# 第一章 Ceph分布式存储发展趋势

## Ceph发展历程

## Ceph与其他几种分布式存储的比较

比较架构

## 国内外Ceph的案例和现状

CERN

目前存储厂商及市场占有率

分布式存储使用场景

# 第二章 分布式存储概述

## 单机存储系统

## 分布式存储系统

分布式概念

## 分布式存储 VS 传统存储

# 第三章 Ceph 体系结构

## 集群大脑monitor

mon作用

paxos算法实现

## 存储基石 osd

## 接口管家 mgr

## 对象存储网关 rgw

## 分布式文件系统 mds

## 资源池pool

## 归置组pg

 pg作用

~~pg状态切换~~

 pg实现

 数据校验 scrub

## 后端存储 objectstore

filestore

bluestore （扩展seastore）

## 定位算法

hash算法

计算寻址 CRUSH

crush 规则与算法

crush 算法实践

## Ceph IO流程及数据分布

## Ceph的认证机制cephx

认证机制简介

认证方式

认证配置

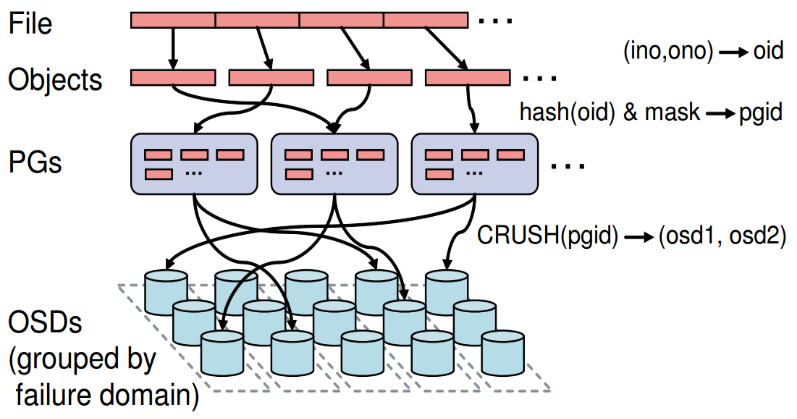
# 第四章 Ceph 状态迁移

## 4.1 PG简介

### 4.1.1 何为PG

PG（Placement Group），简单而言就是客户端数据存储的容器，在filestore里对应的就是本地文件系统的一个子目录，它是一个逻辑概念，类似于一致性hash算法中引入的虚拟节点；它处于客户端到具体磁盘（OSD）中间，起一个承上启下的作用，对客户端可以屏蔽底层磁盘，数据的存储只与PG有关，只要PG状态正常就可以存取数据。PG通过二次映射到OSD中。

以下为Ceph寻址流程示意图：



从上图可以PG所处的位置，以及其上下关系。

通过引入PG，可以带来如下好处：

* 减小集群扩缩容时带来的数据迁移问题；
* 减少hash分布带来的数据分布不均衡问题；
* 在数据迁移或异常恢复时，所有磁盘都可参与，避免出现热点磁盘，提高效率。

#### 4.1.1.1 PG与OSD的关系

PG的命名规则为：

***{pool-id}*.*{pg-id}***

其中，

* pool-id，即存储池的ID，十进制表示；
* pg-id采用16进制，其取值范围为[0，pg\_num-1] 。

PG与OSD的映射关系可以通过如下命令查询：

|  |
| --- |
| [root@node0 ~]# ceph pg map 1.2d  osdmap e385 pg 1.2d (1.2d) -> up [5,6,1] acting [5,6,1] |

从查询结果来看，与PG相关的OSD集合包括up和acting两个，其中acting表示集合内的OSD均处于活跃状态，并可以处理PG相关的请求，up表示包含有该PG，并且是处于up状态的OSD。正常情况下PG的up和acting两个集合中的OSD保持一致，且第一个为主OSD。

