零声教育 C/C++Linux服务器开发/高级架构师【零声教育】-学习视频教程-腾讯课堂 (qq.com)

1 块存储，文件存储，对象存储

1.1 简介

- **文件存储**：分层次存储，文件存储在文件夹中；访问文件时系统需要知道文件所在的路径。

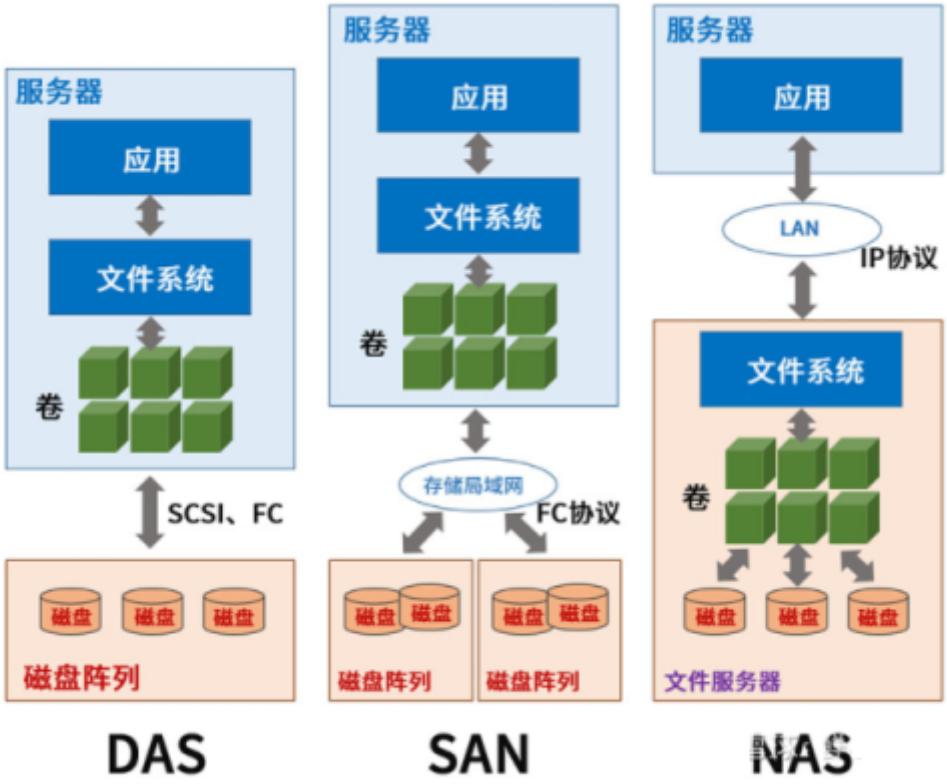
举例：企业部门之间运用网络存储器（NAS）进行文件共享。

- **块存储**：将数据拆分成块，并单独存储各个块，块是一段标准长度（块大小）的字节或比特。

举例：可用于绝大部分通用业务场景下的数据存储。

- **对象存储**：扁平结构，数据会被分解为称为"对象"的离散单元，并保持在单个存储空间中；通过API接口供客户端使用。

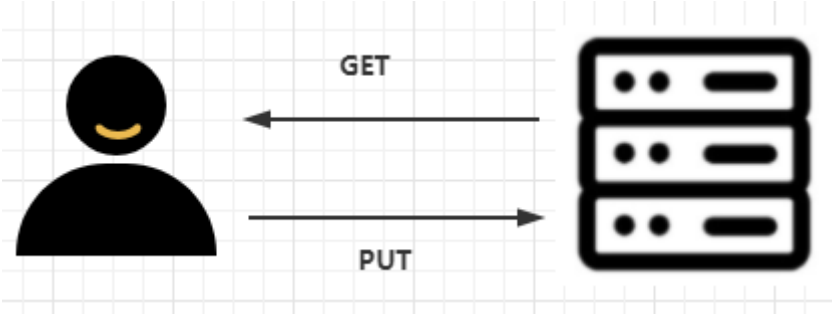
举例：AWS S3, 阿里云OSS



- 直连式存储(Direct-Attached Storage,**简称DAS**)
- 网络化存储(Fabric-Attached Storage,**简称FAS**);
- 网络化存储根据传输协议又分为：网络接入存储(Network-Attached Storage,简称NAS)和存储区域网络(Storage Area Network,**简称SAN**)。

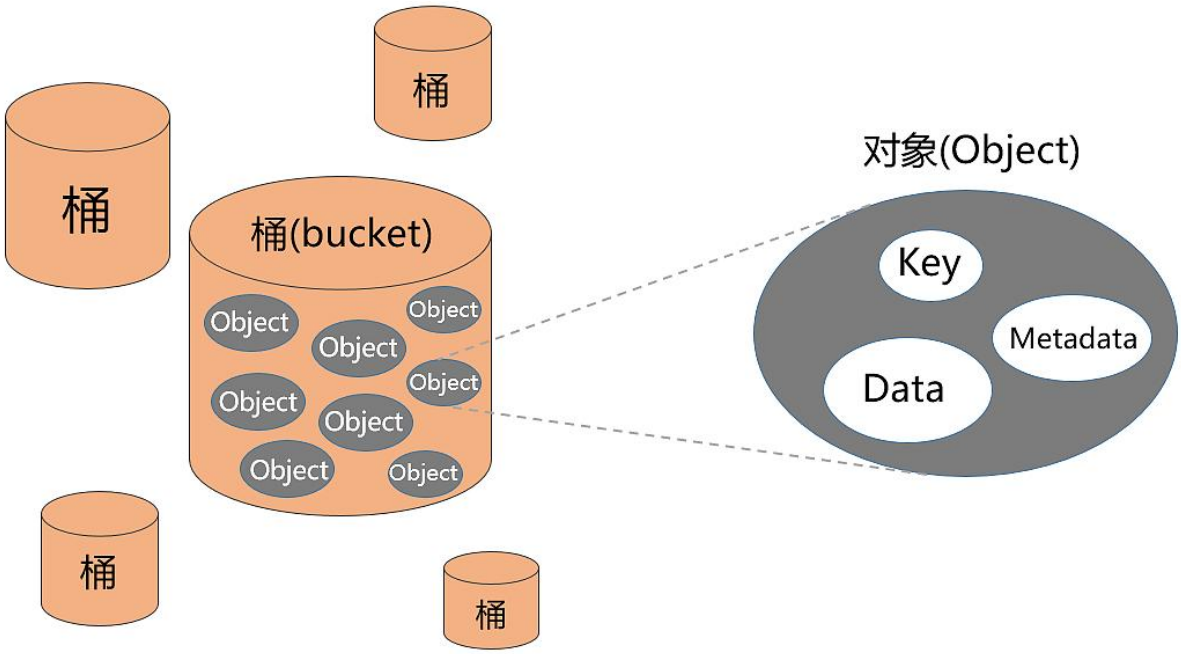
在DAS和SAN中，存储资源就像一块一块的硬盘，直接挂载在主机上，我们称之为**块存储。**
而在NAS中，呈现出来的是一个基于文件系统的目录架构，有目录、子目录、孙目录、文件，我们称之为**文件存储。**
文件存储的最大特点，就是所有存储资源都是多级路径方式进行访问的。
例如：C:\Program Files (x86)\Tencent\WeChat\WeChat.exe

- **块存储**，操作对象是磁盘。存储协议是SCSI、iSCSI、FC。以 SCSI 为例，主要接口命令有 Read/Write/Read Capacity/Inquiry 等等。
- **文件存储**，操作对象是文件和文件夹。存储协议是NFS、SAMBA（SMB）、POSIX等
- **对象存储，主要操作对象是对象（Object）。存储协议是S3、Swift等。**以 S3 为例，主要接口命令有 PUT/GET/DELETE 等。接口命令非常简洁，没有那种目录树的概念



1.2 对象存储

对象存储中的数据组成
对象存储呈现出来的是一个“桶”（bucket），你可以往“桶”里面放“对象（Object）”。这个对象包括三个部分：Key、Data、Metadata



Key

可以理解文件名，是该对象的全局唯一标识符（UID）。

Key是用于检索对象，服务器和用户不需要知道数据的物理地址，也能通过它找到对象。这种方法极大地简化了数据存储。

下面这行，就是一个对象的地址范例：

```
https://xiaozaojun.cos.ap-chengdu.myqcloud.com/Satomi_Ishihara.mp4
```

用户ID　云服务提供商　对象名

看上去就是一个URL网址。如果该对象被设置为“公开”，所有互联网用户都可以通过这个地址访问它。

Data

也就是用户数据本体。这个不用解释了。

Metadata

Metadata叫做**元数据**，它是对象存储一个非常独特的概念。

元数据有点类似数据的标签，标签的条目类型和数量是没有限制的，可以是对象的各种描述信息。

举个例子，如果对象是一张人物照片，那么元数据可以是姓名、性别、国籍、年龄、拍摄地点、拍摄时间等。

中文名	石原里美
外文名	石原さとみ、いしはらさとみ
别　名	十元、石原聪美、石神国子
国　籍	日本
民　族	大和族
星　座	摩羯座
血　型	A型
身　高	157cm
出生地	日本东京都

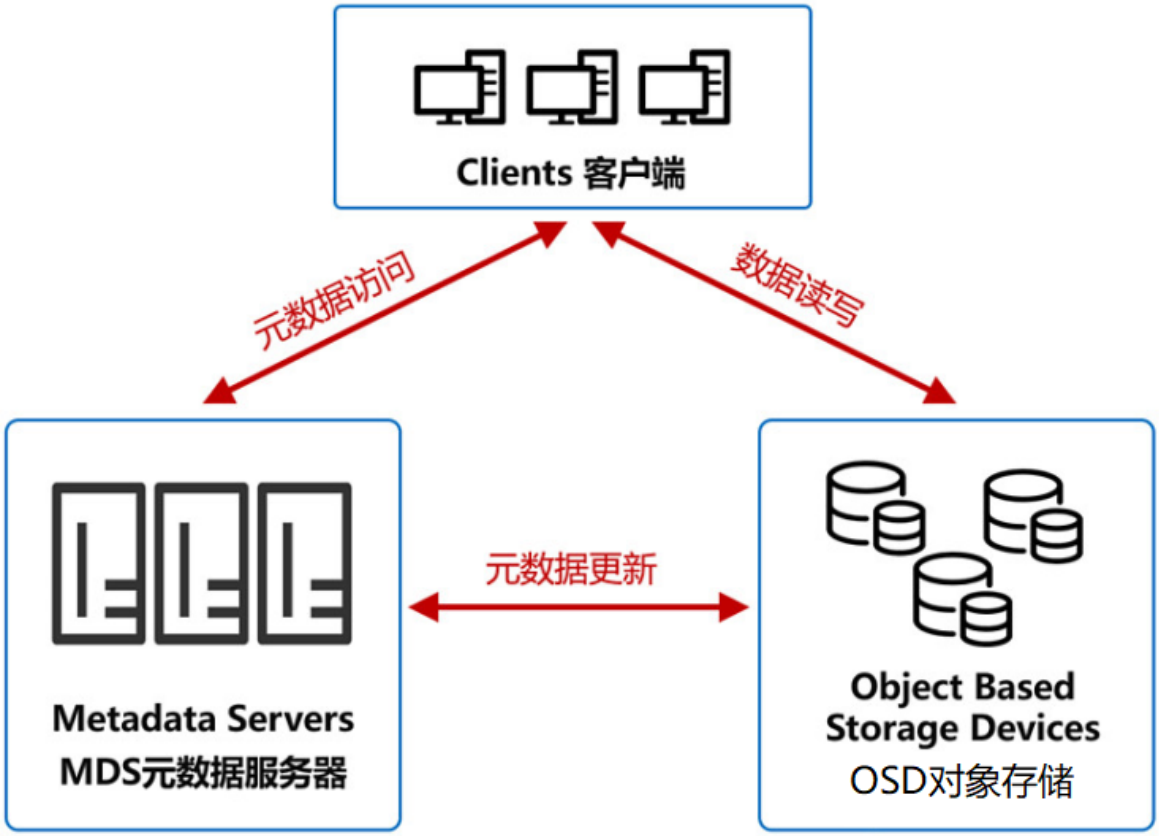
元数据可以有很多

在传统的文件存储里，这类信息属于文件本身，和文件一起封装存储。而对象存储中，元数据是独立出来的，并不在数据内部封装。

元数据的好处非常明显，可以大大加快对象的排序，还有分类和查找。

对象存储的架构

对象存储的架构是怎样的呢？如下图所示，分为3个主要部分：



对象存储的简单架构示意图

OSD对象存储设备

- 这是对象存储的核心，具有自己的CPU、内存、网络和磁盘系统。它的主要功能当然是存储数据。同时，它还会利用自己的算力，优化数据分布，并且支持数据预读取，提升磁盘性能。

MDS元数据服务器

- 它控制Client和OSD的交互，还会管理着限额控制、目录和文件的创建与删除，以及访问控制权限。

Client客户端

- 提供文件系统接口，方便外部访问。

根据上面的架构可以看出，对象存储系统可以是一个提供海量存储服务的分布式架构。

对象存储的优点

对象存储的优点很多，简单归纳如下：

容量无限大

- 对象存储的容量是EB级以上。EB有多大？大家的硬盘普遍是TB级别。1EB约等于1TB的一百万倍，请自行脑补...

