**ceph数据修复方法（初稿）**

# 常见数据恢复问题

局点常见的数据恢复问题包括（但不局限于）以下几种：

1. **PG incomplete**
2. **PG down**
3. **Objects unfound**
4. **PG inconsistent**
5. 由于任何数据都可能丢失，问题现象往往难以预测，例如各种断言，本文暂不讨论断言问题的修复手段。

………

# 常见数据恢复问题的原因

以上1~3三种故障类型对于ceph集群来说都属于广义的数据丢失，主要原因无外乎以下两种：

## 数据真正丢失，盘上都找不到

常见原因有：raid卡缓存打开（默认打开）情况下插拔盘、硬盘缓存没关情况下异常掉电、raid卡电源故障情况下重启主机等。

**此种情况下，丢失的数据无法恢复，集群数据只能恢复到故障前的状态。**

## 数据在盘上，但是无法顺利获取

常见的场景有：OSD因为断言进程起不来、xfs文件系统损坏导致OSD mount失败、硬盘故障OSD无法启动等。

**此种情况下，理论上数据可以完全恢复，有必要时可以通过数据修复公司将问题盘上的数据都导出来以修复集群中的故障PG。**

而对于4.PG inconsistent，意思是主从间数据不一致，导致的原因有：比如主上数据是完整的，但是从上数据由于数据丢失是不完整的；或者从上硬盘发生数据静默错误等等。

# 故障修复方法

如果ceph –s看到同时拥有incomplete、down、inconsistent、objects unfound状态，通常先修复incomplete或down的PG，再修复objects unfound，inconsistent的PG可放在最后修复。

接下来介绍针对本文开头的四种故障类型的修复方法

## PG incomplete

Incomplete往往是由于pginfo或者pglog这两种元数据损坏导致的。这种元数据损坏往往属于第二节《常见数据恢复问题的原因》中的第一点“数据真正丢失，盘上都找不到”，此时集群数据只能恢复到出故障前的状态，故障过程中丢失的数据无法找回。

代码逻辑中导致PG incomplete有两种场景：

1. 权威选取失败；
2. 选取acting set失败（如纠删码k+m，结果选出的acting\_size < k）

此处拓展补充下，peering过程是ceph数据一致性的重要过程，**peering过程的关键就两点：一是选取权威PG，使得其余副本或分片完成数据同步从而实现数据一致性；二是选取合适的acting set来承担业务io**，众所周知recovery过程的恢复速度高于backfill的恢复速度，所以构成acting set的OSD都必须是非backfill的OSD，这样业务io如果正好是待修复的object，只需要通过recovery来快速修复而不需要通过backfill，尽可能减少数据修复对业务io的影响（**主要是针对读的性能优化，写io会发往actingbackfill中的所有OSD**）。

### 1.权威选取失败

**对于“场景（1）权威选取失败导致的PG incomplete”的修复方法：**

#### 方法一

*ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-17/ --journal-path /var/lib/ceph/osd/ceph-17/journal --type filestore --pgid 2.615 --op mark-complete*

如果提示命令无效，可能是当前版本不支持此命令，则需要同步ceph-objectstore-tool最新代码，出包替换后再用此命令修复。

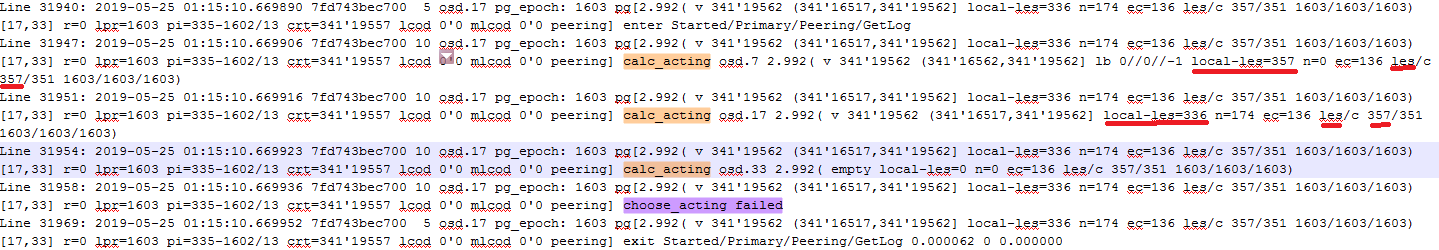
（社区tracker对应pr10098,请参考）

#### 方法二

（此方法无需替换代码）

1. 确认问题现象：

ceph health detail | grep incomplete查看incomplete的PGid，将该PG主OSD日志级别开到10后重启，从而打印incomplete的原因：



首先看到紫色部分的打印：“choose\_acting failed”有且仅有此三个字母的打印才是场景(1)权威选取失败导致的PG incomplete,否则是场景(2)