#### 4.1.1.2 PG的特点

从应用的角度来看，PG具有如下特点：

* 以存储池（pool）为单位进行设置，在创建存储池时即指定数目，pg的命名中包含有pool的id信息；
* 存储池的特性，如容错策略、分布策略、数据校验、容灾恢复等均依托PG来实现；
* PG的数量一般设置为2的N次方，支持增加，一般不修改，在N版本之前不支持减少；
* 集群内所有存储池PG之和一般设置为OSD数量的100倍，在实践中也常设置为200倍左右；
* PG在OSD中无法做到完全均匀分布，目前可通过ceph-balancer进行自动调整优化；
* 在同一个存储池中，各PG承载的数据块数量基本相同，空间使用也基本相同。

PG的命名规则

### 4.1.3 与PG相关的操作

#### 4.1.3.1 常用的PG查询命令

|  |  |
| --- | --- |
| **查询目的** | **命令行** |
| 查询pg整体状态 | ceph -s  ceph pg stat |
| 列出所有pg | ceph pg ls |
| 按池列出所有pg | ceph pg ls-by-pool <pool\_name> |
| 按OSD列出所有pg | ceph pg ls-by-osd <osdname|id> |
| 列出指定OSD上的主PG | ceph pg ls-by-primary <osdname|id> |
| 设置某存储池PG数量 | ceph osd pool set <poolname> pg\_num xxxx |
| 查询某存储池PG数量 | ceph osd pool get <poolname> pg\_num |
| 获取集群PG的统计信息 | ceph pg dump |
| 列出异常PG | ceph pg dump\_stuck {inactive|unclean|stale|undersized|degraded} |
| 查询指定PG查询 | ceph pg <pg\_id> query  ceph tell <pg\_id> query |
| 查询PG到OSD的映射关系 | ceph pg map <pg\_id> |
| 获取所有PG信息 | ceph pg dump |
| 离线查询OSD上所有PG | ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-0/ --op list-pgs |
| 离线查询OSD上某个PG的信息 | ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-0/ --pgid 1.3c --op info |
| 离线导出OSD上某个PG的内容 | ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-0/ --pgid 1.3c --op export --file 1.3c |
| 离线导入某个PG到OSD中 | ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-0/ --pgid 1.3c --op import --file 1.3c |
| 列出指定PG中不连续的对象 | rados list-inconsistent-obj <pgid> |
| 列出指定PG中不连续的快照 | rados list-inconsistent-snapset <pgid> |

#### 4.1.3.2 PG相关的配置参数

在使用过程中，可能引起集群告警或操作异常的参数有：

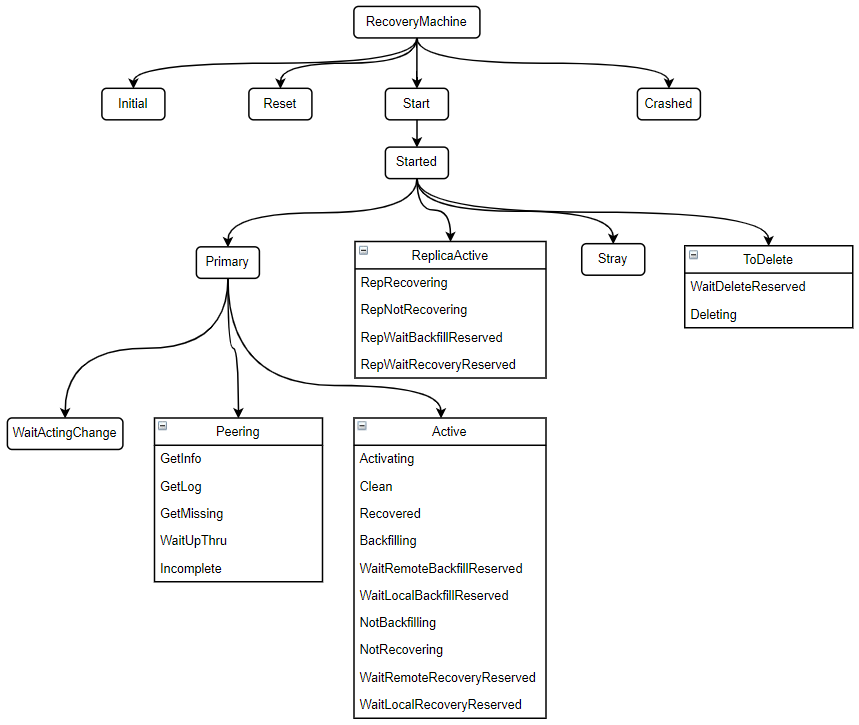
|  |  |
| --- | --- |
| 参数 | 含义 |
| mon\_max\_pool\_pg\_num | 存储池PG数量上限，默认为65536，当创建存储池时指定的pg\_num大于此参数，会导致创建失败。 |
| mon pg warn min per osd | 当集群中OSD的平均PG数小于此参数时，集群报TOO\_FEW\_PGS，在N版Ceph里，此值为0，默认关闭此告警，早期版本会因为此参数引起集群告警。  PG数设置过少，可能导致数据分布不均衡比较严重。 |
| mon pg warn max object skew | 当集群中出现MANY\_OBJECTS\_PER\_PG告警时，通过调整该参数可以消除，该告警说明某个或某几个存储池PG中存储的object数大于所有平均数的mon\_pg\_warn\_max\_object\_skew倍，即集群中存储池数据分布不均衡比较严重。默认为10。 |