- 对象存储的所有业务、存储节点采用分布式集群方式工作，各功能节点、集群都可以独立扩容。从理论上来说，某个对象存储系统或单个桶（bucket），并没有总数据容量和对象数量的限制。
- 换句话说，只要你有足够的money，服务商就可以不停地往架构里增加资源，这个存储空间就是无限的。
- 你可以根据自身需求购买相应大小的对象存储空间。如果需要调整大小，也是支持弹性伸缩的，你不要进行数据迁移和人工干预。

数据安全可靠

- 对象存储采用了分布式架构，对数据进行多设备冗余存储（至少三个以上节点），实现异地容灾和资源隔离。
- 根据云服务商的承诺，数据可靠性至少可以达到99.999999999%（不用数了，一共11个9）(强一致性)。这意味着，1000亿个文件里，每月最多只会有1个文件发生数据丢失。这比一个人被陨石击中的概率还要小143000倍。
- 数据访问方面，所有的桶和对象都有ACL等访问控制策略，所有的连接都支持SSL加密，OBS系统会对访问用户进行身份鉴权。因为数据是分片存储在不同硬盘上的，所以即使有坏人偷了硬盘，也无法还原出完整的对象数据

对象存储的应用场景
目前国内有大量的云服务提供商，他们把对象存储当作云存储在卖。
他们通常会把存储业务分为3个等级，即标准型、低频型、归档型。对应的应用场景如下：

- **标准类型**：移动应用 | 大型网站 | 图片分享 | 热点音视频
- **低频访问类型**：移动设备 | 应用与企业数据备份 | 监控数据 | 网盘应用
- **归档类型**：各种长期保存的档案数据 | 医疗影像 | 影视素材

1.3 总结

下图简要的总结了三者之间的差异：

	块存储	**文件存储**	**对象存储**
概念	用高速协议连接专业主机服务器的一种存储方式	使用文件系统，具有目录树结构	将数据和元数据当作一个对象
速度	低延迟(10ms)，热点突出	不同技术各有不同	100ms~1s，冷数据
可分布性	异地不现实	可分布式，但有瓶颈	分布并发能力高
文件大小	大小都可以，热点突出	适合大文件	适合各种大小
接口	driver，kernel module	POSIX	Restful API
典型技术	SAN	GFS,FTP,NAS	Swift, Amazon S3
适合场景	银行	数据中心	网络媒体文件存储

2 什么是ceph

众所周知，ceph是一种分布式存储系统，是有着“ceph之父”之称的Sage Weil读博期间的研究课题，项目诞生于2004年，在2006年基于开源协议开源了Ceph的源代码，在经过了数年的发展之后，已经成为了开源社区受关注较高的项目之一。

Ceph可以将多台服务器组成一个超大集群，把这些机器中的磁盘资源整合到一块儿，形成一个大的资源池（支持PB级别），然后按需分配给客户端应用使用。

ceph官网：https://ceph.com/

ceph官方文档（英文）：https://docs.ceph.com/

ceph官方文档（中文）：http://docs.ceph.org.cn/

根据官方网站上的信息，Cpeh的生态系统参加下图：

不难看出，图中列出的厂商或组织带有明显的云计算气息。

2.1 Ceph特点

1. 支持三种 **对象存储，块存储，文件存储** 接口，称之为统一存储
2. 采用CRUSH算法，数据分布均衡，并行度高，不需要维护固定的元数据结构
3. 数据具有强一致性，确保所有副本写入完成后才返回确认，适合读多写少的场景
4. 去中心化，没有固定的中心节点，集群扩展灵活

CRUSH需要集群的映射，并使用CRUSH映射在OSDs中伪随机存储和检索数据，数据在集群中均匀分布。

2.2 ceph缺点

1. 去中心化的分布式解决方案，需要提前做好组件和节点部署规划设计
2. ceph扩容时，由于其数据分布均衡的特性，会导致整个存储系统性能下降

2.3 Ceph相较于其他存储方案的优势

1. **CRUSH算法**
   a. CURSH是ceph的两大创新之一（另一大就是去中心化），ceph摒弃了传统的集中式存储元数据寻址的方案，转而使用CRUSH算法计算的方式完成数据的寻址操作；
   b. crush算法有强大的扩展性（即高扩展性），理论上支持上千个存储节点规模。
2. **高可用**
   a. 数据副本数可以灵活调整；
   b. 可以通过crush算法指定副本的物理存放位置以分割故障域，支持数据强一致性；
   c. 支持多种故障场景自动尝试进行修复；

d. 支持多份强一致性副本，副本能够垮主机、机架、机房、数据中心存放，安全可靠；
e. 存储节点可以自管理、自动修复。无单点故障，容错性强

3. **高性能**
   a. 因为是多个副本，因此在读写操作时能够做到高度并行化，理论上，节点越多，整个cpeh集群的IOPS和吞吐量就越高；
   b. ceph客户端读写数据可直接与存储设备-OSD交互，在块存储和对象存储中无需元数据-MDS服务
4. **特性丰富**
   a. 支持三种存储接口：对象存储，块存储，文件存储，三种方式可一同使用
   b. 支持自定义接口，支持多种语言驱动

3 Ceph的三种存储接口(块设备、文件系统、对象存储)

Ceph可以一套存储系统同时提供块设备存储、文件系统存储和对象存储三种存储功能。没有存储基础的用户则比较难理解Ceph的块存储、文件系统存储和对象存储接口。该章节先让大家大体理解这三者在ceph的区别我们再去研究ceph的架构和原理。

3.1 Ceph的块设备存储接口

首先，什么是块设备?

块设备是i/o设备中的一类，是将信息存储在固定大小的块中，每个块都有自己的地址，还可以在设备的任意位置读取一定长度的数据。看不懂？那就暂且认为块设备就是硬盘或虚拟硬盘吧。

查看下Linux环境中的设备：

```
root@nb:~$ ls /dev/
/dev/sda/ dev/sda1 /dev/sda2 /dev/sdb /dev/sdb1 /dev/hda
/dev/rbd1 /dev/rbd2 …
```

上面的/dev/sda、/dev/sdb和/dev/hda都是块设备文件，这些文件是怎么出现的呢?
当给计算机连接块设备（硬盘）后，系统检测的有新的块设备，该类型块设备的驱动程序就在/dev/下创建个对应的**块设备**设备文件，用户就可以通过设备文件使用该块设备了。

它们怎么有的叫 sda？有的叫 sdb？有的叫 hda?
以sd开头的块设备文件对应的是SATA接口的硬盘，而以hd开头的块设备文件对应的是IDE接口的硬盘。那SATA接口的硬盘跟IDE接口的硬盘有啥区别？你只需要知道，IDE接口硬盘已经很少见到了，逐渐被淘汰中，而SATA接口的硬盘是目前的主流。而sda和sdb的区别呢？当系统检测到多个SATA硬盘时，会根据检测到的顺序对硬盘设备进行字母顺序的命名。

怎么还有的叫 rbd1 和 rbd2 呢? rbd(rados block devices**)**
被你发现了，rbd就是我们压轴主角了。*rbd*就是由Ceph集群提供出来的块设备。可以这样理解，sda和hda都是通过数据线连接到了真实的硬盘，而rbd是通过网络连接到了Ceph集群中的一块存储区域，往rbd设备文件写入数据，最终会被存储到Ceph集群的这块区域中。

那么块设备怎么用呢？这里举个例子：
打个比方，一个块设备是一个粮仓，数据就是粮食。农民伯伯可以存粮食（写数据）了，需要存100斤玉米，粮仓（块设备）这么大放哪里呢，就挨着放（顺序写）吧。又需要存1000斤花生，还是挨着放吧。又需要存……

后来，农民伯伯来提粮食（读数据）了，他当时存了1000斤小麦，哎呀妈呀，粮仓这么大，小麦在哪里啊？仓库管理员找啊找，然后哭晕在了厕所……

新管理员到任后，想了个法子来解决这个问题，用油漆把仓库划分成了方格状，并且编了号，在仓库门口的方格那挂了个黑板，当农民伯伯来存粮食时，管理员在黑板记录，张三存了1000斤小麦在xx方格处。后来，农民伯伯张三来取粮食时，仓库管理员根据小黑板的记录很快提取了粮食。

故事到此为止了，没有方格和黑板的仓库（块设备）称为**裸设备。**由上例可见，裸设备对于用户使用是很不友好的，直接导致了旧仓库管理员的狗带。例子中**划分方格和挂黑板的过程其实是在块设备上构建文件系统的过程，文件系统可以帮助块设备对存储空间进行条理的组织和管理，于是新管理员通过文件系统（格子和黑板）迅速找到了用户（农民伯伯张三）存储的数据（1000斤小麦）。**针对多种多样的使用场景，衍生出了很多的文件系统。有的文件系统能够提供更好的读性能，有的文件系统能提供更好的写性能。我们平时常用的文件系统如xfs、ext4是读写性能等各方面比较均衡的通用文件系统。

能否直接使用不含有文件系统块设备呢?

可以的，xfs和ext4等通用的文件系统旨在满足大多数用户的存储需求，所以在数据存储的各方面的性能比较均衡。然而，很多应用往往并不需要这种均衡，而需要突出某一方面的性能，如小文件的存储性能。此时，xfs、ext4等通用文件系统如果不能满足应用的需求，应用往往会在裸设备上实现自己的数据组织和管理方式。简单的说，**就是应用为了强化某种存储特性而实现自己定制的数据组织和管理方式，而不使用通用的文件系统。**

总结一下，块设备可理解成一块硬盘，用户可以直接使用不含文件系统的块设备，也可以将其格式化成特定的文件系统，由文件系统来组织管理存储空间，从而为用户提供丰富而友好的数据操作支持。

3.2 Ceph的文件系统存储接口

什么是Ceph的文件系统接口?