查看osd.7、osd.17、osd.33的calc\_acting的打印信息，（osd.7中由于有lb的打印，证明其未完成backfill，所以不能选其作为权威，osd.33的pginfo信息是empty所以也不能选osd.33作为权威。）由于osd.17的info.history.last\_epoch\_started （对应日志中的les）=357，大于info.last\_epoch\_started(对应日志中的local-les)的336，所以osd.17也不能被选作权威（原因见下面的拓展补充），故权威选取失败。

此处拓展补充下，info.history.last\_epoch\_started（对应日志中的les）值是在该PG所有副本都进入active状态时才会更新内存中的值，而info.last\_epoch\_started(对应日志中的local-les)是本地（主或从）完成active就会更新内存中的值，所以这两个值是在peering的不同阶段更新的，理论上info.history.last\_epoch\_started的值不可能大于info.last\_epoch\_started的值，但是现网很多局点会发生本例中的情况（即info.history.last\_epoch\_started>info.last\_epoch\_started）,究其原因就是因为info.last\_epoch\_starte数据丢失导致新的版本未能落盘，所以在选权威时如果出现info.history.last\_epoch\_started>info.last\_epoch\_starte这种逻辑异常，会直接导致选取权威失败。

至于为什么每次都是info.last\_epoch\_starte数据丢失而不是info.history.last\_epoch\_started数据丢失这涉及到peering过程中的pginfo元数据落盘过程，由于落盘过程的差异导致info.last\_epoch\_starte数据丢失概率远高于info.history.last\_epoch\_started数据丢失的概率（理论上任何数据都可能丢失，结果往往难以预测）。

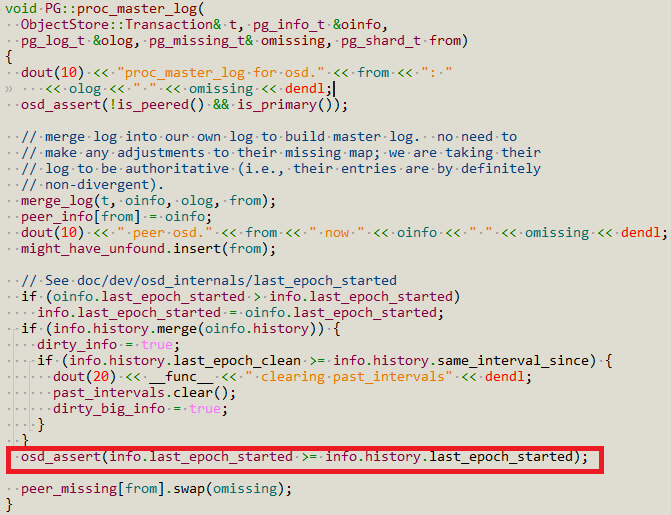
**只有日志信息满足上述描述（建议服务同事拍下上述日志信息经过开发确认后再进行下一步操作），才能按照以下方法修复**：

1. 在主osd所在节点的/etc/ceph/ceph.conf文件中的osd下添加：

osd\_find\_best\_info\_ignore\_history\_les=true

保存后重启主osd进程（确保此时业务已经停止，才可以重启OSD操作！）

1. 重启主OSD进程后，该PG会重新peering，此时能正常选出权威，但是如果该PG的peering过程需要合并日志（主与权威的日志不一致），则还会出现以下断言：



（融合版本社区已经修改此断言，当osd\_find\_best\_info\_ignore\_history\_les=true时不会有断言）

此时如果是**副本策略**，则需要将权威PG导出，导入到各个非权威PG上，从而避开合并日志过程。具体方法如下：

1. 首先选择权威副本，上例中显然选择OSD.17作为权威，在OSD.17上执行：

*ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-17/ --journal-path /var/lib/ceph/osd/ceph-17/journal --type filestore --pgid 2.615 --op export –file pg2\_615-17*

在其余非权威OSD上执行类似操作将PG导出备份，导出文件注意区分名字

再将OSD.17上导出的pg2\_615-17导入到其余非权威OSD上，在其余非权威OSD上先删除原有PG：

*ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-7/ --journal-path /var/lib/ceph/osd/ceph-7/journal --type filestore --pgid 2.615 --op remove*

然后将OSD.17上拷过来的PG导入到其余非权威OSD上：

*ceph-objectstore-tool --data-path /var/lib/ceph/osd/ceph-7/ --journal-path /var/lib/ceph/osd/ceph-7/journal --type filestore --op import –file pg2\_615-17*

