以部署在多台服务器中，这就是ceph分布式的由来，高扩展性的由来。

Ceph都是通过librados库来实现数据访问，librados是上层应用访问Ceph存储的入口。

## RADOS接口说明

Ceph现在支持的客户端：

1. librados (C/C++)

librados.so是Ceph提供的原始RADOS库，它允许开发人员直接访问和操作Ceph集群。

适用场景：适用于需要直接与Ceph集群交互的应用，如分布式数据库、文件系统等，对于上层应用开发不友好，使用比较复杂。

1. rbd (Ceph块设备)

rbd是Ceph提供的分布式块设备，可提供与本地块设备类似的功能，可作为虚拟机的存储设备使用，上层应用挂载后，当作一个本地文件系统使用，不需要再开发接口。

适用场景：适用于需要高性能块存储的虚拟化环境，如OpenStack和KVM等。缺点是一个块设备只能由于一个应用挂载，否则会出现数据冲突问题，具体参考《Ceph入门学习交流》和《rbd块设备部署.txt》。

rbd需要ceph相关的工具支持，只能在服务器类应用上适用，app类应用不适合。

1. Radosgw (RADOS网关):

Radosgw是Ceph提供的对象存储的RESTful接口，Radosgw集成CivetWeb/Mongoose WEB服务器和S3模块，可通过S3和Swift协议访问。已经有的python（boto3）、C++（aws-sdk-cpp-s3）和java（aws-sdk-java）实现，通过RESTful接口与CivetWeb/Mongoose交互存储数据。

也可以用S3兼容的其他工具，象S3cmd，aws-sdk等封装访问Radosgw。

适用场景：适用于需要与云存储服务兼容的应用，如Web应用、备份存储等，这类客户端对上层应用编程方式不友好。

适用场景：适用于需要使用对象存储的应用，如图片存储、视频存储等，是目前最为广泛使用的方式。

基于AWS的S3接口开发包

Python：使用boto3包

C/C++：使用aws-sdk-cpp-s3的开发包或libs3库

Java：使用amazon的software.amazon.awssdk/s3/kms/s3control包

# 总体设计

## 设计思想

对象存储的接口，基于对现有rados接口理解，直接使用librados.so的方式开发不适合，大多数都是在原生RESTful接口或基于RESTful接口的封装基本上实现。应用一般直接使用现在SDK方法开发：

如：AWS SDK（boto3/libs3/aws-sdk-cpp-s3/aws-sdk-java）

我们提供的方案：

1. 使用radosgw的原生RESTful接口（需要在请求头中增加aws鉴权信息），应用自行开发；
2. 使用AWS SDK，直接API函数调用（手机端不适合），应用自行开发；
3. 在AWS SDK基础上再封装SDK，屏蔽ceph的桶的概念；
4. 在AWS SDK基础上再封装WEB服务，提供相对简单的RESTful接口；

## 块存储

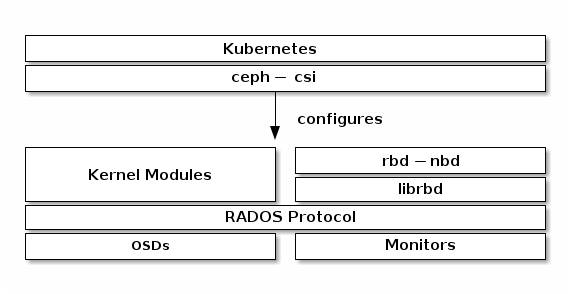
远程应用服务器可以通过远程挂载方式使用RBD块设备。详细挂载方法参考：《rbd块设备部署》

### 块存储接口框架图

Ceph CSI 是一个基于容器存储接口（Container Storage Interface, CSI）的项目，它为RBD提供了一套全面的插件，让你可以在Kubernetes环境中无缝地动态创建、删除和管理存储卷。这个项目不仅提供了便捷的存储操作，还支持包括Provisioner、Attacher、Resizer在内的多种功能，确保了Kubernetes应用的数据持久化需求。

