CEPH分布式集群调研

## 简介

Ceph 独一无二地用统一的系统提供了对象、块、和文件存储功能，它通过C++编写并提供python、C、C++、java、php调用，它可靠性高、管理简便、并且开源，可以进行二次开发。Ceph 可提供极大的伸缩性——供成千用户访问 PB 乃至 EB 级的数据。 Ceph 节点以普通硬件和智能守护进程作为支撑点， Ceph 存储集群组织起了大量节点，它们之间靠相互通讯来复制数据、并动态地重分布数据。

### 架构介绍

存储集群包含两种类型的守护进程：OSDs（OSD的守护进程）和Ceph监视器。Ceph 监视器维护着集群运行图的主副本，而且监视器也是集群，以确保去中心化；Ceph OSD 守护进程检查自身状态、以及其它 OSD 的状态，并报告给监视器们。当存取数据时，客户端和OSD各个守护进程都会通过Crush算法去寻址文件对象，而不是依赖于中心化查询表，这一块画个图：

MDS（可选）

DISK

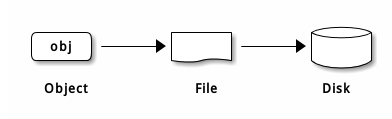
Monitors

OSDS

Client

### 数据存储

Ceph 存储集群从 Ceph 客户端接收数据——不管是来自 Ceph 块设备、 Ceph 对象存储、 Ceph 文件系统、还是基于 librados 的自定义实现—并存储为对象。每个对象是文件系统中的一个文件，它们存储在对象存储设备上。由 Ceph OSD 守护进程处理存储设备上的读/写操作。



### 数据存储过程

上面提到了任何类型数据过来都会成为Object，那么Object是怎么映射到OSD里面呢，简单通过图来梳理一下这个过程：



解释一下算法：

Ceph为了保存一个对象，对上构建了一个逻辑层，也就是池(pool)，用于保存对象，这个池的翻译很好的解释了pool的特征，如果把pool比喻成一个中国象棋棋盘，那么保存一个对象的过程就类似于把一粒芝麻放置到棋盘上。Pool再一次进行了细分，即将一个pool划分为若干的PG(归置组 Placement Group)，这类似于棋盘上的方格，所有的方格构成了整个棋盘，也就是说所有的PG构成了一个pool。

1. 计算PGID：其作用就类似于找到棋盘里面的方块，对于每一个对象的hash值都是确定的，将一个对象hash之后其实问题就转为了数字计算了，计算PGID其实就是通过poolId+HASH后的数字和归置组进行求余，从而找到我当前对象的存储位置，这样做具有随机性，随机的理解就是当样本足够大，那么数据分布式趋于平均的。
2. 通过计算PGID，我们知道了数据存储的块，但是一个块里面有很多OSD，也就是物理磁盘位置了，我们要通过CRUSH算法来实现这个寻址过程，简单讲一下我对这个算法的理解：我们拥有不同的PGID，和不同的OSDID当然还得有OSD的存储容量，这个称为权重，这个算法简单来说就是优先选择大容量的OSD，但避免总去选择最大的OSD，这就体现了权重的重要性了。

CRUSH\_HASH( PG\_ID, OSD\_ID, r ) =⇒ draw

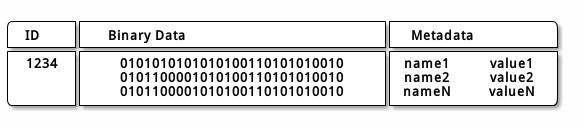
( draw &0xffff ) \* osd\_weight =⇒ osd\_straw

pick up high\_osd\_straw

解决一个PG映射到多个OSD的问题，还记得那个常量r吗？我们把r+1，再求一遍随机数，再去乘以每个OSD的权重，再去选出乘积最大的OSD，如果和之前的OSD编号不一样，那么就选中它，如果和之前的OSD编号一样的话，那么再把r+2，再次选一次，直到选出我们需要的三个不一样编号的OSD为止！

### 数据存储大小影响

Ceph OSD 在扁平的命名空间内把所有数据存储为对象（也就是没有目录层次）。对象包含一个标识符、二进制数据、和由名字/值对组成的元数据，元数据语义完全取决于 Ceph 客户端。例如， CephFS 用元数据存储文件属性，如文件所有者、创建日期、最后修改日期等等。



