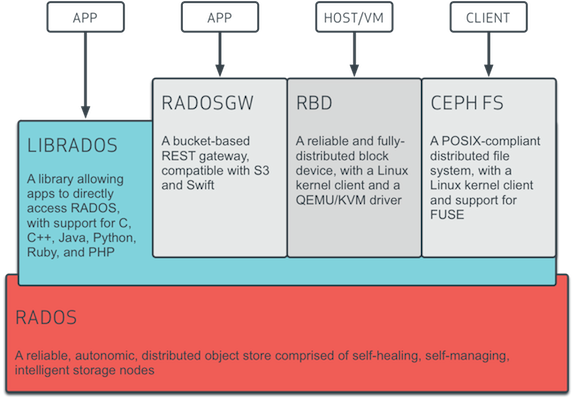
分布式文件系统代表：CEPH



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。

Ceph底层是PRDOS，是一个标准的对象存储，在此基础上Ceph提供文件，块和对象三种存储服务，其中RBD提供了块存储，CephFS模块和相应的Clinet提供了文件服务。当然，Ceph也通过RadosGW存储网关模块向外提供对象存储服务，并且和对象存储的事实标准Amazon S3以及Swift兼容。

## **Ceph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。

## ****1.5 三种存储类型-块存储****



rbd

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。

…

## ****1.6 三种存储类型-文件存储****



fs

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。

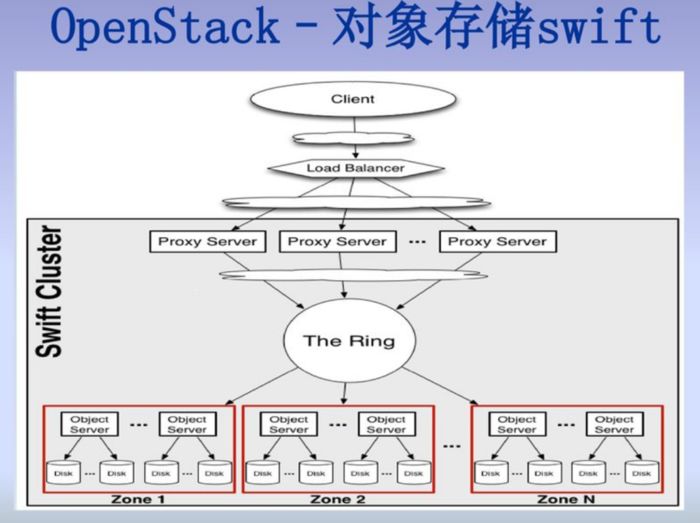
传输速率慢。

使用场景：

日志存储。

有目录结构的文件存储。

## ****1.7 三种存储类型-对象存储****



rgw

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。

…

## ****Ceph核心组件及概念介绍****

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。

Libradio  
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。

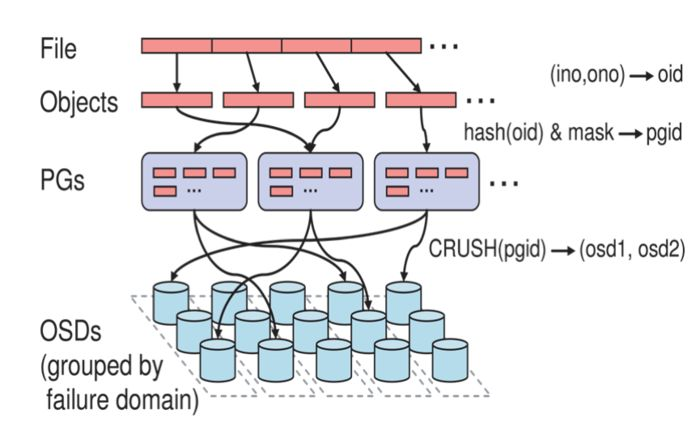
CRUSH  
CRUSH是Ceph使用的数据分布算法，类似一致性哈希，让数据分配到预期的地方。

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。

**Ceph IO算法流程**

****

1. File用户需要读写的文件。File->Object映射：

a. ino (File的元数据，File的唯一id)。  
b. ono(File切分产生的某个object的序号，默认以4M切分一个块大小)。  
c. oid(object id: ino + ono)。