## 4.2 PG状态机

### 4.1.1 PG状态及转化过程

PG随着存储池的创建而产生，也随着存储池的删除而消亡，在其可长可短的一生当中，会经历数量增长（扩容时），也会经历数量减少（N版后支持），同时也会由于经受各种考验而呈现出不同的状态；对于运维人员而言，最喜欢也最放心的状态是active+clean，也是Ceph运维的目标。

PG的状态管理由RecoveryMachine成员管理，其状态转换如下图所示：



引起PG状态变化的主要场景有：

1. 新建存储池，PG处于创建过程中；
2. 新增或者删除OSD，进行数据迁移；
3. OSD出现故障，进行数据恢复；
4. 修改CRUSH map，进行数据迁移；
5. PG各副本之间出现数据不一致；
6. PG副本在进行数据校验；
7. 存储系统无足够空间进行数据迁移或者恢复；
8. OSD重启等。

### 4.1.2 各状态含义及触发条件

在osd/osd\_types.h中对PG的状态进行定义，如下图所示：



|  |  |
| --- | --- |
| **PG状态** | **代表的含义/产生的条件** |
| **creating** | 表示PG 正在被创建, 通常以下两种情况下出现： 1. 当存储池正在被创建； 2. 增加一个存储池的 PG 数量时。 |
| **down** | 表示PG 处于失效状态, 一般出现在PG所在的OSD异常时。 |
| **repair** | 表示PG 正在被检查, 被发现的任何不一致都将尽可能被修复。 |
| **peering** | peering 由主 osd 发起，使存放 PG 副本的所有 OSD 对该PG的所有对象和元素数据的状态达成一致的过程。  peering过程完成后, 主OSD就可以接受客户端写请求 |
| **active** | 1. PG完成peering后会变成 active状态；  2. 该状态意味着PG中的数据可以被读写, 对该PG的操作请求会被处理。 |
| **clean** | 该状态表示PG中的所有数据副本处于同步状态，且处于正确的副本数。 |
| **replay** | 当某OSD崩溃后，PG正在等待客户端重新发起操作。 |
| **degraded** | 1. PG处于该状态表明PG数据实际副本数低于目标副本数，通常出现在节点或磁盘故障时； 2. 通常会与其它状态一并出现，如“active+degraded+recovering”、“active+degrade+backfilling”等。 |
| **inconsistent** | 1. PG副本之间出现数据元数据或者内容不一致，通常出现在PG scrub之后； 2. 常见的是由磁盘坏道引起。 |
| **recoverying** | 1. 该过程表明PG正在做数据恢复，恢复到正确的副本数。 2. 通常在系统出现磁盘或者节点故障一段时间后，其恢复速度和开始时间均可以精确控制并在线修改。 3. 常用的控制参数包括：   osd\_recovery\_max\_active  osd\_recovery\_delay\_start  osd\_recovery\_op\_priority  osd\_recovery\_priority  osd\_recovery\_retry\_interval  osd\_recovery\_sleep  osd\_repair\_during\_recovery  osd\_scrub\_during\_recovery |