还记得上面说的块设备上的文件系统吗，用户可以在块设备上创建xfs文件系统，也可以创建ext4等其他文件系统。如图3.1，Ceph集群实现了自己的文件系统来组织管理集群的存储空间，用户可以直接将Ceph集群的文件系统挂载到用户机上使用。

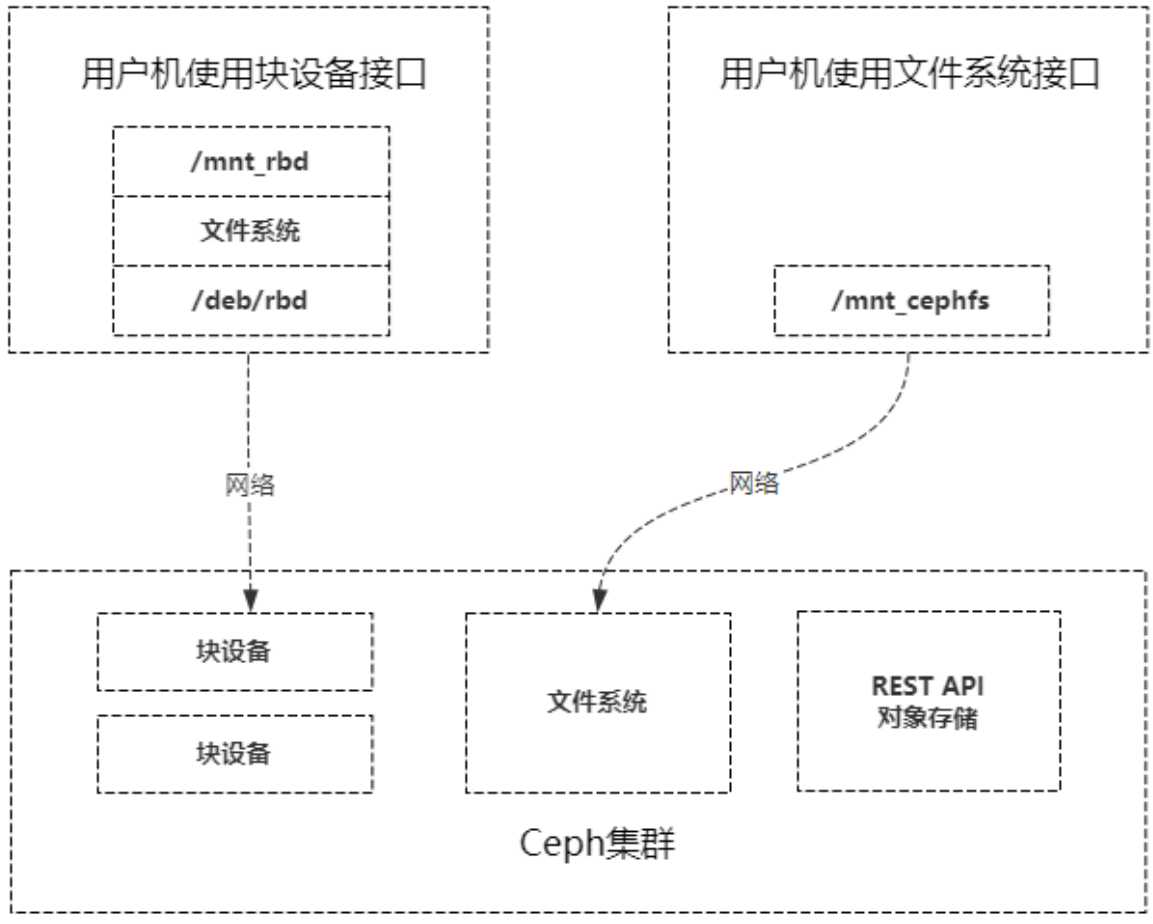
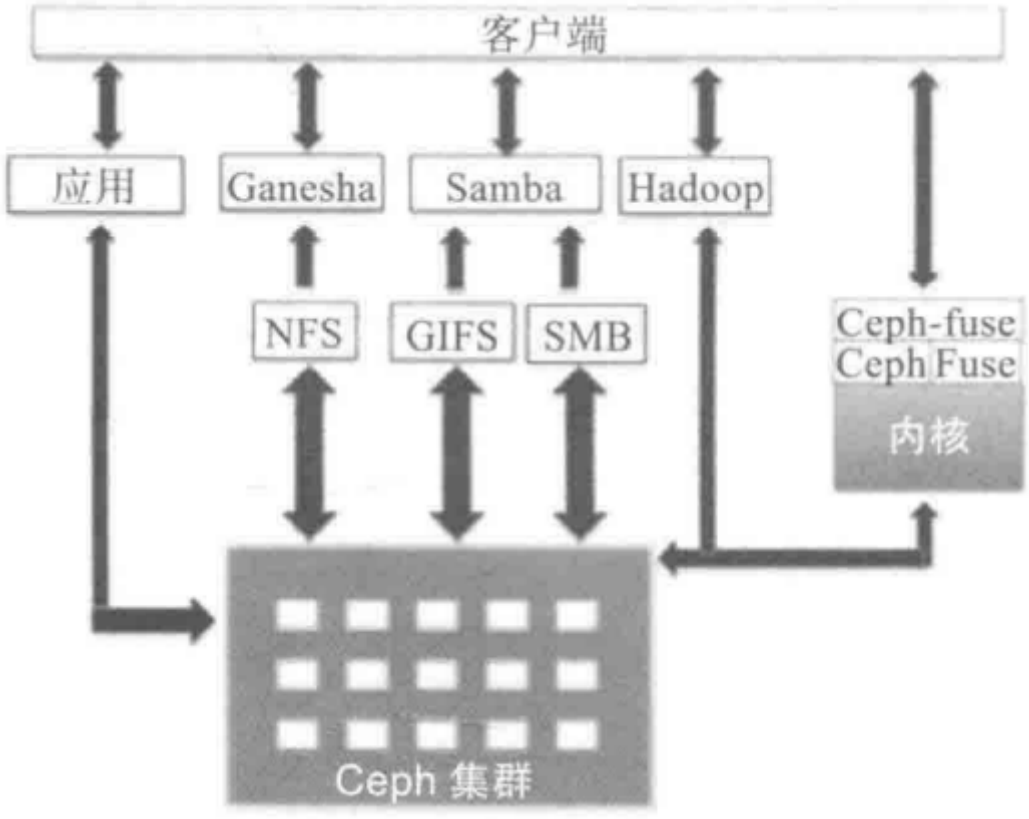


图3.1 Ceph的块设备接口和文件系统接口对比



块存储和文件存储

Ceph有了块设备接口，在块设备上完全可以构建一个文件系统，那么Ceph为什么还需要文件系统接口呢?

主要是因为应用场景的不同，Ceph的块设备具有优异的读写性能，**但不能多处挂载同时读写，目前主要用在OpenStack上作为虚拟磁盘，而Ceph的文件系统接口读写性能较块设备接口差，但具有优异的共享性。**

为什么Ceph的块设备接口不具有共享性，**而Ceph的文件系统接口具有呢**?

对于Ceph的块设备接口，如图3.2，文件系统的结构状态是维护在**各用户机内存**中的，假设Ceph块设备同时挂载到了用户机1和用户机2，当在用户机1上的文件系统中写入数据后，更新了用户机1的内存中文件系统状态，最终数据存储到了Ceph集群中，但是此时用户机2内存中的文件系统并不能得知底层Ceph集群数据已经变化而维持数据结构不变，因此用户无法从用户机2上读取用户机1上新写入的数据。

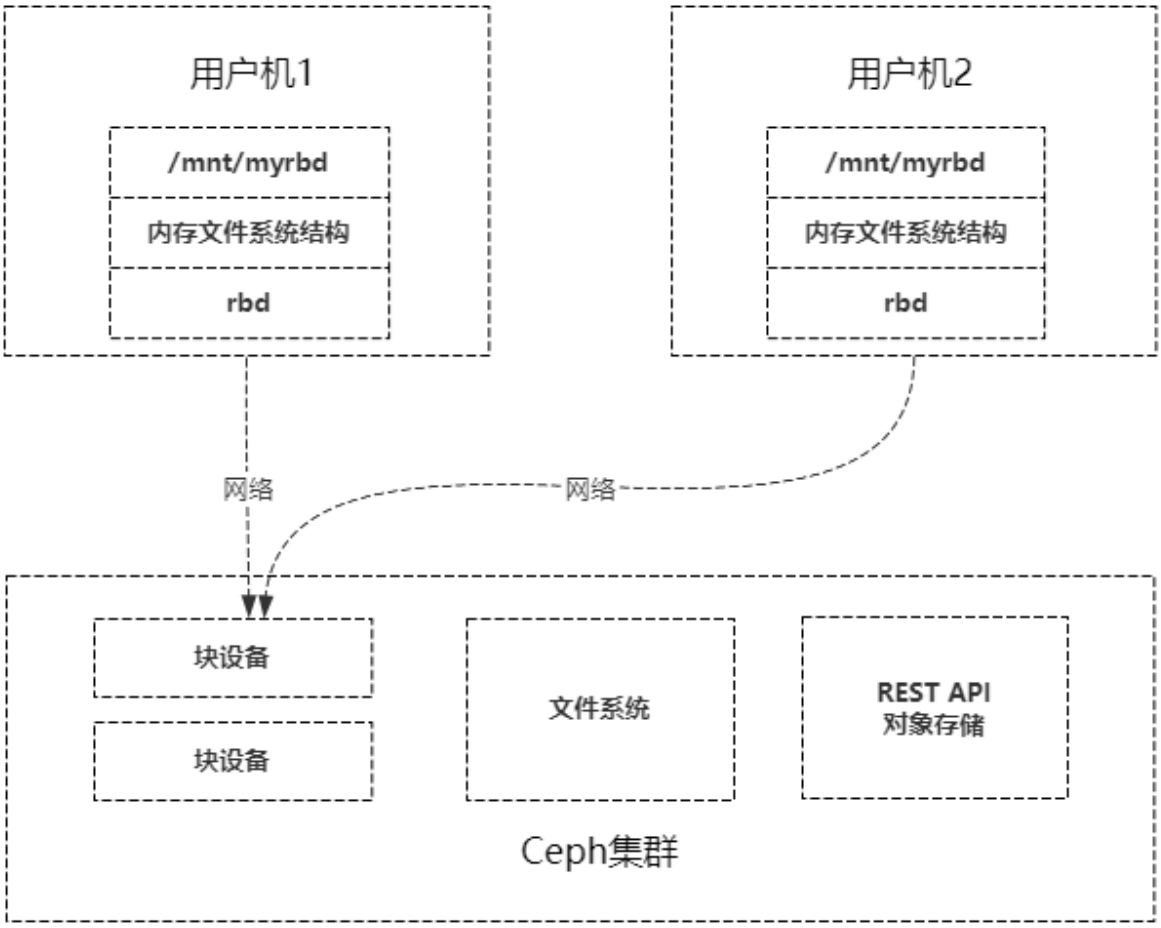


图3.2 Ceph块设备接口共享性

对于Ceph的文件系统接口，如图3.3，文件系统的结构状态是维护在远端Ceph集群中的，Ceph文件系统同时挂载到了用户机1和用户机2，当往用户机1的挂载点写入数据后，远端Ceph集群中的文件系统状态结构随之更新，当从用户机2的挂载点访问数据时会去远端Ceph集群取数据，由于远端Ceph集群已更新，所有用户机2能够获取最新的数据。

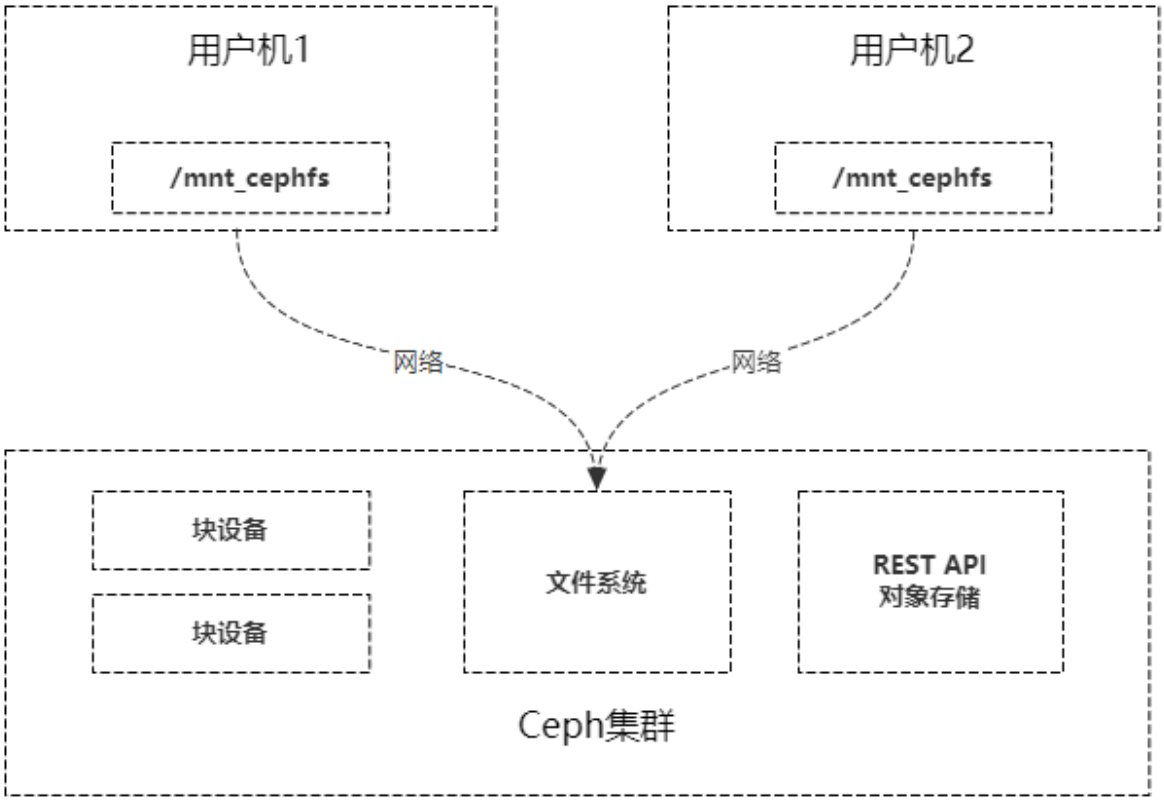


图3.3 Ceph文件系统接口共享性

Ceph的文件系统接口使用方式?