Ceph CSI利用了Kubernetes的CSI规范，实现了一个与CO平台无关的存储接口。通过独立的RBD，它可以对接Ceph集群，提供块设备（Block）和文件系统（Filesystem）两种模式的存储服务。插件集成了最新的CSI功能，如动态卷配额调整、快照管理和扩容和缩容。

Ceph-csi使用框架如下图示：



### ceph-csi使用方法

Ceph-csi部署基本流程如下：

1. 新建一个 Ceph Pool
2. 新建用户
3. 拉取 ceph-csi
4. 修改 Configmap
5. 新建 Secret密钥
6. RBAC 授权
7. 部署 CSI sidecar
8. 部署 RBD CSI driver
9. 创建 Storageclass
10. 试用 ceph-csi

请参考《kubernetes部署ceph-csi rbd方法》

## 对象存储

### 对象存储接口框架图

****

Radosgw已经提供了一层RESTful的WEB服务，Amazon S3 SDK（支持java/C++）已经实现对S3API的封装且已经很成熟，为了操作方便，可以考虑在AWS SDK基础上自封装一套。

如果需要直接RESTful接口访问ceph，原则上可以使用radosgw提供的接口，减少中间环节，降低服务复杂度、提高系统性能和稳定性。

使用AWS SDK for C++ 做二次封装，实现功能集合。

考虑到操作便利性，如果业务确定操作在一个桶中，接口函数可以省略桶名bucketName参数，减少桶操作的接口函数。

### 调用AWS -S3接口流程

AWS-SDK-CPP-S3提供了功能完整的radosgw访问接口，直接使用更安全。

使用基本流程：

1. 初始化和关闭接口；

{

Aws::SDKOptions options;

Aws::InitAPI(options);

{

// make your SDK calls here.

}

Aws::ShutdownAPI(options);

}

1. 设置SDK选项，参考Aws::SDKOptions；
2. 构造配置参数据，参考Aws::Client::ClientConfiguration；
3. 构造S3客户端，参考Aws::S3::S3Client；
4. 使用S3客户端操作操作数据存储和获取；

调用AWS-SDK-CPP-S3接口基本使用如下：

Aws::Client::ClientConfiguration clientConfig; // 配置参数

clientConfig.endpointOverride = "172.20.0.111:7480"; // S3服务器地址和端口

clientConfig.scheme = Aws::Http::Scheme::HTTP; // HTTP方法

clientConfig.verifySSL = false; // 不SSL验证

Aws::Auth::AWSCredentials cred(access\_key, access\_secret);

Aws::S3::S3Client client(cred, NULL, clientConfig); // S3初始化

Aws::S3::S3Client client(clientConfig);

Aws::SDKOptions options;

Aws::InitAPI(options);

### AWS S3接口说明

1. **初始化sdk**

函数：Aws::InitAPI(Aws::SDKOptions options);

功能：初始化SDK。

参数：

options– AWS SDK配置选项；

返回值：

1. **关闭sdk**

函数：Aws::ShutdownAPI(Aws::SDKOptions options);

功能：关闭SDK和释放资源。

参数：

options– AWS SDK配置选项；

返回值：

1. **获取桶列表**

函数：Aws::S3::S3Client::ListBuckets();

功能：获取所有桶名。

参数：

无；

返回值：

outcome – 返回桶列表。

1. **创建桶**

函数：Aws::S3::S3Client::CreateBucket (

Aws::S3::Model:: CreateBucketRequest request);

功能：根据指定的桶名创建一个新桶。

参数：

request – 请求创建桶的参数，桶名等参数在中设置；

返回值：

参考Aws::S3::Model::CreateBucketOutcome。

1. **删除桶**

Aws::S3::S3Client:: DeleteBucket (

Aws::S3::Model:: CreateBucketRequest request);