| **backfilling** | 该状态表明数据正在做全量同步，或者迁移，通常发生在两种场景：  1. PG副本间数据相差较大，recovery不了，直接全量同步；  2.系统扩缩容时的数据重新均衡。  在 backfill 操作期间，可以看到多种状态： 1. backfill\_wait 表示条件不具备，操作还未开始，如手动停止、目标磁盘容量不够等； 2. backfilling 表示操作正在执行； 3. backfill\_too\_full 表示由于目标磁盘容量不足，操作停止；  常用的相关参数有：  mon\_osd\_backfillfull\_ratio  osd\_backfill\_retry\_interval  osd\_max\_backfills  osd\_backfill\_scan\_min  osd\_backfill\_scan\_max osd\_max\_backfills |
| **remapped** | 该状态表明PG将更新与OSD的映射关系，此时使用ceph pg map命令查询时，会发现acting set和up set不一致。  通常出现在集群Map发生变化时，如集群扩缩容、CRUSH规则修改、OSD权值修改等。  在remapped过程中，当主OSD发生变化时，旧的主OSD会一直服务到数据迁移完成，即PG恢复正常。 |
| **stale** | 处于该状态时，意味着无法获得主OSD，有如下可能性：   1. 集群才启动，PG peering还未完成； 2. 由于网络异常，导致主OSD无法向monitor及时同步状态信息； 3. 短时间内，PG所对应的OSD全部故障离线。 |
| **scrubbing** | PG 在做数据一致性校验，包括scrub和deep-scrub，前者只进行基本的元数据信息校验，后者包括数据完整性校验。  是防止数据出现静默错误的有效手段。  常用的控制参数包括：  mon\_scrub\_interval  osd\_deep\_scrub\_interval  osd\_max\_scrubs  osd\_scrub\_begin\_hour  osd\_scrub\_end\_hour  osd\_scrub\_chunk\_min  osd\_scrub\_chunk\_max  osd\_scrub\_priority  osd\_scrub\_sleep  osd\_scrub\_load\_threshold  手动操作命令有：  ceph pg {scrub|deep-scrub} <pg\_id>  ceph osd {set|unset} {noscrub|nodeep-scrub} |
| **inactive** | PG 很长时间没有显示为 acitve 状态, 处于此状态的PG 不可以执行读写, 要等待 OSD 更新数据到最新的备份状态，通常都是由于OSD离线引起 |
| **unclean** | PG很长时间都不是clean状态 (如无法完成恢复操作), PG包含对象没有完成相应的复制副本数量, 通常都要执行恢复操作 |
| **incomplete** | Peering过程中由于无法选出权威日志或者通过choose\_acting选出的acting不足以完成数据恢复，（例如针对纠删码，存活的副本数小于k值）等，导致Peering无法正常完成。即PG元数据丢失，无法恢复PG状态。 |

## 4.2 常见的PG问题及解决

参考：

* [Troubleshooting PGs — Ceph Documentation](https://docs.ceph.com/en/nautilus/rados/troubleshooting/troubleshooting-pg/)
* 《Red\_Hat\_Ceph\_Storage-3-Troubleshooting\_Guide-en-US》

在Ceph运维过程中，通过ceph -s或者ceph health detail查询可以看到PG相关状态信息，按照ceph health显示的状态划分有如下几种常见情况：

|  |  |
| --- | --- |
| **HEALTH\_ERR** | **HEALTH\_WARN** |
| pgs down | pgs stale |
| pgs inconsistent | unfound |
| scrub errors | incomplete |
|  |  |