将Ceph的文件系统挂载到用户机目录

```
/* 保证/etc/ceph目录下有Ceph集群的配置文件ceph.conf和ceph.client.admin.keyring */

mkdir -p /mnt/ceph_fuse

ceph-fuse /mnt/ceph_fuse
```

大功告成，在/mnt/ceph_fuse下读写数据，都是读写远程Ceph集群

总结一下，Ceph的文件系统接口弥补了Ceph的块设备接口在共享性方面的不足，**Ceph的文件系统接口符合POSIX标准，用户可以像使用本地存储目录一样使用Ceph的文件系统的挂载目录。**还是不懂？这样理解吧，无需修改你的程序，就可以将程序的底层存储换成空间无限并可多处共享读写的Ceph集群文件系统。

3.3 Ceph的对象存储接口

首先，通过图4来看下对象存储接口是怎么用的?

简单了说，使用方式就是通过http协议上传下载删除对象（文件即对象）。

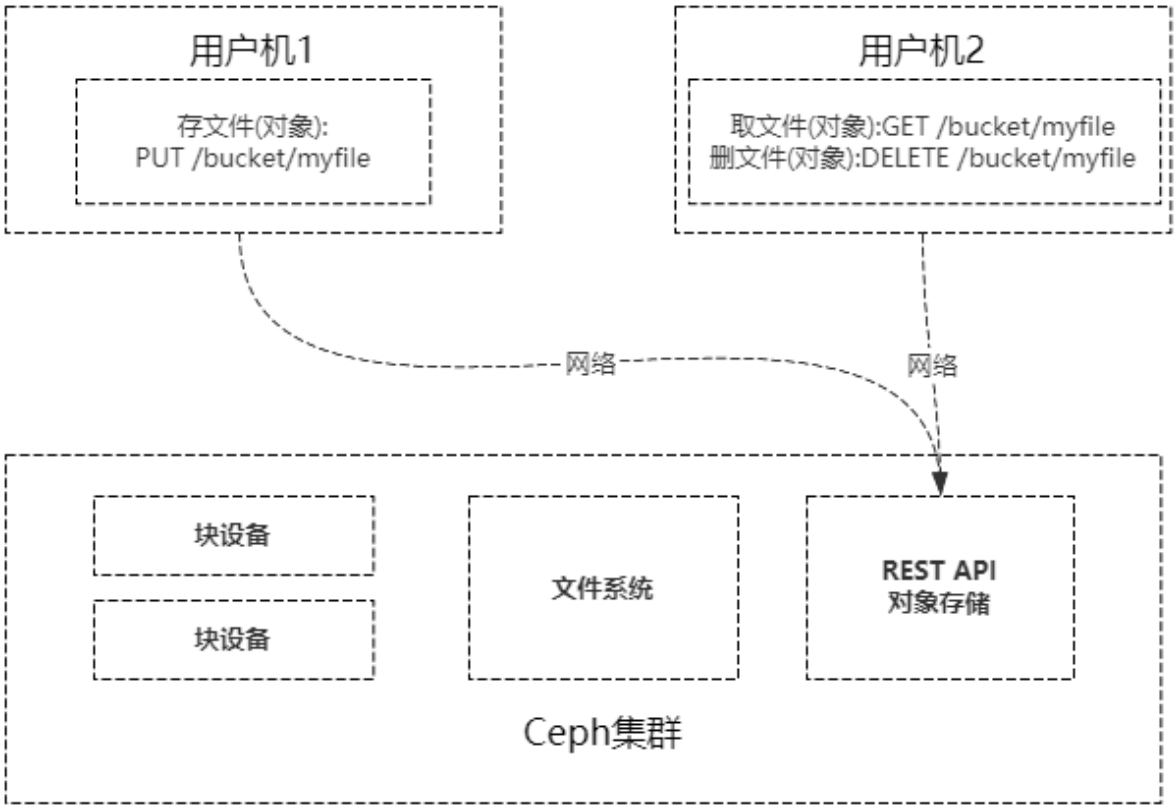
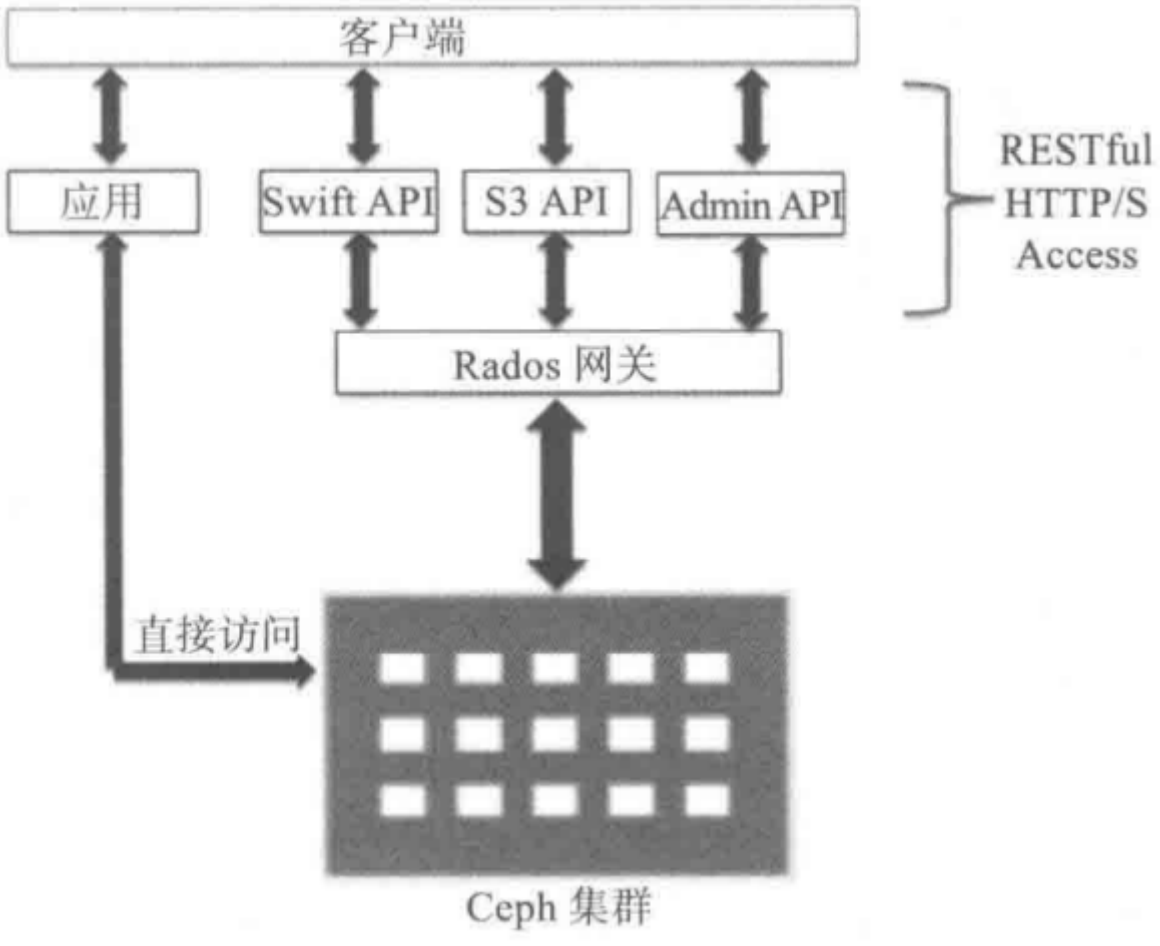


图4 对象存储接口的使用方式



老问题来了，有了块设备接口存储和文件系统接口存储，为什么还整个对象存储呢?

往简单了说，Ceph的块设备存储具有优异的存储性能但不具有共享性，而Ceph的文件系统具有共享性然而性能较块设备存储差，为什么不权衡一下存储性能和共享性，整个具有共享性而存储性能好于文件系统存储的存储呢，对象存储就这样出现了。

对象存储为什么性能会比文件系统好?
原因是多方面的，主要原因是：

- 对象存储组织数据的方式相对简单，只有bucket和对象两个层次（对象存储在bucket中），对对象的操作也相对简单。
- 而文件系统存储具有复杂的数据组织方式，目录和文件层次可具有无限深度，对目录和文件的操作也复杂的多，因此文件系统存储在维护文件系统的结构数据时会更加繁杂，从而导致文件系统的存储性能偏低。

Ceph的对象存储接口怎么用呢?
Ceph的对象接口符合亚马逊S3接口标准和OpenStack的Swift接口标准，可以自行学习这两种接口。

总结一下，文件系统存储具有复杂的数据组织结构，能够提供给用户更加丰富的数据操作接口，而对象存储精简了数据组织结构，提供给用户有限的数据操作接口，以换取更好的存储性能。对象接口提供了REST API，非常适用于作为web应用的存储。

