# 概述

Ceph是一个统一的分布式存储系统，设计初衷是提供较好的性能、可靠性和可扩展性。

Ceph项目最早起源于Sage就读博士期间的工作（最早的成果于2004年发表），并随后贡献给开源社区。在经过了数年的发展之后，目前已得到众多云计算厂商的支持并被广泛应用。RedHat及OpenStack都可与Ceph整合以支持虚拟机镜像的后端存储。

# 特点

## 高性能

a. 摒弃了传统的集中式存储元数据寻址的方案，采用CRUSH算法，数据分布均衡，并行度高。

b.考虑了容灾域的隔离，能够实现各类负载的副本放置规则，例如跨机房、机架感知等。

c. 能够支持上千个存储节点的规模，支持TB到PB级的数据。

## 高可用性

a. 副本数可以灵活控制。

b. 支持故障域分隔，数据强一致性。

c. 多种故障场景自动进行修复自愈。

d. 没有单点故障，自动管理。

## 高可扩展性

a. 去中心化。

b. 扩展灵活。

c. 随着节点增加而线性增长。

## 特性丰富

a. 支持三种存储接口：块存储、文件存储、对象存储。

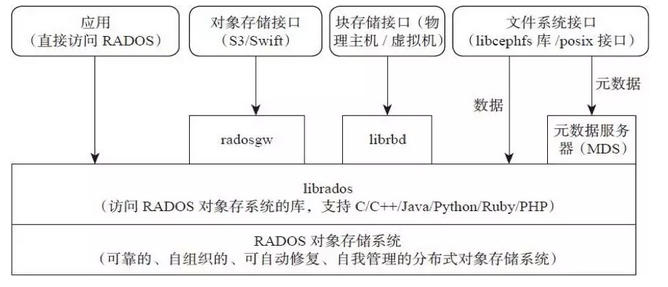
b. 支持自定义接口，支持多种语言驱动。

# 架构

支持三种接口：

Object：有原生的API，而且也兼容Swift和S3的API。

Block：支持精简配置、快照、克隆。

File：Posix接口，支持快照。

# 核心组件

## Monitor

一个Ceph集群需要多个Monitor组成的小集群，它们通过Paxos同步数据，用来保存OSD的元数据。

## OSD

OSD全称Object Storage Device，也就是负责响应客户端请求返回具体数据的进程。一个Ceph集群一般都有很多个OSD。

## MDS

MDS全称Ceph Metadata Server，是CephFS服务依赖的元数据服务。

## 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。

## PG

PG全称Placement Grouops，是一个逻辑的概念，一个PG包含多个OSD。引入PG这一层其实是为了更好的分配数据和定位数据。

## RADOS

RADOS全称Reliable Autonomic Distributed Object Store，是Ceph集群的精华，用户实现数据分配、Failover等集群操作。

## Librados

Librados是Rados提供库，因为RADOS是协议很难直接访问，因此上层的RBD、RGW和CephFS都是通过librados访问的，目前提供PHP、Ruby、Java、Python、C和C++支持。

## CRUSH

CRUSH是Ceph使用的数据分布算法，类似一致性哈希，让数据分配到预期的地方。

## RBD

RBD全称RADOS block device，是Ceph对外提供的块设备服务。

## RGW

RGW全称RADOS gateway，是Ceph对外提供的对象存储服务，接口与S3和Swift兼容。

## CephFS

CephFS全称Ceph File System，是Ceph对外提供的文件系统服务。

# 块存储

**典型设备：** 磁盘阵列，硬盘

主要是将裸磁盘空间映射给主机使用的。

**优点：**

通过Raid与LVM等手段，对数据提供了保护。

多块廉价的硬盘组合起来，提高容量。

多块磁盘组合出来的逻辑盘，提升读写效率。

**缺点：**

采用SAN架构组网时，光纤交换机，造价成本高。

主机之间无法共享数据。

**使用场景：**

docker容器、虚拟机磁盘存储分配。

日志存储。

文件存储。

# 文件存储

**典型设备：**FTP、NFS服务器

为了克服块存储文件无法共享的问题，所以有了文件存储。

在服务器上架设FTP与NFS服务，就是文件存储。

**优点：**

造价低，随便一台机器就可以了。

方便文件共享。

**缺点：**

读写速率低。

传输速率慢。

**使用场景：**

日志存储。

有目录结构的文件存储。

# 对象存储

**典型设备：**内置大容量硬盘的分布式服务器(swift, s3)

多台服务器内置大容量硬盘，安装上对象存储管理软件，对外提供读写访问功能。

**优点：**

具备块存储的读写高速。

具备文件存储的共享等特性。

使用场景： (适合更新变动较少的数据)

图片存储。

视频存储。

# IO

# 数据分布

# 心跳机制

# 通信框架

## 种类

网络通信框架三种不同的实现方式：

Simple线程模式

特点：每一个网络链接，都会创建两个线程，一个用于接收，一个用于发送。

缺点：大量的链接会产生大量的线程，会消耗CPU资源，影响性能。

Async事件的I/O多路复用模式

特点：这种是目前网络通信中广泛采用的方式。k版默认已经使用Asnyc了。

XIO方式使用了开源的网络通信库accelio来实现

特点：这种方式需要依赖第三方的库accelio稳定性，目前处于试验阶段。

## 设计模式

# 故障定位

## 故障定位思路

1. 检查配置文件是否正确，包括IP、端口、mon、osd等；
2. 检查系统资源是否满足要求，比如磁盘个数、CPU、内存；
3. 检查系统内核版本，ceph使用的很多第三方工具可能和系统版本冲突，可以再官网上进行支持的内核版本验证；
4. 通过ceph –s查询具体的错误信息；
5. Log显示超时或者断言错误，出现这类错误或者断言，可以收集好打印信息提供给ceph社区进行故障分析和支持。

