# 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

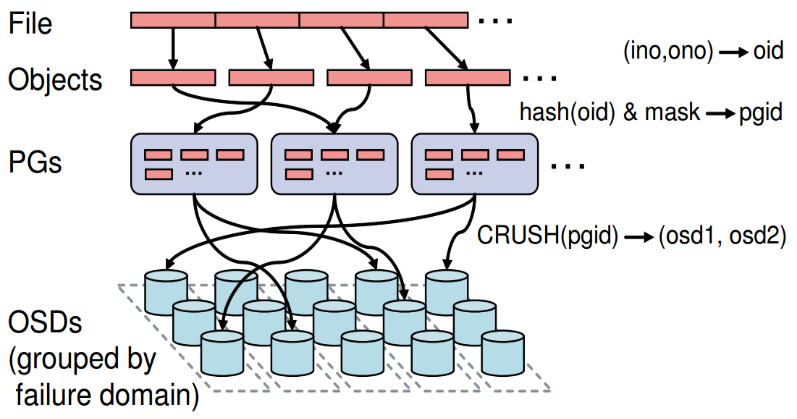
# 第四章 Ceph 状态迁移

## 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。

### 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：

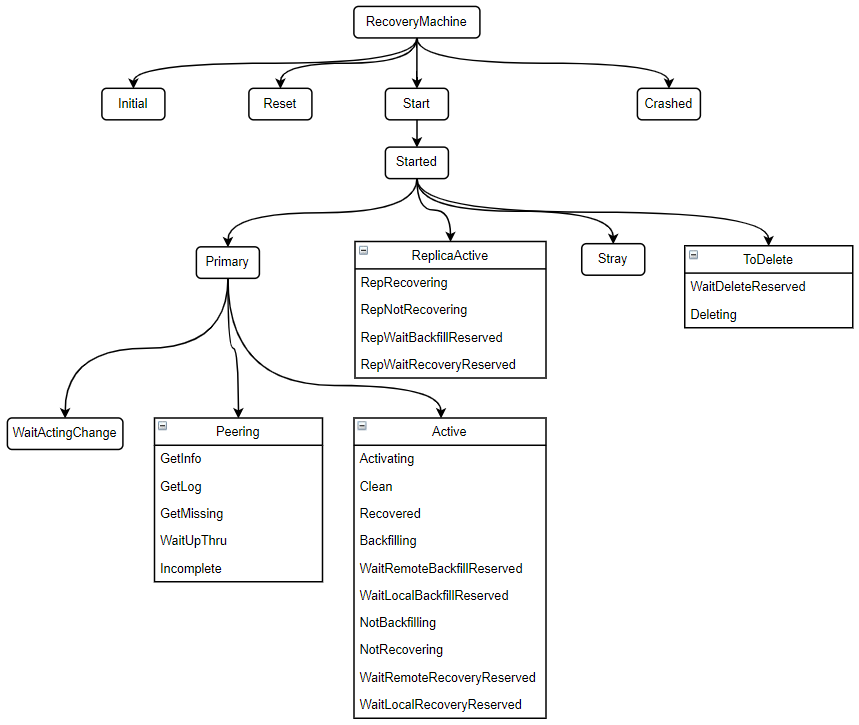
|  |  |
| --- | --- |
| 参数 | 含义 |
| mon\_max\_pool\_pg\_num | 存储池PG数量上限，默认为65536，当创建存储池时指定的pg\_num大于此参数，会导致创建失败。 |
| mon pg warn min per osd | 当集群中OSD的平均PG数小于此参数时，集群报TOO\_FEW\_PGS，在N版Ceph里，此值为0，默认关闭此告警，早期版本会因为此参数引起集群告警。  PG数设置过少，可能导致数据分布不均衡比较严重。 |
| mon pg warn max object skew | 当集群中出现MANY\_OBJECTS\_PER\_PG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** | 1. 当 ceph peering pg, ceph 将会把 PG 副本协定导入 osd, 当 ceph 完成 peering, 意味着 osd 同意当前 PG 状态, 并允许写入。 2. PG 处于 peering 过程中, peering 由主 osd 发起的使存放 PG 副本的所有 OSD 就 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. 当CEPH完成 peering 过程, pg 将会变成 active, active 状态意味着 pg 中的数据变得可用, 主 pg 将可执行读写操作 2. PG 是活动的, 意味着 PG 中的数据可以被读写, 对该 PG 的操作请求都将会被处理. |
| **clean** | 当 pg 显示 clean 状态, 主 osd 与副本 osd 成功同步并且没有异步复制, ceph 在 pg 中所有对象具有正确的副本数量。 |
| **replay** | 某 OSD 崩溃后, PG 正在等待客户端重新发起操作 |
| **degraded** | 1 当客户端写对象到主 osd, 主 OSD 会把数据写复制到对应复制 OSD, 在主 OSD 把对象写入存储后, PG 会显示为 degraded 状态, 直到主 osd 从复制 OSD 中接收到创建副本对象完成信息 2 PG 处于 active+degraded 原因是因为 OSD 是处于活跃, 但并没有完成所有的对象副本写入, 假如 OSD DOWN, CEPH 标记每个 PG 分配到这个相关 OSD 的状态为 degraded, 当 OSD 重新上线, OSD 将会重新恢复,  3 假如 OSD DOWN 并且 degraded 状态持续, CEPH 会标记 DOWN OSD, 并会对集群迁移相关 OSD 的数据, 对应时间由 mon osd down out interval 参数决定 4 PG 可以被标记为 degraded, 因为 ceph 在对应 PG 中无法找到一个或者多个相关的对象, 你不可以读写 unfound 对象, 你仍然可以访问标记为 degraded PG 的其他数据 5 PG 中部分对象的副本数量未达到规定的数量 |
| **inconsistent** | PG副本出现不一致, 对象大小不正确或者恢复结束后某个副本出现对象丢失现象 |
| **recoverying** | ceph 设备故障容忍在一定范围的软件与硬件问题, 当 OSD 变 DOWN, 那么包含该 OSD 的 PG 副本都会有问题, 当 OSD 恢复, OSD 对应的 PG 将会更新并反映出当前状态, 在一段时间周期后, OSD 将会恢复 recoverying 状态  recovery 并非永远都有效, 因为硬件故障可能会导致多个 OSD 故障, 例如, 网络交换机故障, 可以导致集群中的多个主机及主机包含的 OSD 故障，当网络恢复之后, 每个 OSD 都必须执行恢复  CEPH 提供一定数量的设定在新服务请求与恢复 PG 中数据对象时的资源平衡,  osd recovery delay start 设定允许 osd 重启, re-peer 并在启动 恢复之前处理一些回应请求,  osd recovery threads 设定了恢复过程中线程限制 (默认 1 )  osd recovery thread timeout 设定线程超时, 因为可能出现多个 osd 故障, 重启后在 re-peer 过程中可能出现污染 osd recovery max active 设定限制对一个 osd 从故障后, 恢复请求并发数量 osd recovery max chunk 限制恢复时的数据 chunk 大小, 预防网络堵塞  PG 正在迁移或者同步对象及其副本, 一个 OSD 停止服务(DOWN), 其内容将会落后与 PG 内的其他副本, 这时 PG 将会进入 recoverying 状态, 该 OSD 上的对象将从其他副本同步过来 |
