# RADOS

参考：http://blog.csdn.net/it_yuan/article/details/8966065

## 概念

RADOS是一个支持海量存储对象的分布式对象存储系统。RADOS最大的特色是使用CRUSH算法维护存储对象与存储服务器的对应关系，并采用了无Master设计， 对象复制，集群扩容，数据迁移，故障检测和处理能复杂功能由OSD提供， 既避免了单点失败，又提升了集群扩展性。

## 数据定位

对象映射关系： object ------> PGs ------> OSDS

### object

Ceph的基本的概念就是对象object，在ceph的rados概念中，一个对象就是一个文件系统中的文件。

#### 定位Oject的实际存放位置

```
[root@vmlq-2 ~]# ceph osd map lqing obj_test
osdmap e45 pool 'lqing' (1) object 'obj_test' -> pg 1.8810ba5b (1.5b) -> up ([2,1], p2) acting ([2,1], p2)
```

上述的信息显示：obj_test 属于的pg是1.5b 对应的指纹是8810ba5b，该PG对应到OSD 1和OSD 2

登录到OSD1 或者 OSD2 的存储目录，这里是/var/local/osd1/
在以下路径可以找的到对应的文件：

```
[root@vmlq-3 1.5b_head]# pwd
/var/local/osd1/current/1.5b_head
```

在对应的PG文件夹中，通过指纹能够找到object对象

```
[root@vmlq-3 1.5b_head]# ll
total 8
-rw-r--r--. 1 ceph ceph 0 Sep 11 05:08 __head_0000005B__1
-rw-r--r--. 1 ceph ceph 6 Sep 12 07:53 obj\utest__head_8810BA5B__1
```

```
[root@vmlq-3 1.5b_head]# cat obj\\utest__head_8810BA5B__1
Hello
```

#### 定位Image的位置

红框就是Image对应的Object

```
[root@vmlq-3 1.5b_head]# rbd info lqing/test
rbd image 'test':
        size 1024 MB in 256 objects
        order 22 (4096 kB objects)
        block_name_prefix: rbd_data.10bf2ae8944a
        format: 2
        features: layering
        flags:
```

之后方式和定位oject相同

```
[root@vmlq-3 1.5b_head]# rados -p lqing ls
rbd_directory
rbd_id.test2
rbd_header.10b02ae8944a
rbd_header.10bf2ae8944a
rbd_id.test
obj_test
```

可以注意到image的文件已rbd开头

```
[root@vmlq-2 1.2b_head]# ll
total 8
-rw-r--r--. 1 ceph ceph 0 Sep 11 05:08 __head_0000002B__1
-rw-r--r--. 1 ceph ceph 0 Sep 11 07:19 rbd\uheader.10b02ae8944a__head_8DDE2A2B__1
-rw-r--r--. 1 ceph ceph 0 Sep 11 07:43 rbd\uheader.10bf2ae8944a__head_12DA936B__1
```

**PG**

参考：http://qujunorz.blog.51cto.com/6378776/1861136

**特点**

1. 一组存储对象集合， 是复制和负载均衡的最基本单位。
2. PG 确定了 pool 中的对象和 OSD 之间的映射关系。一个 object 只会存在于一个 PG 中，但是多个 object 可以在同一个 PG 内。
3. Pool 的 PG 数目是创建 pool 时候指定的，Ceph 官方有推荐的计算方法。其值与 OSD 的总数的关系密切。当Ceph 集群扩展 OSD 增多时，根据需要，可以增加 pool 的 PG 数目。
   http://ceph.com/pgcalc/
   简单技术公式：
   Total PGs = ((Total_number_of_OSD * 100) / max_replication_count) / pool_count
   结算的结果往上取靠近2的N次方的值。比如总共OSD数量是160，复制份数3，pool数量也是3，那么按上述公式计算出的结果是1777.7。取跟它接近的2的N次方是2048，那么每个pool分配的PG数量就是2048。
   注意pgp的数目要和pg一致

```
# 获取Pool中的pg_num和pgp_num
ceph osd pool get <pool name> pg_num
ceph osd pool get <pool name> pgp_num
# 设定Pool的pg_num和pgp_num
ceph osd pool set <pool name>pg_num 256
ceph osd pool set <pool name> pgp_num 256
```

1. PG 和 OSD 之间的映射关系由 CRUSH 决定，而它做决定的依据是 CRUSH 规则（rules）。CRUSH 将所有的存储设备（OSD）组织成一个分层结构，该结构能区分故障域（failure domain），该结构中每个节点都是一个 CRUSH bucket。详细情况请阅读 CRUSH 相关的文档。
2. 关于PG的ID
   PG的由两个部分组成：.，创建Pool时指定了多少PG，就会生成多少。

### OSD和PG的动态关系

当有OSD加入，或者有OSD退出时，CRUSH会动态移动OSD上的PG

**OSD**

**Pool**

pool是ceph存储数据时的逻辑分区，它起到namespace的作用。每个pool包含一定数量的PG，PG里的对象被映射到不同的OSD上，因此pool是分布到整个集群的。除了隔离数据，我们也可以分别对不同的POOL设置不同的优化策略，比如副本数、数据清洗次数、数据块及对象大小等。
参考：http://www.cnblogs.com/vincenshen/articles/6284384.html

Pool可以设定以下内容：设定副本数目、设定PG数目、设定CRUSH Rules、进行快照、设定Ownership

### Pool增强数据可用性

1. 通过副本
   设定Pool的副本数

```
ceph osd pool set <pool name> size <num>
```

1. 通过EC（erasure coding 纠错码）
   纠错码型 pool（类似于 Software RAID）。在这种 pool 中，每个数据对象都被存放在 K+M 个数据块中：对象被分成 K 个数据块和 M 个编码块；pool 的大小被定义成 K+M 块，每个块存储在一个 OSD 中；块的顺序号作为 object 的属性保存在对象中。
   EC将原始数据分为K份，之后通过Ceph Erasure算法生成K+M个新数据，任意K个新数据都可以生成原始数据，因此对于EC类型的Pool，有M个OSD损坏时数据任然不丢失。
   这种 pool 用更少的空间实现存储，即节约空间；纠删码实现了高速的计算，但有2个缺点，一个是速度慢，一个是只支持对象的部分操作（比如：不支持局部写）。

### 创建Pool的快照

```
# 生成快照
rados mksnap <snapshot name> -p <pool name>
# 查看快照
rados lssnap -p <pool name>
# 从快照里回滚某个object
rados rollback -p <pool> <object> <snapshot>
```

### object ------> PGs 是的第一级映射

pgid= hash(o) &m
输入包括：
1.对象标识o（oid）
2.掩码m，一般情形m为PG数目减一，PG数目一般为2的整数幂

### CRUSH Rules

数据映射的策略。系统默认提供 “replicated_ruleset"。
用户可以自定义策略来灵活地设置 object 存放的区域。比如可以指定 pool1中所有objecst放置在机架1上，所有objects的第1个副本放置在机架1上的服务器A上，第2个

副本分布在机架1上的服务器B上。 pool2中所有的object分布在机架2、3、4上，所有Object的第1个副本分布在机架2的服务器上，第2个副本分布在机架3的服 器上，第3个副本分布在机架4的服务器上。