## 安装部署常见故障

1. bootstrp.sls、cluster.sls分别是集群基础环境和集群的配置文件，这两个文件为yaml格式，对缩进等有严格要求；
2. salt-key –L显示无salt-minion，首先查看salt-master节点防火墙是都已关闭，然后可以尝试查看salt-minon节点的服务状态，如果提示认证被拒绝，重启salt-minion服务即可，如果提示服务端key已改变，则按照提示删除相应key并重启服务即可；
3. 环境之前部署过ceph，如果需要重新部署新的ceph集群的时候，需要将之前老的ceph集群的一些配置文件清理干净，比如monmap、crushmap等，不然可能会出现mon选举不出来leader或osd启动异常的错误；

## MON常见故障

1. health HEALTH\_WARN clock skew detected on mon.o

次警告主要是因为mon之间时间偏差超过设置的阈值

1. mon重新创建后，osd没有创建权限

解决办法：在/var/lib/ceph/bootstrap-osd目录下执行

ceph auth add “client.bootstrap-osd” –I ceph.keyring

ceph auth caps “client.bootstrap-osd” mon “allow profile bootstrap-osd”

1. mon的状态为down（ceph -s）

解决办法：首先确认mon.a是运行中状态的，其次检查所有monitor节点的通讯端口和iptables都是打开的，没有拒绝的条目。

## OSD常见故障

OSD节点一些调试命令：

查看磁盘空闲空间：df –h

查询OSD状态：ceph osd tree

查看磁盘IO情况：iostat –x

获取诊断信息：dmesg | grep scsiosd

定位故障的时候，为了防止osd自动移出map，输入命令：ceph osd set noout，然后可以stop ceph-osd id={num}或者start ceph-osd id={num}，当问题诊断完成后，可以执行ceph osd unset noout。

1. 节点运行后发现osd没有正常运行，可以通过下面的步骤分析：

检查配置文件关于osd的配置是否正确，包括IP、端口等；

检查配置文件中关于OSD的路径是否正确；

检查节点最大线程数，如果节点上运行了很多osd，需要查看是不是因为超过了默认最大线程数导致后面的osd不能正常运行；

检查系统内核版本；

Segment错误，收集日志。

1. OSD提示空间不足

OSD默认空间占满85%，就会提示状态osd.x is near full at 85%

OSD默认空间占满95%，就会提示osd.x is full at 95%，并且不接受客户端过来的写请求；

解决办法：添加新的存储空间osd，新的osd加入后ceph会重新分配存储数据到新的osd中达到平衡。需要注意的是这个过程会占用较多的计算资源，所以osd更换尽量在业务不繁忙的时候进行，需要注意的是假如选择删除一个满的osd上面的pg数目，那么需要注意的时候不要删除你数据另外一个副本的osd，否则会出现数据丢失。

## PG常见故障

PG状态：

creating：PG正在创建

active：PG激活状态，可接受请求

clean：所有在PG里的对象都被正常复制

down：有复本数据下线了，PG就down了

splitting：PG被割裂了

scrubbing：PG在被进行强一致性检测

degraded：PG中的对象副本数没有达到设置值

inconsistent：PG出现了不一致

peering：OSD之间正在协商PG的状态

repair：正在对不一致进行修复

recovering：对象正在迁移或同步

recovery\_wait：等待对象恢复

backfill：恢复期间的一个特殊状态，在此期间会扫描、同步PG的内容

backfill-wait：等待进入backfill状态

backfill\_toofull：拒绝继续backfill，OSD太满

incompelete：PG存在断档，日志记录不完整

stale：未知状态，MON未收到PG映射的变化

remapped：PG临时映射到不同的OSD集合

常见故障：

1、PG的状态始终不再clean状态，而是处于active、active+remapped or active+degraded这些状态

可能是配置文件或实际环境不符合，比如当集群只有一个节点的时候，osd crush chooseleaf type的参数大于0，会导致ceph对于某个pgs所在的osd一直向另一个不存在的节点进行交互信息。当集群只有两个osd的时候，osd pool default size的参数大于2，会导致pg处于active+degraded。

解决办法有两个：可以添加osd数目到default size，状态会变为clean；或者配置osd pool default min size参数等于2，这样也可以在degraded状态下进行数据读写。

通过ceph health detail可以查询到详细的PG异常信息。

2、PG状态始终不是clean，有时候是crush map的状态，有时候不正确

PG的状态很长时间是inactive，可以通过命令ceph pg dump\_stuck inactive来查询，问题原因可能是OSD的副本不一致导致，常会看到类似pg is down+peering的打印，重新进行recovery可能可以解决问题；

PG的状态很长时间是unclean，可以通过命令ceph pg dump\_stuck unclean来查询，问题的原因可能是unfound objects导致，常看到类似pg is active+degraded的打印。

# 编程

针对ceph存储池的情况：

1. 通过ceph df等命令获取分区数
2. 通过ceph df | tail –sec\_num | awk ‘{print $1}’获取每个分区的信息
3. 通过ceph df获取存储池pool的名称
4. 设置路径/dev/rbd/path