3.4 总结

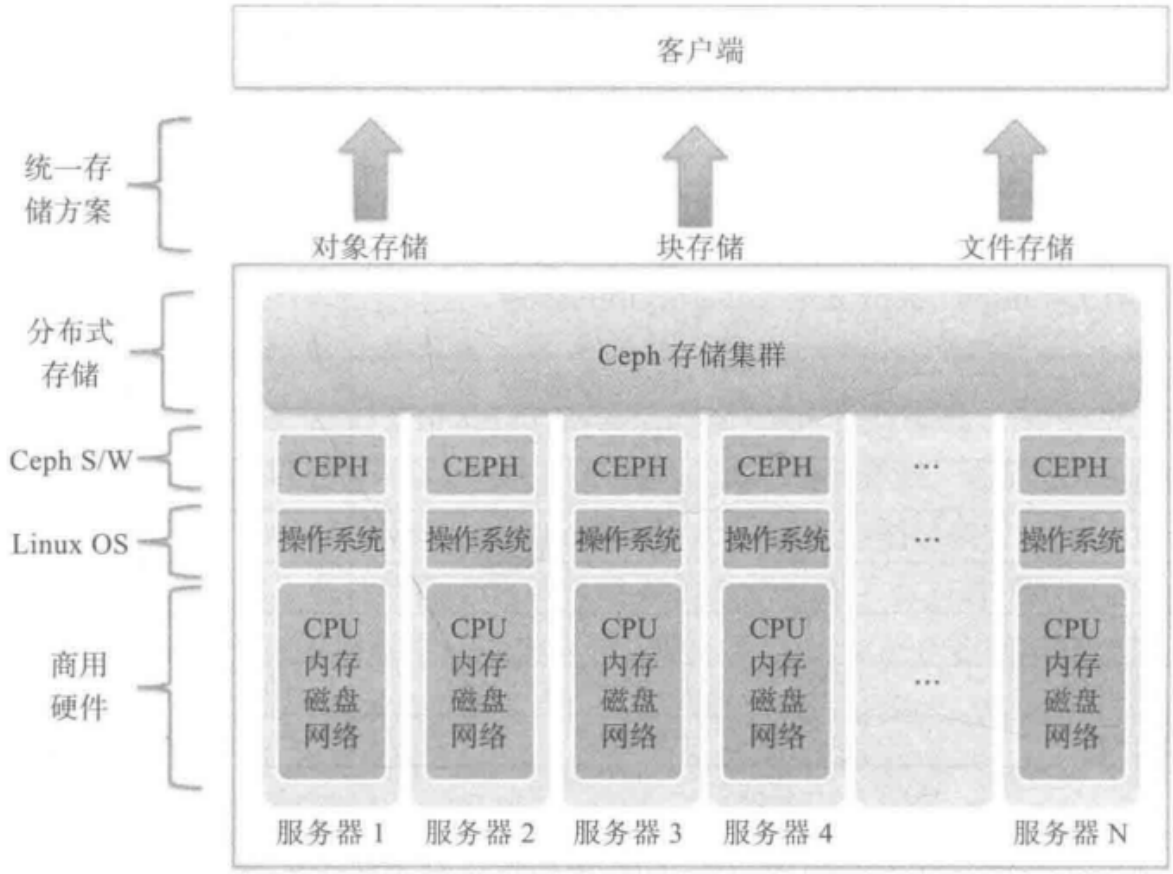
概括一下，块设备速度快，对存储的数据没有进行组织管理，但在大多数场景下，用户数据读写不方便（以块设备**位置offset + 数据的length**来记录数据位置，读写数据）。而在块设备上构建了文件系统后，文件系统帮助块设备组织管理数据，数据存储对用户更加友好（以**文件名**来读写数据）。Ceph文件系统接口解决了“Ceph块设备+本地文件系统”不支持多客户端共享读写的问题，但由于文件系统结构的复杂性导致了存储性能较Ceph块设备差。对象存储接口是一种折中，保证一定的存储性能，同时支持多客户端共享读写。

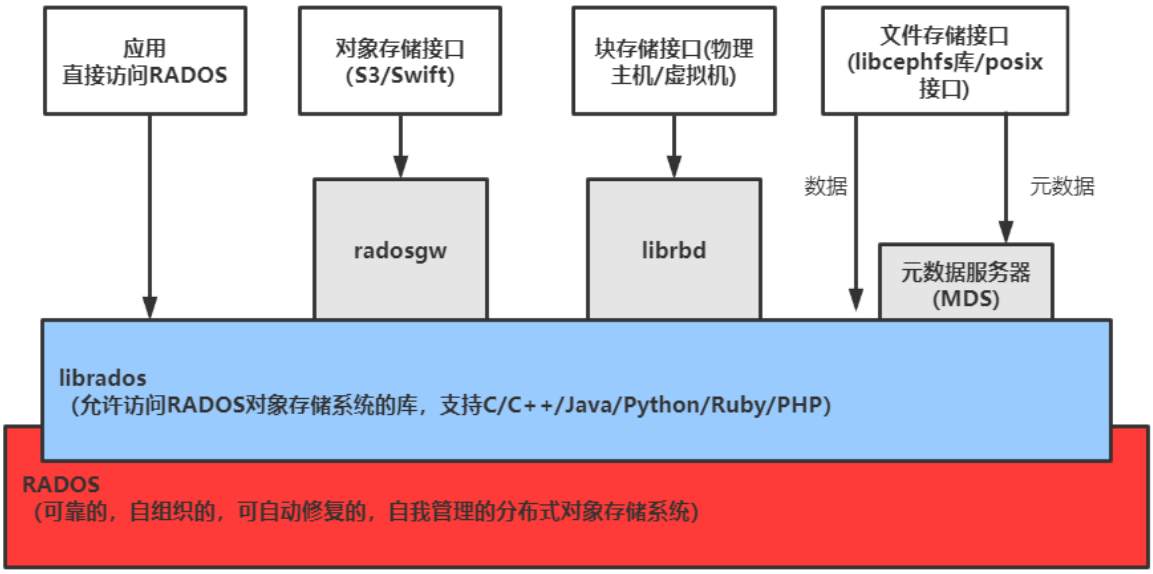
4 ceph的架构

4.1 ceph核心架构组件

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。





官方原架构图请看https://docs.ceph.com/en/latest

由上图所示，自下往上，逻辑上可以分为四个层次：

1. **基础存储系统RADOS(Reliable Autonomic Object Store，可靠、自动、分布式对象存储)**

RADOS是ceph存储集群的基础，这一层本身就是一个完整的对象存储系统。Ceph的高可靠、高可扩展、高性能、高自动化等等特性本质上也都是由这一层所提供的，在ceph中，所有数据都以对象的形式存储，并且无论什么数据类型，RADOS对象存储都将负责保存这些对象，确保了数据一致性和可靠性。RADOS系统主要由两部分组成，分别是**OSD(对象存储设备)和Monitor（监控OSD）**。

2. **基础库LIBRADOS**

LIBRADOS基于RADOS之上，它允许应用程序通过访问该库来与RADOS系统进行交互，支持多种编程语言，比如C、C++、Python等。

3. **上层接口RADOSGW、RBD和CEPHFS**

基于LIBRADOS层开发的三个接口，其作用是在librados库的基础上提供抽象层次更高、更便于应用或客户端使用的上层接口。

- RADOS GW（简称**RGW**）提供**对象存储服务**，是一套基于RESTFUL协议的网关，支持对象存储，**兼容S3和Swift**
- RBD提供分布式的**块存储设备**接口，主要面向虚拟机提供虚拟磁盘，可以被映射、格式化，像磁盘一样挂载到服务器使用。
- CephFS是一个POSIX兼容的**分布式文件系统**，依赖MDS来跟踪文件层次结构，基于librados封装原生接口，它跟传统的文件系统如 Ext4 是一个类型的，但区别在于分布式存储提供了并行化的能力，像NFS等也是属于文件系统存储。

PS：两个对象的区分，需要说明下，这里提到两个对象的概念。一个是**RGW中的对象存储**；一个是**Ceph 的后端存储的对象**，这两个需要区分：

- 第一个对象面向用户，是用户接口能访问到的对象；
- 第二个对象是ceph 服务端操作的对象

eg：使用RGW接口，存放一个1G的文件，在用户接口看到的就是存放了一个对象；而后通过RGW 分片成多个对象后最终存储到磁盘上；

RGW为RADOS Gateway的缩写，ceph通过RGW为互联网云服务提供商提供对象存储服务。RGW在librados之上向应用提供访问ceph集群的RestAPI，支持Amazon S3和openstack swift两种接口。对RGW最直接的理解就是一个协议转换层，把从上层应用符合S3或Swift协议的请求转换成rados的请求，将数据保存在rados集群中。

4.2 Ceph核心组件及概念介绍

- **Monitor（ceph-mon）**
  维护集群Cluster Map的状态，维护集群的Cluster MAP二进制表，**保证集群数据的一致性。**
  Cluster MAP描述了对象块存储的物理位置，以及一个将设备聚合到物理位置的桶列表，map中包含monitor组件信息，manger 组件信息，osd 组件信息，mds 组件信息，crush 算法信息。还负责ceph集群的身份验证功能，client 在连接ceph集群时通过此组件进行验证。
- **OSD（ceph-osd）**
  OSD全称Object Storage Device，用于集群中所有数据与对象的存储。ceph 管理物理硬盘时，引入了OSD概念，每一块盘都会针对的运行一个OSD进程。换句话说，ceph 集群通过管理 OSD 来管理物理硬盘。**负责处理集群数据的复制、恢复、回填、再均衡**，并向其他osd守护进程发送心跳，然后向Mon提供一些监控信息。当Ceph存储集群设定数据有两个副本时（一共存两份），则至少需要三个OSD守护进程即三个OSD节点，集群才能达到active+clean状态，实现冗余和高可用。
- **Manager（ceph-mgr）**

用于 **收集ceph集群状态**、运行指标，比如存储利用率、当前性能指标和系统负载。对外提供 ceph dashboard（ceph ui）和 resetful api。Manager组件开启高可用时，至少2个实现高可用性。

- **MDS（ceph-mds）**

Metadata server，元数据服务。为ceph 文件系统提供元数据计算、缓存与同步服务（ceph 对象存储和块存储不需要MDS）。同样，元数据也是存储在osd节点中的，mds类似于元数据的 **代理缓存服务器**，为 posix 文件系统用户提供性能良好的基础命令（ls，find等）不过只是在需要使用CEPHFS时，才需要配置MDS节点。

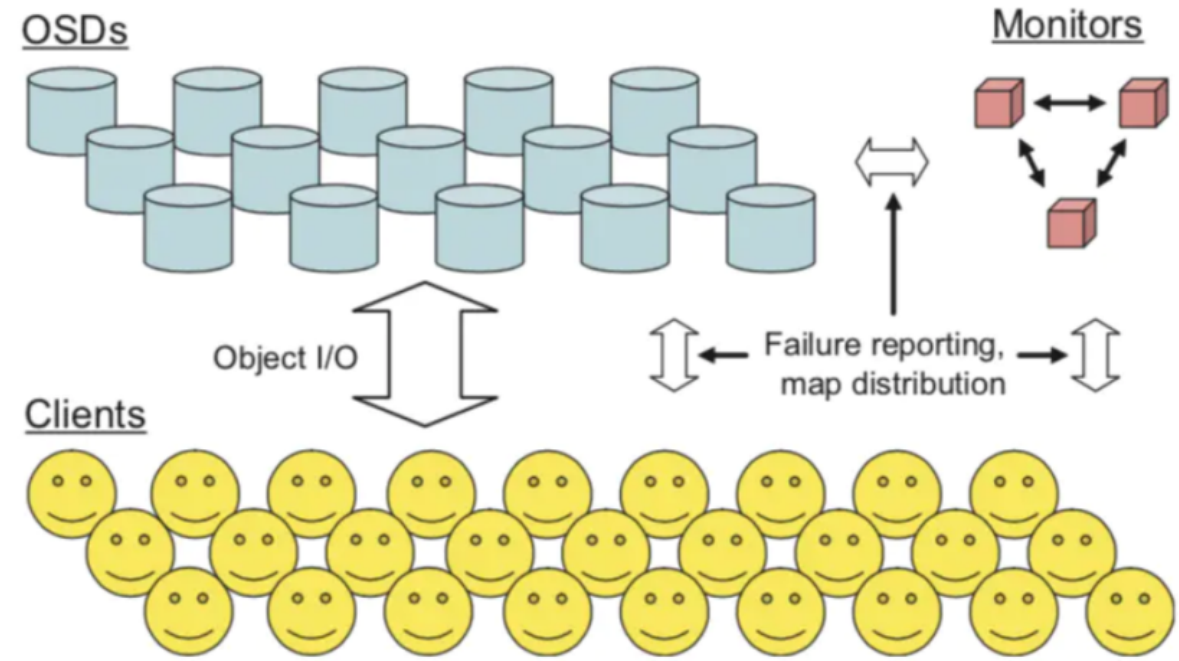
- **Object**
  Ceph最底层的存储单元是Object对象，每个Object包含元数据和原始数据
- **PG**
  PG全称Placement Grouops，是一个逻辑的概念，一个PG包含多个OSD。引入PG这一层其实是为

了更好的分配数据和定位数据。

- **RADOS**
  RADOS全称Reliable Autonomic Distributed Object Store（**可靠、自治、分布式对象存储**），是Ceph集群的精华，用户实现数据分配、Failover（故障转移）等集群操作。
- **Libradio驱动库**
  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**R**ADOS** 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。