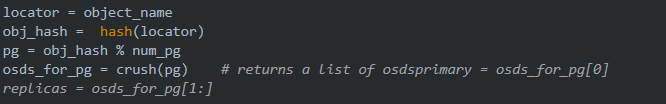
2. Object是RADOS需要的对象。Ceph指定一个静态hash函数计算oid的值，将oid映射成一个近似均匀分布的伪随机值，然后和mask按位相与，得到pgid。Object->PG映射：

a. hash(oid) & mask-> pgid 。  
b. mask = PG总数m(m为2的整数幂)-1 。

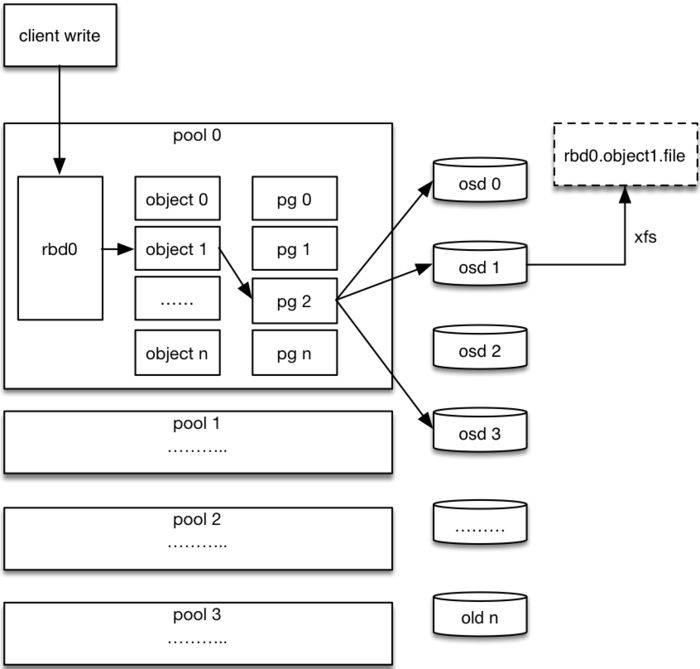
3. PG(Placement Group),用途是对object的存储进行组织和位置映射, (类似于redis cluster里面的slot的概念) 一个PG里面会有很多object。采用CRUSH算法，将pgid代入其中，然后得到一组OSD。PG->OSD映射：

a. CRUSH(pgid)->(osd1,osd2,osd3) 。

**伪代码流程**



**Ceph RBD IO流程**

****

步骤：

1. 客户端创建一个pool，需要为这个pool指定pg的数量。

2. 创建pool/image rbd设备进行挂载。

3. 用户写入的数据进行切块，每个块的大小默认为4M，并且每个块都有一个名字，名字就是object+序号。

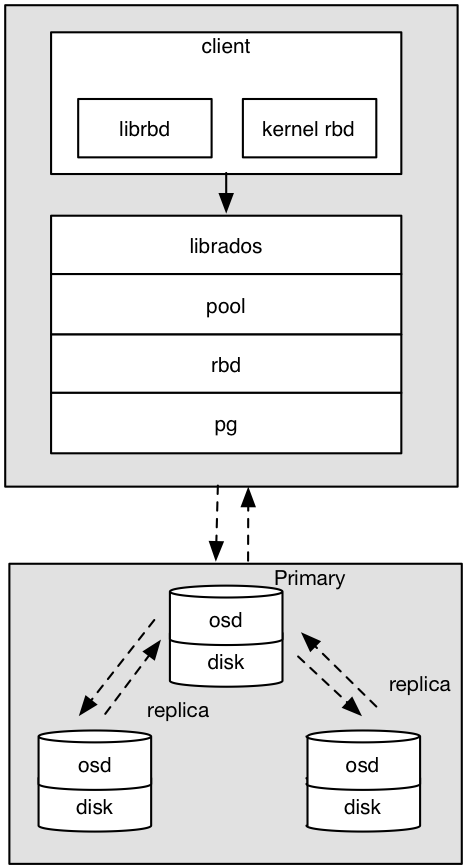
4. 将每个object通过pg进行副本位置的分配。

5. pg根据cursh算法会寻找3个osd，把这个object分别保存在这三个osd上。

6. osd上实际是把底层的disk进行了格式化操作，一般部署工具会将它格式化为xfs文件系统。

7. object的存储就变成了存储一个文rbd0.object1.file。

**Ceph RBD IO框架图**



客户端写数据osd过程：

1. 采用的是librbd的形式，使用librbd创建一个块设备，向这个块设备中写入数据。

2. 在客户端本地同过调用librados接口，然后经过pool，rbd，object、pg进行层层映射,在PG这一层中，可以知道数据保存在哪3个OSD上，这3个OSD分为主从的关系。

1. 客户端与primay OSD建立SOCKET 通信，将要写入的数据传给primary OSD，由primary OSD再将数据发送给其他replica OSD数据节点。