| **backfilling** | 当新 OSD 加入集群, CRUSH 将会为集群新添加的 OSD 重新分配 PG, 强制新的 OSD 接受重新分配的 PG 并把一定数量的负载转移到新 OSD 中back filling OSD 会在后台处理, 当 backfilling 完成, 新的 OSD 完成后, 将开始对请求进行服务  在 backfill 操作期间, 你可以看到多种状态,  backfill\_wait 表示 backfill 操作挂起, 但 backfill 操作还没有开始 ( PG 正在等待开始回填操作 ) backfill 表示 backfill 操作正在执行 backfill\_too\_full 表示在请求 backfill 操作, 由于存储能力问题, 但不可以完成,   ceph 提供设定管理装载重新分配 PG 关联到新的 OSD osd\_max\_backfills 设定最大数量并发 backfills 到一个 OSD, 默认 10 osd backfill full ratio 当 osd 达到负载, 允许 OSD 拒绝 backfill 请求, 默认 85%,  假如 OSD 拒绝 backfill 请求, osd backfill retry interval 将会生效, 默认 10 秒后重试 osd backfill scan min , osd backfill scan max 管理检测时间间隔  一个新 OSD 加入集群后, CRUSH 会把集群先有的一部分 PG 分配给他, 该过程称为回填, 回填进程完成后, 新 OSD 准备好了就可以对外提供服务 |
| **remapped** | 当 pg 改变, 数据从旧的 osd 迁移到新的 osd, 新的主 osd 应该请求将会花费一段时间, 在这段时间内, 将会继续向旧主 osd 请求服务, 直到 PG 迁移完成, 当数据迁移完成, mapping 将会使用新的 OSD 响应主 OSD 服务  当 PG 的 action set 变化后, 数据将会从旧 acting set 迁移到新 action set, 新主 OSD 需要过一段时间后才能提供服务, 因此它会让老的主 OSD 继续提供服务, 直到PG 迁移完成, 数据迁移完成后, PG map 将会使用新 acting set 中的主 OSD |
| **stale** | 当 ceph 使用 heartbeat 确认主机与进程是否运行, ceph osd daemon 可能由于网络临时故障, 获得一个卡住状态 (stuck state) 没有得到心跳回应。 默认osd daemon 会每 0.5 秒报告 PG, up 状态, 启动与故障分析,  假如 PG 中主 OSD 因为故障没有回应 monitor 或者其他 OSD 报告 主 osd down, 那么 monitor 将会标记 PG stale,  当你重启集群, 通常会看到 stale 状态, 直到 peering 处理完成,  在集群运行一段时候, 看到 stale 状态, 表示主 osd PG DOWN 或者主 osd 没有报告PG信息到 monitor 中 PG 处于未知状态, monitors 在PG map 改变后还没有收到过 PG 的更新, 启用一个集群后, 常常会看到主 peering 过程结束前 PG 处于该状态 |
| **scrubbing** | PG 在做数据一致性校验 |
| **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 (不可以完成之前恢复的操作), 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 |
| **stale** | PG 状态很长时间没有被 ceph-osd 更新过, 表示存储在该 GP 中的节点显示为 DOWN, PG 处于unknown 状态, 因为 OSD 没有报告 monitor 由 mon osd report timeout 定义超时时间。 |
| **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