4.2.1 RADOS的系统逻辑结构

RADOS (Reliable, Autonomic Distributed Object Store) 是Ceph的核心之一，作为Ceph分布式文件系统的一个子项目，特别为Ceph的需求设计，能够在动态变化和异质结构的存储设备机群之上提供一种稳定、可扩展、高性能的单一逻辑对象(Object)存储接口和能够实现节点的自适应和自管理的存储系统。在传统分布式存储架构中，存储节点往往仅作为被动查询对象来使用，随着存储规模的增加，数据一致性的管理会出现很多问题。而新型的存储架构倾向于将基本的块分配决策和安全保证等操作交给存储节点来做，然后通过提倡客户端和存储节点直接交互来简化数据布局并减小io瓶颈。



628983 Ceph分布式存储学习指南-有书签.pdf

设计图来源于ceph官方RADOS文档：https://ceph.com/wp-content/uploads/2016/08/weil-rados-pdsw07.pdf
服务端 RADOS 集群主要由两种节点组成：

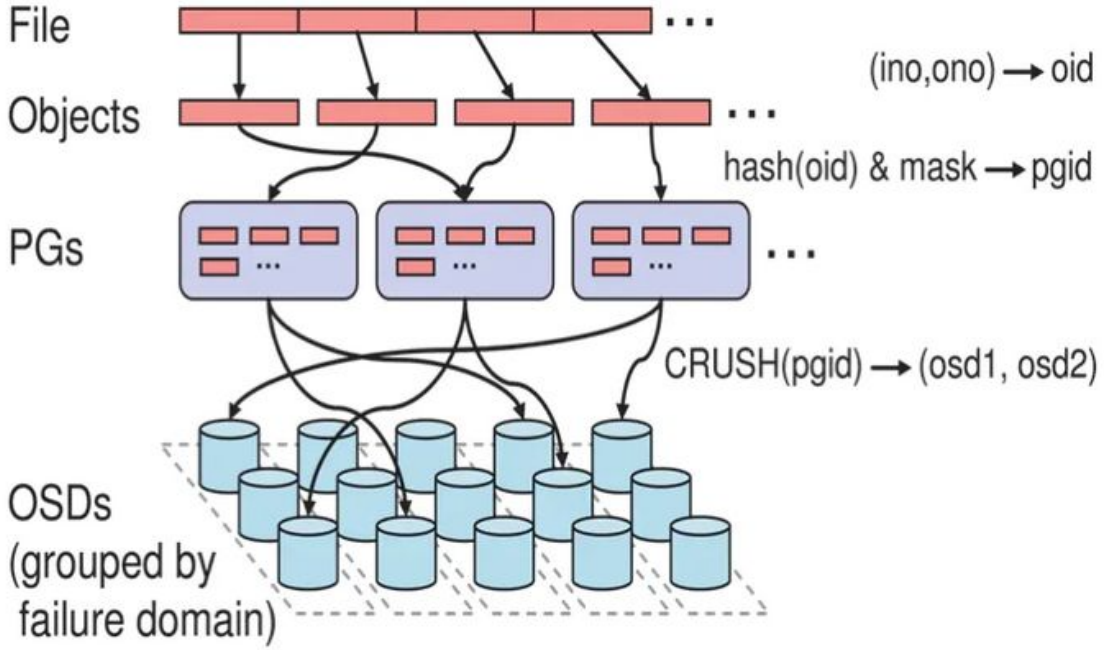
- 一种是为数众多的、负责完成数据存储和维护功能的OSD（Object Storage Device）；
- 另一种则是若干个负责完成系统状态检测和维护的monitor，monitor是一些独立的进程，以及少量的本地存储，monitor之间通过一致性算法保证数据的一致性。

4.2.1 Cluster Map使用

- 集群通过monitor集群操作cluster map来实现集群成员的管理。cluster map 描述了哪些OSD被包含进存储集群以及所有数据在存储集群中的分布。cluster map不仅存储在monitor节点，它被复制到集群中的每一个存储节点，以及和集群交互的client。当因为一些原因，比如设备崩溃、数据迁移等，cluster map的内容需要改变时，cluster map的版本号被增加，map的版本号可以使通信的双方确认自己的map是否是最新的，版本旧的一方会先将map更新成对方的map，然后才会进行后续操作。
- 而RADOS也通过 cluster map来实现这些存储半自动化的功能，cluster map会被复制到集群中的所有部分（存储节点、控制节点，甚至是客户端），并且通过怠惰地传播小增量更新而更新。Cluster map中存储了整个集群的数据的分布以及成员。通过在每个存储节点存储完整的Cluster map，存储设备可以表现的半自动化，通过peer-to-peer的方式（比如定义协议）来进行数据备份、更新，错误检测、数据迁移等等操作。这无疑减轻了占少数的monitor cluster（管理节点组成的集群）的负担。

5 Ceph底层存储过程（Data Placement)

无论使用哪种存储方式（对象，块，文件），存储的数据当底层保存时，都会被切分成一个个大小固定的对象（Objects），对象大小可以由管理员自定义调整，RADOS中基本的存储单位就是Objects，一般为2MB或4MB（最后一个对象大小有可能不同）（文件9M, 4M 4M 1M）（1024TB -> PB ，1024PB -> EB)。



如上图，一个个文件（File）被切割成大小固定的Objects后，将这些对象分配到一个PG（Placement Group）中，然后PG会通过多副本的方式复制几份，随机分派给不同的存储节点（也可指定）。

当新的存储节点（OSD）被加入集群后，会在已有数据中随机抽取一部分数据迁移到新节点，这种概率平衡的分布方式可以保证设备在潜在的高负载下正常工作，更重要的事，数据的分布过程仅需要做几次随机映射，不需要大型的集中式分配表，方便且快速，不会对集群产生大的影响。

5.1 逻辑概念说明

上图两侧逻辑概念词的含义：

- **File：用户需要存储或者访问的文件**
- **Objects：RADOS的基本存储单元，即对象。** Object与上面提到的file的区别是，object的最大size由RADOS限定（通常为2MB或4MB），以便实现底层存储的组织管理。以便实现底层存储的

组织管理。因此，当上层应用向RADOS存入size很大的file时，需要将file切分成统一大小的一系列object进行存储。

- **PG（Placement Group）：英文直译过来即为放置组，所以这里PG作用是对object的存储进行组织和位置映射**，它是一个逻辑概念，在Linux系统中可以直接看到对象，但是无法直接看到PG，它在数据寻址中类似于数据库中的索引。用来放置若干个object（可以数千个甚至更多），**但一个object只能被映射到一个PG中**，即，PG和object之间是“一对多”的映射关系。同时，一个PG会被映射到n个OSD上，而每个OSD上都会承载大量的PG，即，PG和OSD之间是“多对多”映射关系。在实践当中，n至少为2（n代表冗余的份数），如果用于生产环境，则至少为3。一个OSD上的PG则可达到数百个。
- **OSD（object storage device）**：前文已详细说明，不再赘述，不过osd数量关系到系统的数据分布均匀性和性能问题，所以也不能太少。
- **oid 每个object都会有一个唯一的OID，由ino和ono生成。**ino即文件的File ID，用于在全局唯一标识每一个文件，ono则是分片编号（对象编号）。例如：一个文件FileID为A，被切割为对象编号是A0,A1的两个对象。
- pgid 使用静态hash函数对OID做hash去除特征码，用特征码与PG的数量去模，得到的序号即是PGID，由于这种设计方式，所以PG数量的多少会直接决定了数据分布的均匀性，所以需要合理设置PG的数量可以很好的提升CEPH集群的性能并使数据均匀分布。

5.2 存储过程中各层次之间的映射关系

- **file -> object**
  object的最大size是由RADOS配置的，当用户要存储一个file，需要将file切分成若干个object。
- **object -> PG**
  每个object都会被映射到一个PG中，然后以PG为单位进行副本备份以及进一步映射到具体的OSD上。
- **PG -> OSD**
  通过CRUSH算法来实现，根据用户设置的冗余存储的个数r，PG会最终存储到r个OSD上。

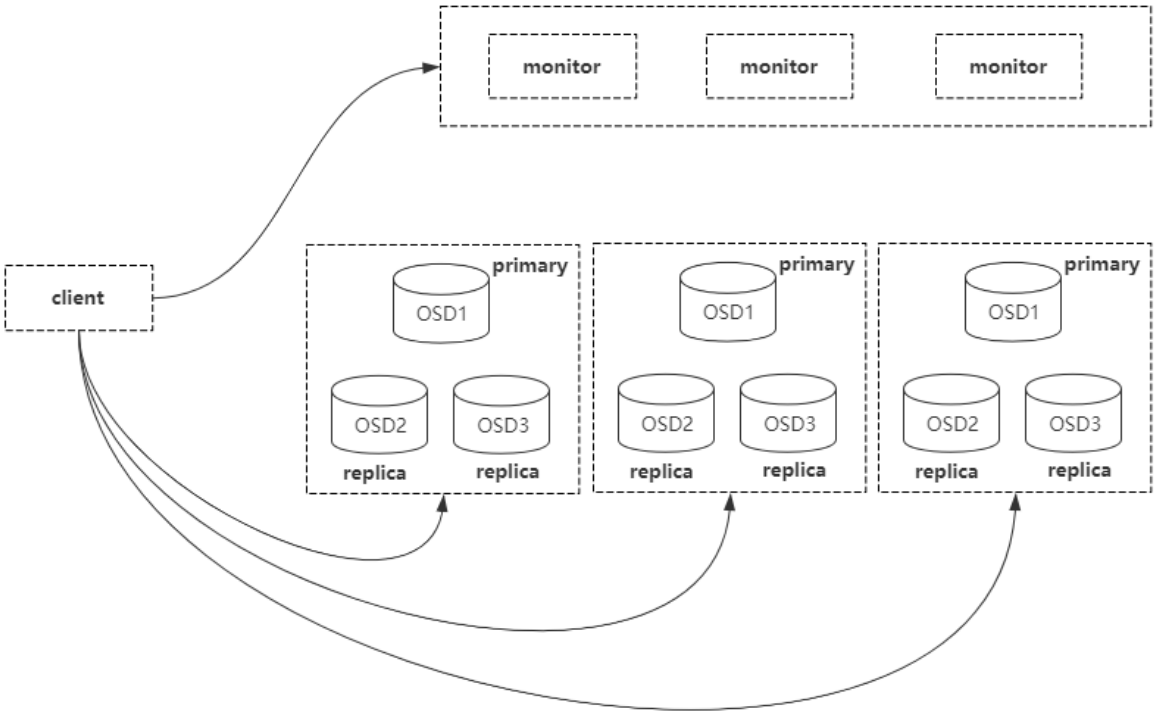
5.3 存储过程具体说明

用户通过客户端存储一个文件时，在RAODS中，该File（文件）会被分割为多个2MB/4MB大小的**Objects（对象）**。而每个文件都会有一个文件ID，例如A，那么这些对象的ID就是A0,A1,A2等等。然而在分布式存储系统中，有成千上万个对象，只是遍历就要花很久时间，所以对象会先通过hash-取模运算，存放到一个PG中。
PG相当于数据库的索引（PG的数量是固定的，不会随着OSD的增加或者删除而改变），这样只需要首先定位到PG位置，然后在PG中查询对象即可。之后PG中的对象又会根据设置的副本数量进行复制，并根据CRUSH算法存储到OSD节点上。

5.4 为什么引入PG概念?