### 扩容方式

方案一：

Ceph支持在一个已有的集群中增加一个带有一组磁盘的新节点来扩展其容量，而且在服务不宕机的情况下进行扩展。操作过程如下：

1. 在新节点上安装ceph软件包，保证和ceph集群的软件版本一致

2. 列出新节点上所有可用磁盘

3. 将2中查询出来的磁盘加入到ceph集群，当新的节点加入集群，ceph集群开始将部分现有的数据重新平衡到新加入的OSD上。（IO操作）

注：在生产环境中，一般不会再新节点加入ceph集群后，立即开始数据回填，这样会影响集群性能。所以我们需要设置一些标志位，来完成这个目的。

上面的方案到了巨型数据的时候，可能出现卡顿，当存在MDS时，也会导致MDS卡掉的问题。

方案二：

我们可以考虑部署多套集群，并且来真正的实现了数据处理能力的横向扩展，因为MDS，可以是多个的了，那么比较重要的问题就是统一命名空间的问题了，但是提前规划好命名空间，比如按年来决定文件存储集群，这样避免了大量的数据迁移，也更适合海量小文件的场景。

### 单点故障

Ceph 客户端读或写数据前必须先连接到某个 Ceph 监视器、获得最新的集群运行图副本。一个 Ceph 存储集群只需要单个监视器就能运行，但它就成了单一故障点（即如果此监视器宕机， Ceph 客户端就不能读写数据了），为增强可靠性和容错能力， Ceph 支持监视器集群；在一个监视器集群内，延时以及其它错误会导致一到多个监视器滞后于集群的当前状态，因此， Ceph 的各监视器例程必须就集群的当前状态达成一致。Ceph 总是使用大多数监视器（如： 1 、 2:3 、 3:5 、 4:6 等等）和Paxos 算法就集群的当前状态达成一致。

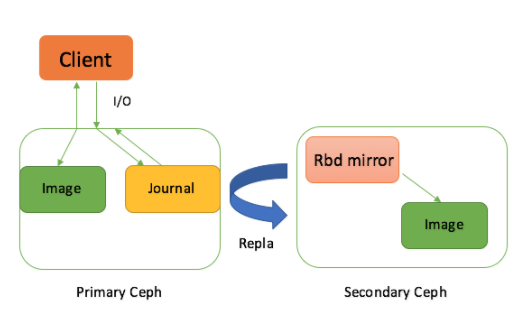
### 备份机制

和 Ceph 客户端一样， OSD 也用 CRUSH 算法，但用于计算副本存到哪里（也用于重均衡）。一个典型的写情形是，一客户端用 CRUSH 算法算出对象应存到哪里，并把对象映射到存储池和归置组，然后查找 CRUSH 图来确定此归置组的主 OSD 。

客户端把对象写入目标归置组的主 OSD ，然后这个主 OSD 再用它的 CRUSH 图副本找出用于放对象副本的第二、第三个 OSD ，并把数据复制到适当的归置组所对应的第二、第三 OSD （要多少副本就有多少 OSD ），最终，确认数据成功存储后反馈给客户端。

### 同步策略

Ceph是一个数据强一致性的分布式存储系统，只有在所有副本都写入的情况下才认为一个写操作完成，所以在做跨地域式的数据同步延迟非常高。为解决这种情况Ceph在2016年发布的Jewel版本中提供了一种异步同步机制允许不同集群之间复制块设备。



1. 当有一个写入操作，会先写入日志；

2. 然后知会客户端，再写入rbd image 中；

3. 这时远端rbd-mirror进程将会照本地日志写入远端的rbd image。

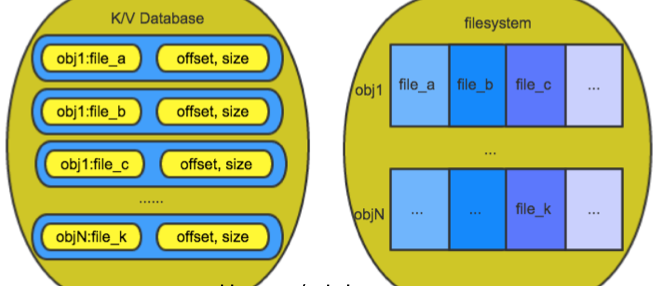
以上就是一个rbd-mirror工作周期内的流程（如图1），在现有的Jewel版本中30s为一次工作周期，暂时不能改变这个周期时间。