**Ceph Pool和PG分布情况**

说明：

pool是ceph存储数据时的逻辑分区，它起到namespace的作用。

每个pool包含一定数量(可配置)的PG。

PG里的对象被映射到不同的Object上。

pool是分布到整个集群的。

pool可以做故障隔离域，根据不同的用户场景不一进行隔离。

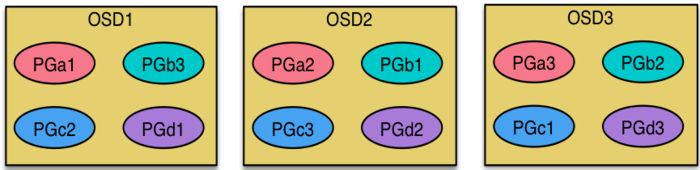
2.8 Ceph 数据扩容PG分布

场景数据迁移流程：

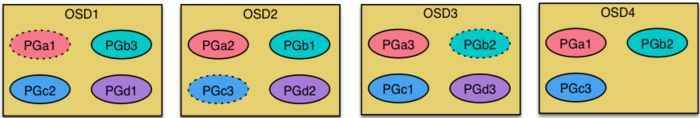
现状3个OSD, 4个PG

扩容到4个OSD, 4个PG

现状：



扩容后：



说明

每个OSD上分布很多PG, 并且每个PG会自动散落在不同的OSD上。如果扩容那么相应的PG会进行迁移到新的OSD上，保证PG数量的均衡。

**CEPH的使用：**、

预备条件：

Debian或RPM软件包依赖项列表可以安装为：



搭建CEPH

构建指令：



注意，这些指令是为开发和测试编译代码的开发人员编写的。要构建适合安装的二进制文件，我们建议您构建DEB或RPM包，或者参考CEPH.spec.in或Debian/规则，查看为生产版本指定了哪些配置选项。 前提条件：CMAKE 3.5.1 构建说明：（注意：doucmake.sh现在默认为创建CEPH的调试构建，在某些工作负载下，它的速度可以降低5倍。请通过“-dcmake\_build\_type=relwithdebinfo”执行“u cmake.sh”以创建非调试版本。） 这假定您将build dir设置为ceph.git签出的子目录。如果你把它放在别的地方，只要换掉……在do-cmake.sh中，具有正确的签出路径。在调用do-cmake之前，可以指定任何其他cmake参数来设置参数。有关详细信息，请参阅cmake选项。如 

要仅生成特定目标，请使用：



下载：

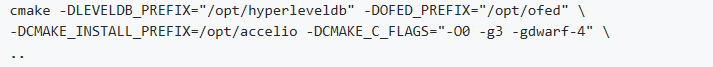


**CMake 选项**

如果手动运行cmake命令，可以使用“-d”设置许多选项。例如，构建Rados网关的选项默认为打开。在没有雷达罩网关的情况下建造：



下面的另一个示例是使用调试和几个外部依赖项的备用位置进行构建：



要查看-d选项的详细列表，可以使用以下命令调用cmake：



如果您经常将make转换为less，并且希望维护错误和警告的诊断颜色（如果编译器支持该颜色），则可以使用以下命令调用cmake：



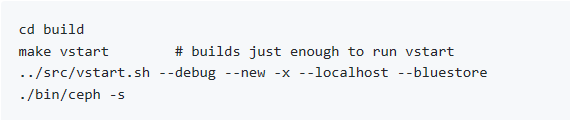
然后在执行以下操作时，您将获得诊断颜色：



“DIAGNOSTICS\_COLOR”的其他可用值为“auto”（默认）和“never”。

**构建一个源tarball**

要构建一个完整的源tarball以及从源代码构建和/或构建一个（deb或rpm）包所需的一切，请运行

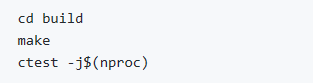


几乎所有常用命令都在bin/目录中可用。例如，

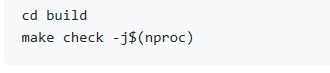


**运行测试单元**

要构建和运行所有测试（并行使用所有处理器），请使用ctest:



在CEPH中构建和运行所有测试及其依赖项，而不需要其他不必要的目标：



要手动运行单个测试，请使用-r运行ctest（regex匹配）：



**建立文档**

**先决条件**

构建文档的包依赖项列表可以在doc\_deps.deb.txt中找到：



#### 建立文档

要构建文档，请确保您位于顶层/ceph目录中，然后执行构建脚本。例如：