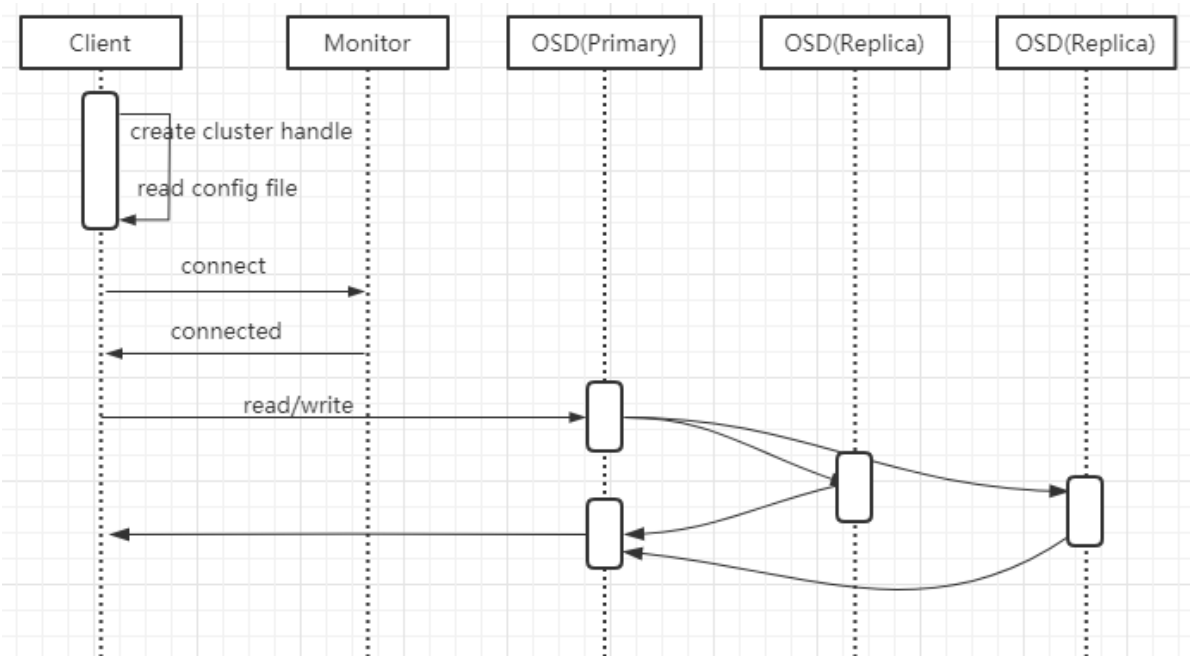
因为Object对象的size很小，并不会直接存储进OSD中，在一个大规模的集群中可能会存在几百到几千万个对象，这么多对象如果遍历寻址，那速度是很缓慢的，并且如果将对象直接通过某种固定映射的哈希算法映射到osd上，那么当这个osd损坏时，对象就无法自动迁移到其他osd上（因为映射函数不允许）。为了解决这些问题，ceph便引入了归置组的概念，即PG。
最后PG会根据管理设置的副本数量进行副本级别的复制，然后通过CRUSH算法存储到不同的osd上（其实是把PG中的所有对象存储到节点上），第一个osd节点即为主节点，其余均为从节点。

6 Ceph IO流程及数据分布



线上应用，分布式文件系统，建议3个存储，存3份。

6.1 正常IO流程图

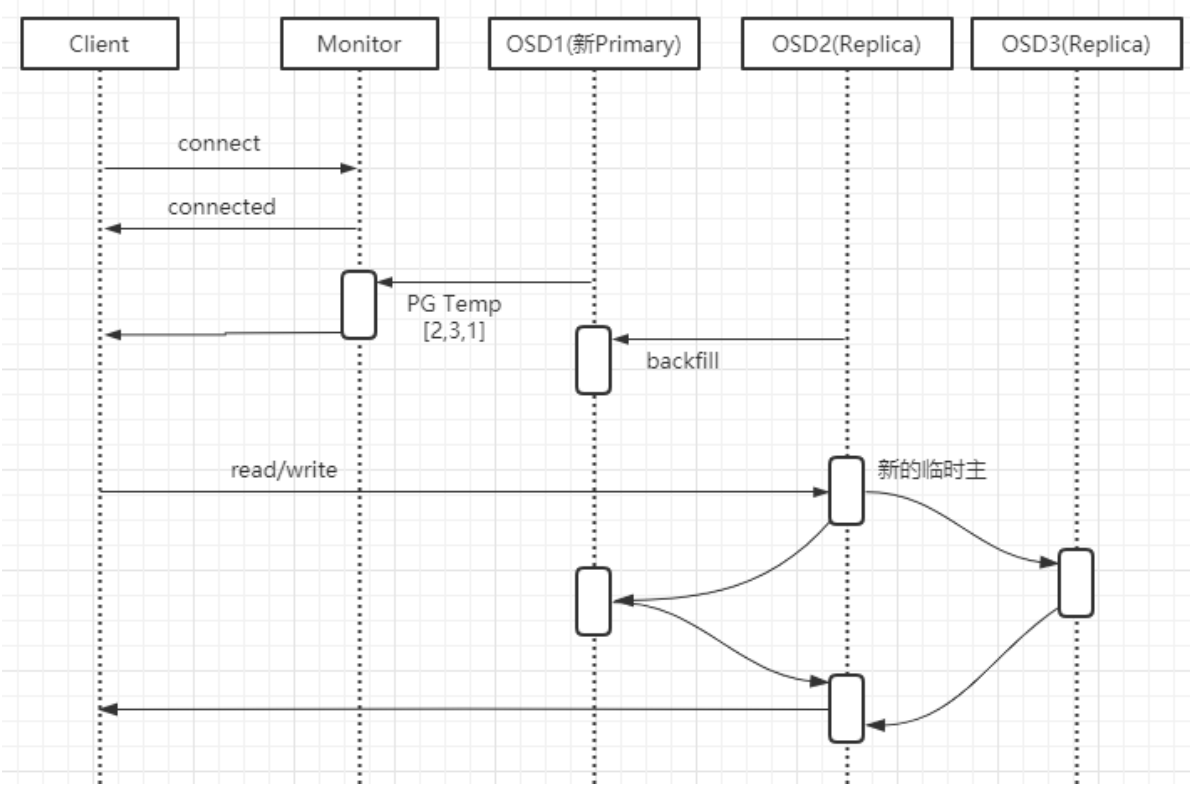


步骤：

1. client 创建cluster handler。
2. client 读取配置文件。
3. client 连接上monitor，获取集群map信息。
4. client 读写io 根据crshmap 算法请求对应的主osd数据节点。
5. 主osd数据节点同时写入另外两个副本节点数据。
6. 等待主节点以及另外两个副本节点写完数据状态。
7. 主节点及副本节点写入状态都成功后，返回给client，io写入完成。

6.2 新主IO流程图

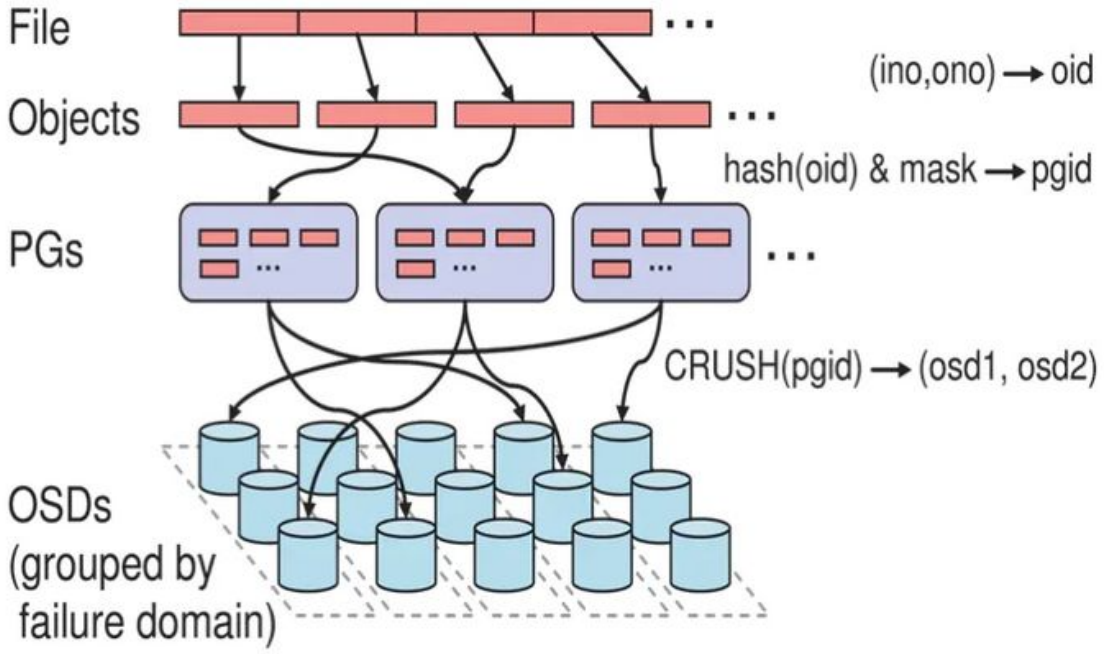
说明：
如果新加入的OSD1取代了原有的 OSD成为 Primary OSD, 由于 OSD1 上未创建 PG , 不存在数据，那么 PG 上的 I/O 无法进行，怎样工作的呢?



步骤：

1. client连接monitor获取集群map信息。
2. 同时新主osd1由于没有pg数据会主动上报monitor告知让osd2临时接替为主。
3. 临时主osd2会把数据全量同步给新主osd1。
4. client IO读写直接连接临时主osd2进行读写。
5. osd2收到读写io，同时写入另外两副本节点。
6. 等待osd2以及另外两副本写入成功。
7. osd2三份数据都写入成功返回给client, 此时client io读写完毕。
8. 如果osd1数据同步完毕，临时主osd2会交出主角色。
9. osd1成为主节点，osd2变成副本。

6.3 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)。

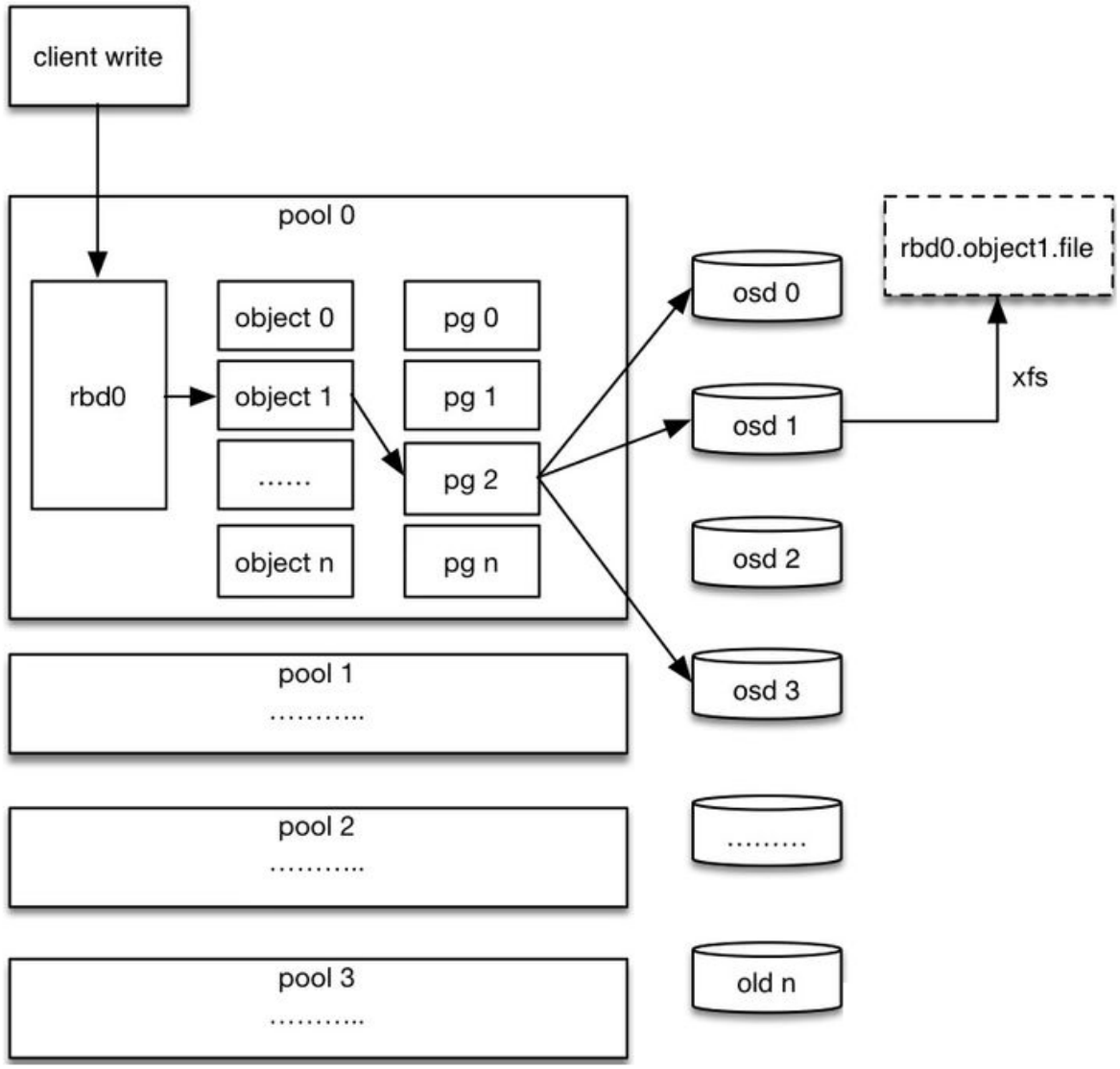
2. Object是RADOS需要的对象。Ceph指定一个静态hash函数计算oid的值，将oid映射成一个近似均匀分布的伪随机值，然后和mask按位相与，得到pgid。Object->PG映射：
    - a. hash(oid) & mask-> pgid 。
    - b. mask = PG总数m(m为2的整数幂)-1 。