功能：清空桶中已有所有对象和所有版本并删除桶。

参数：

request – 请求创建桶的参数，桶名等参数在中设置；

返回值：

参考Aws::S3::Model::DeleteBucketOutcome。

1. **获取桶内对象列表**

Aws::S3::S3Client::ListObjects(

Aws::S3::Model::ListObjectsRequest request);

参数：

Request – 请求列出对象的参数，桶名等参数在中设置;

返回值：

1. **上传文件**

函数：Aws::S3::S3Client::PutObject(

Aws::S3::Model::PutObjectRequest request);

功能：上传文件，请request中指定参数。

参数：

request – 请求上传参数，桶名等参数在中设置；

返回值：

参考SDK错误码定义。

1. **从桶下载文件**

函数：Aws::S3::S3Client::GetObject(

Aws::S3::Model::GetObjectRequest request);

功能：上传文件，请request中指定参数。

参数：

request – 请求下载参数，桶名等参数在中设置；

返回值：

返回结果存储在Aws::S3::Model::GetObjectOutcome对象中。

1. **从桶删除对象**

函数：Aws::S3::S3Client::DeleteObject(

Aws::S3::Model::DeleteObjectRequest request);

功能：从指定的桶里删除一个对象。

参数：

request – 删除对象指定参数，桶名等参数在中设置。

返回值：

参考SDK错误码定义。

1. **从桶删除多个对象**

函数：Aws::S3::S3Client:: DeleteObjects (

Aws::S3::Model::DeleteObjectRequest request);

功能：从指定的桶里删除一个对象。

参数：

request – 删除对象指定参数，桶名等参数在中设置。

request.SetDelete(删除对像列表);

request.SetBucket(桶名);

返回值：

DeleteObjectsOutcome类型数据。

1. **获取对像下载URL**

函数：Aws::String Aws::S3::S3Client:: GeneratePresignedUrl (

const Aws::String &bucketName，

const Aws::String &key,

const Aws::Http::HttpMethod method,

uint64\_t expirationSeconds)

功能：从指定的桶里删除一个对象。

参数：

bucketName – 桶名；

key – 对象名；

method – HTTP请求方法GET；

expirationSeconds – 超时时间；

返回值：

返回下载的URL。

1. **检索或查询对象**

函数：Aws::String Aws::S3::S3Client:: SelectObjectContent (

S3::Model::SelectObjectContentRequest request);

功能：从指定的桶里检索符合条件的对象。

参数：

request – 指定对象指定参数。

参数设置示例：

request.WithBucket("newcmbucket")

.WithKey("functest/foo.csv")

.WithExpression("select \* from s3object s")

.WithExpressionType(Aws::S3::Model::ExpressionType::SQL)

.WithInputSerialization(

Aws::S3::Model::InputSerialization().WithCSV(

Aws::S3::Model::CSVInput().WithFileHeaderInfo(

Aws::S3::Model::FileHeaderInfo::USE)

)

)

.WithOutputSerialization(

Aws::S3::Model::OutputSerialization().WithCSV(Aws::S3::Model::CSVOutput()));

返回值：

auto outcome = Aws::String Aws::S3::S3Client:: SelectObjectContent(request)

返回值类型根据不同的请求返回不同的结构，根据请求类型分析。

。

1. **设置对象元数据**

函数：Aws::S3::S3Client:: PutObjectTagging (

S3::Model:: [PutObjectTaggingReques](https://sdk.amazonaws.com/cpp/api/LATEST/aws-cpp-sdk-s3/html/class_aws_1_1_s3_1_1_model_1_1_put_object_tagging_request.html) request);

功能：设置对象的元数据。

参数：

request – 指定对象的请求参数。

返回值：

1. **获取对象元数据**

函数：Aws::S3::S3Client:: GetObjectAttributes (

S3::Model:: GetObjectAttributesRequest request);