### 4.2.2 pg处于stale状态

#### 故障现象

执行ceph -s可以看到如下信息：

|  |
| --- |
| [root@node0 ~]# ceph -s  cluster:  id: 8bd1a02d-5c91-464c-86ee-7ff61405ce77  health: HEALTH\_WARN  3 osds down  Reduced data availability: 4 pgs inactive, 4 pgs peering, 4 **pgs stale**  …… |

#### 原因分析

PG被标记为stale，说明无法找到该PG的主OSD，或者monitor一直无法收到主PG更新PG状态信息。

出现这种状态，通常有如下可能性：

1. 在集群启动后，PG peering一直无法完成；
2. 集群运行过程中，PG对应故障域内的OSD在短时间（故障恢复未完成前）内全部故障。

#### 处理过程

1. 首先确定处于stale状态的PG，以及它所对应的OSD；

|  |
| --- |
| [root@node0 ~]# ceph health detail | grep stale  HEALTH\_WARN Reduced data availability: 4 pgs inactive, 4 pgs peering, 4 pgs stale; Degraded data redundancy: 16602/70167 objects degraded (23.661%), 34 pgs degraded, 34 pgs undersized  PG\_AVAILABILITY Reduced data availability: 4 pgs inactive, 4 pgs peering, 4 pgs stale  pg 1.0 is stuck stale for 808.238834, current state stale+peering, last acting [7]  pg 1.9 is stuck stale for 808.238820, current state stale+peering, last acting [7]  pg 1.20 is stuck stale for 808.238792, current state stale+peering, last acting [7]  pg 1.2a is stuck stale for 808.238764, current state stale+peering, last acting [7] |

从上述截图可以得到两点信息（标红）：

* 处于stale状态的PG共有4个；
* PG异常前，最后一个正常的OSD；

1. 修复相关OSD。

根据确定的OSD，进行相关修复。

#### 故障构建

1. 使用ceph pg map确定某个PG所对应的所有OSD，关闭或者kill掉所有对应的OSD；
2. 集群启动时，将某PG对应的所有OSD均不启动。

### 4.2.3 pg处于down状态

#### 故障现象

执行ceph -s可以看到如下信息：

|  |
| --- |
| [root@node0 ~]# ceph -s  cluster:  id: 8bd1a02d-5c91-464c-86ee-7ff61405ce77  health: HEALTH\_WARN  Reduced data availability: 3 pgs inactive, 4 pgs down  …… |

#### 原因分析

出现该现象意味着PG无法完成peering，通常都是由于OSD异常引起，如反复重启OSD等。

#### 处理过程

1. 获取异常PG。

|  |
| --- |
| [root@node0 ~]# ceph health detail | grep down  HEALTH\_WARN Reduced data availability: 4 pgs inactive, 4 pgs down  PG\_AVAILABILITY Reduced data availability: 4 pgs inactive, 4 pgs down  pg 1.0 is down, acting [8,4,2]  pg 1.9 is down, acting [2,5,6]  pg 1.20 is down, acting [2,4,6]  pg 1.2a is down, acting [2,5,6] |

1. 使用如下命令，确认PG无法peering的原因。

|  |
| --- |
| [root@node0 ~]# ceph pg 1.0 query  {  "state": "down",  "snap\_trimq": "[]",  "snap\_trimq\_len": 0,  "epoch": 591,  ……  "probing\_osds": [  "2",  "4",  "8"  ],  "blocked": "peering is blocked due to down osds",  "down\_osds\_we\_would\_probe": [  3,  7  ],  "peering\_blocked\_by": [  {  "osd": 7,  "current\_lost\_at": 0,  "comment": "starting or marking this osd lost may let us proceed"  }  ]  },  …… |