虽然在2016 年Ceph已经推出可以用于生产环境的支持rbd-mirror的发行版本。但是目前还是存在一些不容忽视的问题。比如网络问题，两个跨地域的集群数据同步必然对网络问题有极大挑战。再者比如，现有的版本暂时只支持一个集群一个守护进程的形式，不可否认这样的单进程形式即使在多线程并发工作的情况下也是缺乏一定的灵活性，对性能造成一定影响。

再者Ceph作为分布式存储管理的后端系统，被用于与各种类型云一同协作，特别是Openstack。

### 小文件场景

海量小文件的问题是海量小文件的元数据信息组织与管理以及本地磁盘文件的存储与管理，对于前者问题CEPH通过CRUSH算法实现无中心点查询，而减轻统一管理的压力，但是对于存储数据来说，CEPH会把文件转为对象，大文件会切分为4M大小的对象，而小文件不足4M就会封装成对象，读取一个文件通常需要三次磁盘IO，官方未解决这个问题，论坛通过多文件同一对象的方式解决了海量小文件的存储问题，解决思路如下：



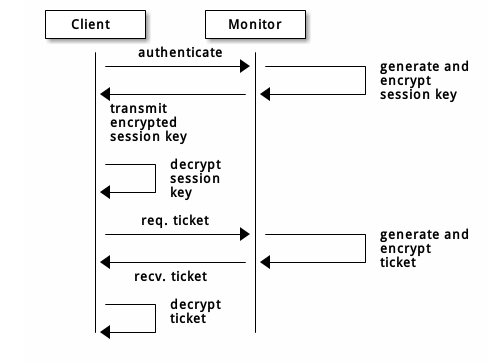
将若干小文件合并存储在RADOS系统的一个对象(object)中，<小文件的名字、小文件在对象中的offset及小文件size>组成kv对，作为相应对象的扩展属性(或者omap，本文以扩展属性表述，ceph都使用kv数据库实现，如leveldb)进行存储，如下图所示，对象的扩展属性数据与对象数据存储在同一块盘上。对于写操作，一次小文件写操作对应两次本地磁盘随机io(逻辑层面)，且不能更少，某些kv数据库(如leveldb)还存在write amplification问题，对于写压力大的业务场景，此方案不能很好地满足；不过对于读操作，我们可以通过配置参数，尽量将kv数据保留在内存中，实现读取操作一次磁盘io的预期目标。

### 安全认证

为识别用户并防止中间人攻击， Ceph 用 cephx 认证系统来认证用户和守护进程。

第一步：admin生成用户：client.admin 用户从命令行调用 ceph auth get-or-create-key 来生成一个用户及其密钥， Ceph 的认证子系统生成了用户名和密钥、副本存到监视器然后把此用户的密钥回传给 client.admin 用户，也就是说客户端和监视器共享着相同的密钥。

第二步：认证：客户端得把用户名传给监视器，然后监视器生成一个会话密钥、并且用此用户的密钥加密它，然后把加密的凭证回传给客户端，客户端用共享密钥解密载荷就可获取会话密钥。会话密钥在当前会话中标识了此用户，客户端再用此会话密钥签署过的用户名请求一个凭证，监视器生成一个凭证、用用户的密钥加密它，然后回传给客户端，客户端解密此凭证，然后用它签署连接集群内 OSD 和元数据服务器的请求。



### 吞吐量以及性能瓶颈

对于大文件可以切分为不同的4M（默认）大小的对象数据，可以分别存在不同的物理磁盘上，实现并发读取，吞吐量不会受IO影响，但对于小文件可能这个有点并不能够适用，上面提到了小文件IO三次的问题，虽然可以合成大文件，但是这样就只能存储在一个物理磁盘上。上面提到了缓存kv数据从而尽可能实现一次io，从而也能提高吞吐量，从更多的角度来看，这个方案的性能瓶颈在带宽上。

### 其他

容灾方面，在存文件的时候配置备份数量能够实现跨机房的数据备份，从而实现跨机房的容灾能力。

**数据重均衡**

你向 Ceph 存储集群新增一 OSD 守护进程时，集群运行图就要用新增的 OSD 更新。回想计算 PG ID ，这个动作会更改集群运行图，因此也改变了对象位置，因为计算时的输入条件变了。下面的图描述了重均衡过程（此图很粗略，因为在大型集群里变动幅度小的多），是其中的一些而不是所有 PG 都从已有 OSD （ OSD 1 和 2 ）迁移到新 OSD （ OSD 3 ）。即使在重均衡中， CRUSH 都是稳定的，很多归置组仍维持最初的配置，且各 OSD 都腾出了些空间，所以重均衡完成后新 OSD 上不会出现负载突增。