功能：获取对象的元数据。

参数：

request – 指定对象的请求参数。

返回值：

1. **删除对象元数据**

函数：Aws::S3::S3Client:: DeleteObjectTagging (

S3::Model:: DeleteObjectTaggingRequest request);

功能：获取对象的元数据。

参数：

request – 指定对象的请求参数。

返回值：

### REST接口说明

接口的返回参数格式

字段 字段说明

code 返回码，请参考<接口错误码>

msg 可读消息

totals 返回的数据条数

data 返回数据（如有）

示例：

{

"code":200,

"msg":"OK",

"totals": 1,

"data":{ … }或"data":[ … ]

}

|  |  |
| --- | --- |
| 接口名称 | 创建桶 |
| 接口描述 | 创建一个新桶 |
| 请求URL | http://{host}:{port}/bucket |
| 请求方式 | POST |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 删除桶 |
| 接口描述 | 删除一个已经存在的桶 |
| 请求URL | http://{host}:{port}/bucket |
| 请求方式 | DELETE |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 列出所有桶名 |
| 接口描述 | 获取用户所有的桶名列表 |
| 请求URL | http://{host}:{port}/bucketsList |
| 请求方式 | GET |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 列出所有桶内文件/对象 |
| 接口描述 | 获取桶中所有文件/对象列表 |
| 请求URL | http://{host}:{port}/{bucketName}/objects |
| 请求方式 | GET |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 上传文件/对象 |
| 接口描述 | 上传一个文件/对象到桶里 |
| 请求URL | http://{host}:{port}/{bucketName}/ |
| 请求方式 | POST |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 获取下载文件URI |
| 接口描述 | 获取一个文件/对象的下载URL |
| 请求URL | http://{host}:{port}/{bucketName}/{objectKey}/url |
| 请求方式 | GET |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 删除文件/对象 |
| 接口描述 | 从指定桶里删除一个对象 |
| 请求URL | http://{host}:{port}/{bucketName}/{ objectKey} |
| 请求方式 | DELETE |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 查找文件/对象 |
| 接口描述 | 从指定桶里查询指定关键字的对象 |
| 请求URL | http://{host}:{port}/{bucketName}/objects?  keys[]=abc&keys[]=def&keys[]=xxx |
| 请求方式 | GET |
| 请求参数 | ?colors[]=1&colors[]=3 |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 查询文件/对象的TAG |
| 接口描述 | 从指定桶里查询指定关键字的对象的TAG |
| 请求URL | http://{host}:{port}/{bucketName}/{objectKey}/tag |
| 请求方式 | GET |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 设置(或修改)文件/对象的TAG |
| 接口描述 | 从指定桶里查询指定关键字的对象的TAG |
| 请求URL | http://{host}:{port}/{bucketName}/{objectKey}/tag |
| 请求方式 | POST/PATCH |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

|  |  |
| --- | --- |
| 接口名称 | 删除文件/对象的TAG |
| 接口描述 | 从指定桶里查询指定关键字的对象的TAG |
| 请求URL | http://{host}:{port}/{bucketName}/{objectKey}/tag |
| 请求方式 | DELETE |
| 请求参数 |  |
| 请求示例 |  |
| 返回参数 |  |
| 返回示例 |  |

### 调用AWS boto3接口

Python暂时可以使用AWS的boto3包，需要时再做封装。

### 调用AWS Java接口

使用AWS SDK for Java 2.x，需要时再做封装。

POM依赖包：

<dependency>

<groupId>com.amazonaws</groupId>

<artifactId>aws-java-sdk-s3</artifactId>

</dependency>

# 接口安全

1. 接口访问权限（公开访问或需要授权）
2. 授权方式和流程
3. 安全措施和防范措施
4. 这部分的目的是为了让读者了解如何安全地使用接口，并且能够正确地进行授权和认证。



**广州广哈通信股份有限公司**

地 址：广州市黄埔区科学城南云一路16号

邮 编：510663