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) 。

6.4 Ceph IO伪代码流程

```
locator = object_name
obj_hash = hash(locator)
pg = obj_hash % num_pg
osds_for_pg = crush(pg) # returns a list of osds
primary = osds_for_pg[0]
replicas = osds_for_pg[1]
```

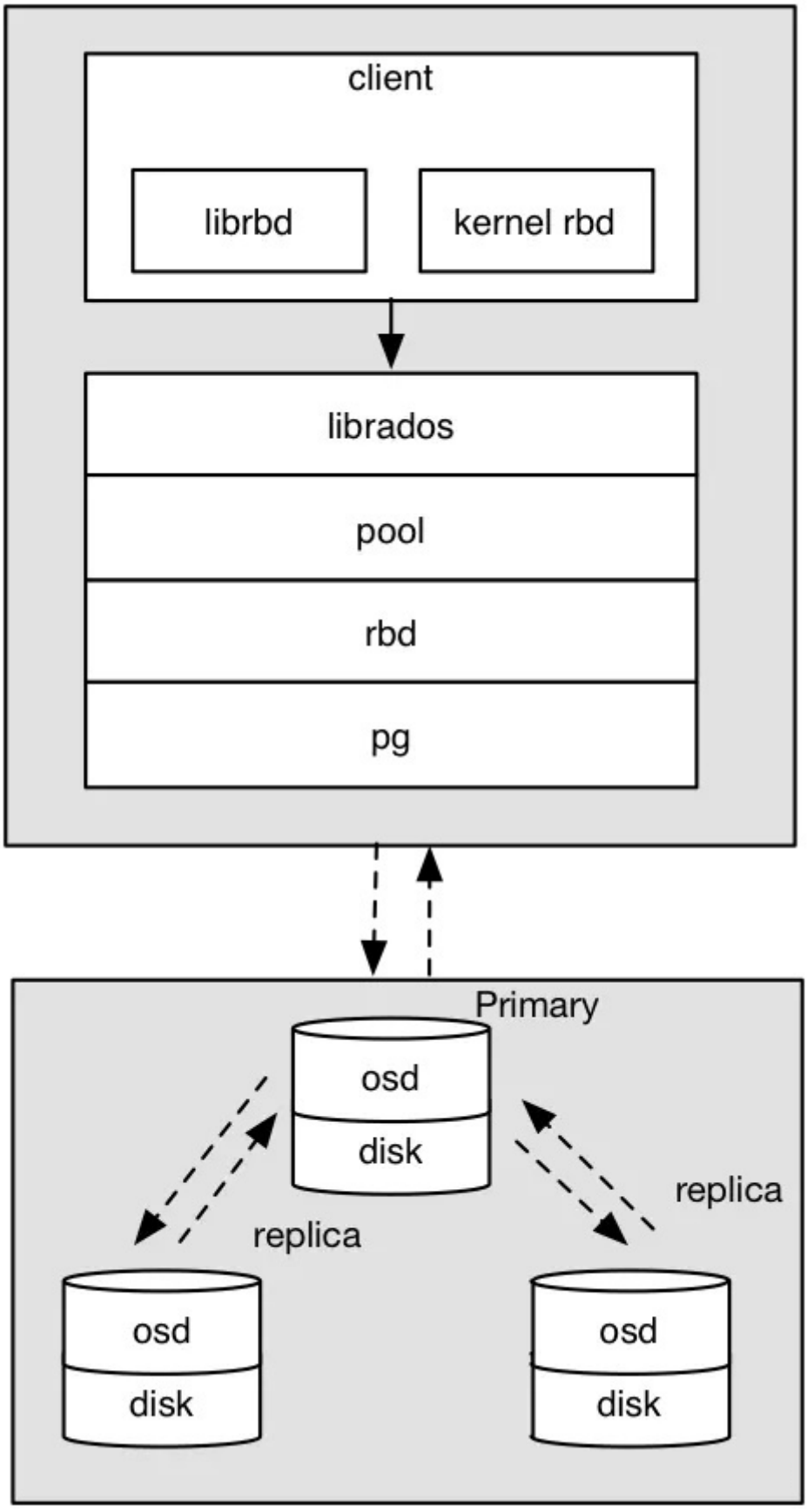
6.5 Ceph RBD IO流程



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。

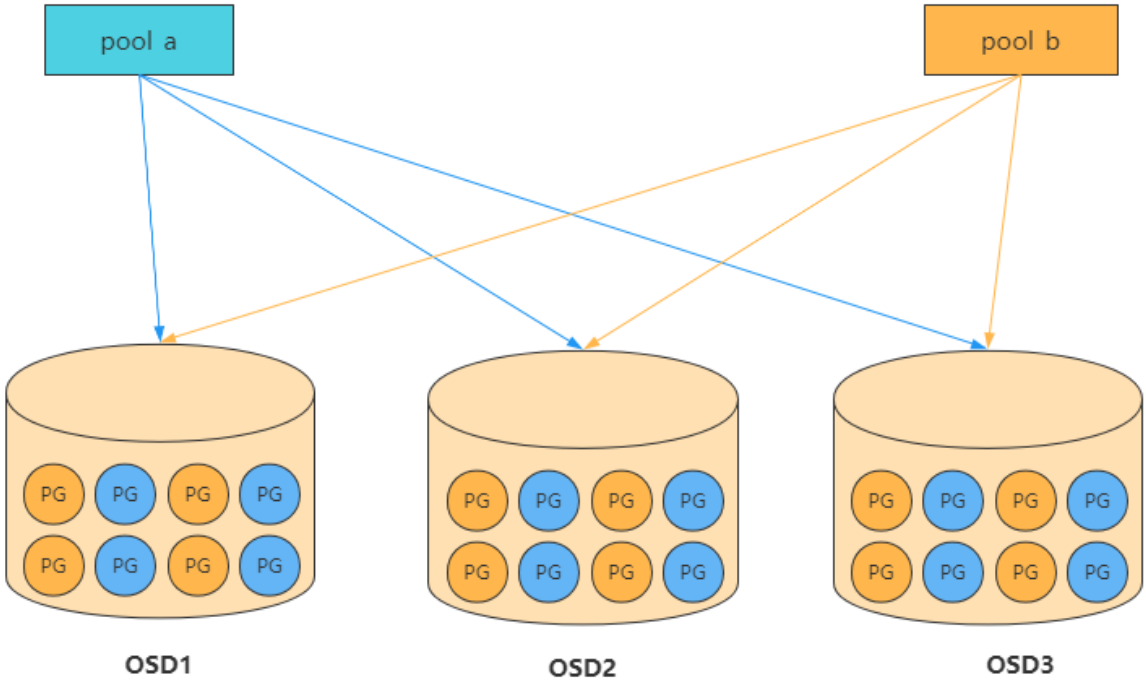
6.6 Ceph RBD IO框架图



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。
3. 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。

6.7 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。

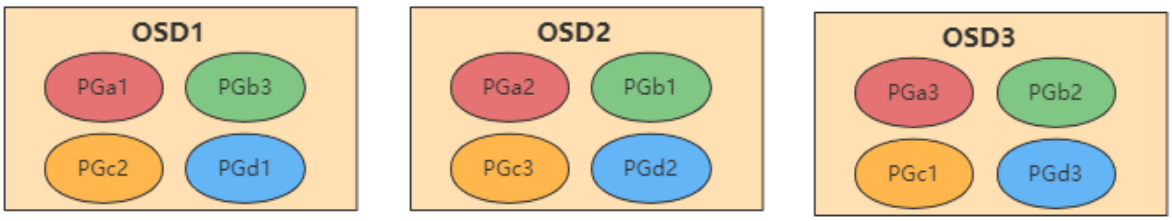
6.8 Ceph 数据扩容PG分布

每个PG对应一个主分区和两个备份分区。

场景数据迁移流程：

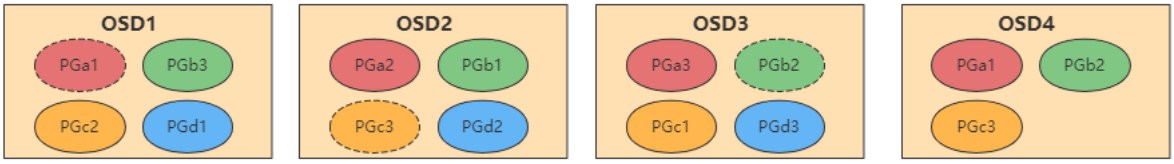
- 现状3个OSD, 4个PG
- 扩容到4个OSD, 4个PG

现状：



PGa -> osd1 2 3
PGb -> osd1 2 3
PBc -> osd1 2 3
PBd -> osd1 2 3

扩容后：



PGa -> osd 2 3 4
PGb -> osd 1 2 4
PBc -> osd 1 3 4
PBd -> osd 1 2 3

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。

7 Ceph,TFS,FastDFS,MogileFS,MooseFS,GlusterFS 对比

| 对比说明

/文件系统	TFS	FastDFS	MogileFS	MooseFS	GlusterFS	Ceph
开发语言	C++	C	Perl	C	C	C++
开源协议	GPL V2	GPL V3	GPL	GPL V3	GPL V3	LGPL
数据存储方式	块	文件/Trunk	文件	块	文件/块	对象/文件/块
集群节点通信协议	私有协议（TCP）	私有协议（TCP）	HTTP	私有协议（TCP）	私有协议（TCP）/ RDAM(远程直接访问内存)	私有协议（TCP）
专用元数据存储点	占用NS	无	占用DB	占用MFS	无	**占用MDS（元数据服务）**
在线扩容	支持	支持	支持	支持	支持	支持
冗余备份	支持	支持	-	支持	支持	支持
单点故障	存在	存在	存在	存在	存在	存在
跨集群同步	支持	部分支持	-	-	支持	不适用
易用性	安装复杂，官方文档少	安装简单，社区相对活跃	-	安装简单，官方文档多	安装简单，官方文档专业化	安装简单，官方文档专业化
适用场景	跨集群的小文件	单集群的中小文件	-	单集群的大中文件	跨集群云存储	单集群的大中小文件

8 参考

块存储、文件存储、对象存储意义及差异 https://www.cnblogs.com/hukey/p/8323853.html
“Ceph浅析”之三——Ceph的设计思想 https://zhuanlan.zhihu.com/p/56087062
“CEPH浅析”系列
关于存储技术的最强入门科普
https://mp.weixin.qq.com/s?__biz=MzI1NTA0MDUyMA==&mid=2456671031&idx=1&sn=6dab8b39023f4586ba628a8a261a7770&chksm=fda57e50cad2f746cca855d6d3d4b1c2ab9cd56a5d2fc7b882358bf5c63b29f9165e1992b

8de&scene=21#wechat_redirect

一篇文章让你理解Ceph的三种存储接口(块设备、文件系统、对象存储)
https://blog.csdn.net/wangmingshuaiguo/article/details/92628036

https://mp.weixin.qq.com/s/tf8vkZ0DAqOe_6eA3NoU3w