（注意：执行ceph-objectstore-tool相关命令时需要先停相关OSD的进程）

等OSD.7和OSD.33导入成功后，启动OSD进程，此时该PG能恢复健康。

如果该PG是**纠删码**，则没有办法，需要修改代码修改此处断言，修改为和融合版本一致才能跳过此断言，或者按照方法一修复（建议采用方法一）。

### 2.acting set选取失败

对于“场景（2）选取acting set失败（如纠删码k+m，结果选出的acting\_size < k）”由于最近没有遇到相关问题，本文暂略，下次遇到再补充。

## （二）PG down

PG down是因为有该PG相关的OSD down，而该down的OSD上可能有部分io对象，如果这个down的OSD起不来这些对象将无法恢复。

**此时如果故障OSD始终无法拉起，如果直接用命令:**

***ceph osd lost osdid*将该OSD标lost使得Peering往下进行，此时会有丢失数据风险，属于高危操作！（用户和开发在未得到用户授权情况下切勿直接标lost！）**

（**其实对于PG down，如果不能将down的OSD拉起，也不找数据修复公司把down的OSD上的数据导出，那么想要恢复集群也只能将down的osd标记为lost这一种方法。**）

下面通过两副本集群场景举例来阐述一次peering过程的基本原理，并解释为何这是高危操作：



如上图所示的peering场景，在当前时刻（也就是图中的current interval）OSD.1要通过peering完成数据一致性过程，但是此时OSD.3由于xfs损坏根本起不来，那么OSD.1能否获得最全的数据呢？

如果在past interval2期间OSD.3**有数据写入**，显然这些数据在当前current interval期间是无法恢复的，此时OSD.3如果起不来，手动将OSD.3标为lost就会丢失past interval2期间写入的数据。

如果在past interval2期间OSD.3**无数据写入**，则在current interval期间OSD.1有最全的数据，可以完成peering过程。

那么在current interval期间，只有OSD.1在场的情况下，OSD.1如何知道past interval2期间（OSD.1当时并不在场）到底有没有数据写入到OSD.3呢？

这个时候就必须要借助mon来实现（可见peering过程不光涉及到osd和osd之间，还涉及到mon，任何环节出问题都可能会导致peering卡住等问题）。



如上图所示，**OSD.1可以根据OSDMap中保存的OSD.3的upthru版本号来判断在past interval 2期间，OSD.3是否*有可能*有数据写入**。注意此处用的词是“有可能”，因为在上图的步骤②中，哪怕mon已经知道了OSD.3在past interval 2期间活着，但是有可能PG在步骤①之后就停止了peering过程（比如此时OSD.3进程突然退出），亦或是PG进入active状态之后压根没有业务IO下来，这些对于mon都是无法感知的。

**所以OSD.1只能通过mon的osdmap中OSD.3的upthru(该值是一个版本号，表示OSD.3在该版本时还活着)来判断在past interval 2期间OSD.3可能有数据写入（判断的结果只有两种：可能有数据写入or无数据写入，如果判断出来无数据写入，current interval期间OSD.3 down也就不会导致PG down）**

所以如果遇到局点有PG down了，也就是说在某个past interval期间可能有数据写入到了当前起不来的OSD上，如果这时把该起不来的OSD标记lost，就等于放弃这个past interval期间写入的数据。

现网很多数据丢失的局点都会遇到PG down，如果开发在用户授权的情况下将起不来的OSD标记为lost，有时候PG状态会由down转变为incomplete，此时按照上文pg incomplete对应的修复方法修复即可。

## （三）objects unfound

“常见数据恢复问题的原因”中描述的两种原因都可能导致objects unfound。

此时需要先分析（分析过程比较复杂，此文暂略，**现网必须要研发介入分析，不可上来就进行mark\_unfound\_lost操作**）到底是因为“数据真正丢失，盘上都找不到”还是“数据在盘上，但是该盘对应OSD起不来”，如果是**前者**，就只能通过命令：

*ceph pg pgid mark\_unfound\_lost revert|delete*

强行回滚/删除这些unfound的对象来修复。

具体介绍可以参考《ceph设计原理与实现》p124表格第一行描述。

如果是**后者**，要尽可能的去拉起down的OSD以恢复unfound的对象。

## （四）PG inconsistent

Ceph提供repair命令修复不一致的PG，步骤如下：

*ceph health detail | grep inconsistent (找出当前不一致的PGid)*

*ceph osd set noscrub（先禁用scrub）*

*ceph osd set nodeep-scrub（先禁用deep-scrub）*

*ceph tell osd.\* injectargs –osd\_max\_scrubs xx (xx为当前不一致的PG数，此步骤是将当前repair的并发数调高，默认是1)*

*ceph pg repair pgid （修复不一致的PG）*

PG修复过程较慢，一个PG可能会持续半小时以上，需要耐心等待，等待修复完成 ceph –s看到pg 没有inconsistent状态之后，重启scrub和deep-scrub：

*ceph osd unset noscrub（先禁用scrub）*

*ceph osd unset nodeep-scrub（先禁用deep-scrub）*

*ceph tell osd.\* injectargs –osd\_max\_scrubs 1(重新设为1)*