从上可以发现是由于OSD.3和OSD.7 Down导致peering被阻塞。

1. 修复OSD.3和OSD.7。

#### 故障构建

1. 关闭PG对应的OSD组中的某一个OSD，在PG未恢复成active+clean前写入数据，再关闭剩余OSD，并启动之前关闭的OSD。

### 4.2.4 pg处于inconsistent状态

#### 故障现象

执行ceph health detail可以看到如下信息：

|  |
| --- |
| $ ceph health detail  HEALTH\_ERR 1 pgs inconsistent; 2 scrub errors  pg 0.6 is active+clean+inconsistent, acting [0,1,2]  2 scrub errors |

#### 原因分析

有三种情况会出现数据不连续：

1. 硬盘坏道，导致PG数据校验（scrub、deep-scrub）时校验失败；
2. 数据写入或者修改时不同步等导致副本之间校验不一致；
3. 节点时间不同步，导致周期性出现active+clean+inconsistent。

#### 处理过程

1. 找到处于inconsistent的异常PG；

|  |
| --- |
| $ ceph health detail  HEALTH\_ERR 1 pgs inconsistent; 2 scrub errors  pg 0.6 is active+clean+inconsistent, acting [0,1,2]  2 scrub errors |

在本例中，PG 0.6异常。

1. 确定出现异常原因；方法有二：
2. 手动启动scrub，根据异常日志判断；

|  |
| --- |
| $ ceph pg deep-scrub 0.6  $ grep scrub /var/log/ceph/ceph-mgr.node0.log |

1. 直接列出异常对象（object），根据相关提示判断。

|  |
| --- |
| $ rados list-inconsistent-obj 0.6 --format=json-pretty |

1. 进行PG修复。

当由于磁盘原因导致数据不一致时，可以使用repair命令行自动修复，除此之外则需要手动判断主副本再进行修复。以下分情况说明：

1. 当出现如下错误信息或者日志时，可以尝试使用ceph pg repair <pgid>自动修复；

|  |
| --- |
| missing attr  digest 0 != known digest  size 0 != known size  deep-scrub stat mismatch  candidate had a read error |

1. 当出现如下错误时，需要手动修复。

|  |
| --- |
| digest <digest> != known digest <digest> // 非0  omap\_digest <digest> != known omap\_digest <digest> |

即有确切的非0值时，可以判断非硬件原因引起的不一致，需人工来判断。

人工判断出正确数据版本后，可以使用**ceph-objectstore-tool**工具手动进行副本替换。

### 4.2.5 pg处于inactive状态

#### 故障现象

使用ceph health查询集群状态时，出现如下统计信息：

|  |
| --- |
| [root@node0 ~]# ceph health  HEALTH\_WARN Reduced data availability: **64 pgs inactive**; Degraded data redundancy: 23389/70167 objects degraded (33.333%), 64 pgs degraded, 64 pgs undersized |

当PG处于inactive状态，意味着：

1. 它将不再响应客户端的I/O请求；

#### 原因分析

通常都是由于OSD down引起。

#### 处理过程

1. 定位异常OSD；

|  |
| --- |
| [root@node0 ~]# ceph osd tree | grep down  3 hdd 0.09769 osd.3 down 0 1.00000  4 hdd 0.09769 osd.4 down 0 1.00000  5 hdd 0.09769 osd.5 down 0 1.00000 |

1. 修复OSD。

#### 故障构建

1. 关闭OSD，让某些PG对应的OSD数量低于min\_size数。

### 4.2.6 unfound object

#### 故障现象

通过ceph health命令查询，可以获取相关故障告警信息如下：

|  |
| --- |
| ceph health detail  HEALTH\_WARN 1 pgs degraded; 78/3778 **unfound** (2.065%)  pg 2.4 is active+degraded, 78 unfound |

出现此告警意味着出现数据块丢失，即该数据块的元数据存在，但集群中找不到对应副本；

#### 原因分析

通常是由于OSD的频繁上下线引起，如在集群做数据恢复过程中，存在完整数据副本的OSD掉线。

#### 处理过程

1. 确定异常PG；

|  |
| --- |
| $ ceph health detail | grep unfound |

1. 如果unfound的数量很大，则可判断大概率是由于OSD离线引起，使用如下命令定位离线OSD，并进行OSD修复。

|  |
| --- |
| $ ceph pg 2.4 query |

在recovery\_state下可以看到如下提示：

|  |
| --- |
| "recovery\_state": [  {  "name": "Started/Primary/Active",  "enter\_time": "2021-08-28 10:36:48.758889",  "might\_have\_unfound": [  {  "osd": "1",  "status": "already probed"  },  {  "osd": "3",  "status": "osd is down"  },  {  "osd": "6",  "status": "osd is down"  }  ], |

根据提示修复对应的OSD。

1. 如果unfound数量较小，可进行单独的对象定位，如果能人工判断正确副本，可用正确副本进行修复。

|  |
| --- |
| $ ceph pg 2.4 list\_unfound | head -n 20  {  "num\_missing": 817,  "num\_unfound": 817,  "objects": [  {  "oid": {  "oid": "benchmark\_data\_node0\_2866182\_object42955",  "key": "",  "snapid": -2,  "hash": 2847686659,  "max": 0,  "pool": 1,  "namespace": ""  },  "need": "711'1502",  "have": "0'0",  "flags": "none",  "locations": []  },  { |

获取丢失对象的id，配合ceph-objectstore-tool工具进行数据块，若实在无法找到，可使用如下命令进行自动处理（存在数据丢失风险）：

|  |
| --- |
| $ ceph pg 2.4 mark\_unfound\_lost <revert|delete> |

#### 故障构建

1. 设置集群norecover，关闭PG中的某个OSD，向对应的存储池写入数据，恢复关闭的OSD，待该PG peering完成处于active状态后，关闭PG中剩余的OSD。
2. 在磁盘上手动删除某个对象的所有副本，手动触发scrub。

# 第五章 Ceph 部署安装

## ceph编译打包

## 部署工具

高于nautilus用cephadm(扩展内容)

低于(含)nautilus用ceph-deploy

容器化部署 rook

ceph-ansible部署ceph

手动部署

## 预装环境准备

网络需求

硬盘及内存

集群容量估算

集群内外网分离

## 集群部署

单机部署

多节点部署

集群清理

## 推荐部署方案

大规模集群稳定性实践

# 第六章 Ceph集群维护

## Ceph常用管理命令

## Ceph Dashboard

## 扩容及缩容

## 数据均衡

## 常见故障及处理方法

硬盘故障处理

硬件故障判断及排除

1) 服务器相关：

a. 存储介质：SSD、HDD

b. 网卡/网口/交换机：参考网络相关

c. RAID/HBA卡

d. CPU、内存：

e. 整机故障

2) 网络相关：

a. 端口故障：bond、交换机端口

b. 时延类：闪断、丢包、错包、拥塞等，不同网络平面

c. 防火墙：部分端口不通、单通

网络分区

## ceph多地容灾介绍

## ceph集群性能分析

性能预估

基准性能测试

# 第七章 Ceph块存储实践

## 块存储介绍

## Ceph RBD介绍及操作

librbd

kernel rbd

nbd

## iscsi

## rbd客户端io流程

## Ceph使用场景

# 第八章 Ceph对象存储实践

## 对象存储介绍

## Ceph RGW介绍及操作

## Ceph RGW 高可用和负载均衡

## Ceph RGW 使用场景

## rgw客户端io流程

## 对象存储性能测试工具

# 第九章 Ceph文件存储实践

## 文件存储介绍

## Ceph Fs介绍及操作

## mds原理

## Ceph FS生产实践

案例：cephfs在大规模ai场景的落地实践

# 第十章 Crush原理及设计

## Crush算法介绍

## Crush数据结构和算法原理

## Crush实践以及生产环境案例分析

梳理不同案例的crush配置

# 第十一章 Ceph运维实战

## 各种典型案例的解决思路

## Ceph集群恢复思路及原理分析