ceph集群运维命令

# 文档说明

官方文档（英文）：<http://docs.ceph.com/docs/master/rados/operations/operating/>

中文社区文档（中文）：http://docs.ceph.org.cn/rados/operations/

ceph的中文文档有很多没有更新，因此本文档以结合官方英文文档与中文社区文档，主要做了：

1、翻译了中文社区文档中没有翻译的部分；

2、对比中英文文档，添加了英文文档中新增、改动的部分，并做了翻译与标注。

在本文档中，有些语句不通顺之处、不理解之处，可参考上述链接中英文文档或中文文档对应的部分，做进一步理解。

# 目录

[ceph集群运维命令 1](#_Toc503452403)

[文档说明 1](#_Toc503452404)

[目录 2](#_Toc503452405)

[序：内容导览 11](#_Toc503452406)

[高级运维 11](#_Toc503452407)

[数据归置 11](#_Toc503452408)

[低级运维 11](#_Toc503452409)

[故障排除 11](#_Toc503452410)

[1. 操纵集群（★英文文档改动） 12](#_Toc503452411)

[1.1 用systemd运行ceph 12](#_Toc503452412)

[1.1.1 在节点上列出ceph systemd单元 13](#_Toc503452413)

[1.1.2 启动Ceph节点上的所有守护进程（不管类型） 14](#_Toc503452414)

[1.1.3 停止Ceph节点上的所有守护进程（不管类型） 15](#_Toc503452415)

[1.1.4 按类型启动守护进程 15](#_Toc503452416)

[1.1.5 按类型停止守护进程 16](#_Toc503452417)

[1.1.6 启动单个进程 16](#_Toc503452418)

[1.1.7 停止单个进程 16](#_Toc503452419)

[1.2 用upstart运行ceph 17](#_Toc503452420)

[1.2.1 启动所有守护程序 17](#_Toc503452421)

[1.2.2 停止所有的守护程序 18](#_Toc503452422)

[1.2.3 按类型启动所有守护进程 18](#_Toc503452423)

[1.2.4 按类型停止所有守护进程 18](#_Toc503452424)

[1.2.5 启动守护进程 18](#_Toc503452425)

[1.2.6 停止守护程序 19](#_Toc503452426)

[1.2 运行ceph 19](#_Toc503452427)

[2. 健康检查（★英文文档新增） 21](#_Toc503452428)

[2.1 概述 21](#_Toc503452429)

[2.2 定义 21](#_Toc503452430)

[2.2.1 OSDS 21](#_Toc503452431)

[2.2.2 数据健康 (池 & PGs) 26](#_Toc503452432)

[3. 监控集群 32](#_Toc503452433)

[3.1 使用命令行 32](#_Toc503452434)

[3.1.1 交互模式 32](#_Toc503452435)

[3.1.2 非默认路径 33](#_Toc503452436)

[3.2 检查集群的状态（★英文文档新增） 33](#_Toc503452437)

[3.3 观察集群 34](#_Toc503452438)

[3.4 检测健康检查 35](#_Toc503452439)

[3.5 检测配置问题 36](#_Toc503452440)

[3.6 检查集群的使用情况统计 36](#_Toc503452441)

[3.7 检查OSD状态（★英文文档改动） 37](#_Toc503452442)

[3.8 检查监视器状态 39](#_Toc503452443)

[3.9 检查MDS状态 39](#_Toc503452444)

[3.10 检查归置组状态 40](#_Toc503452445)

[3.11 使用管理套接字（ADMIN SOCKET） 40](#_Toc503452446)

[4. 监控 OSD 和归置组 41](#_Toc503452447)

[4.1 监控OSD 41](#_Toc503452448)

[4.2 归置组集（PG SETS） 43](#_Toc503452449)

[4.3 节点互联 44](#_Toc503452450)

[4.4 监控归置组状态 44](#_Toc503452451)

[CREATEING 47](#_Toc503452452)

[PEERING 48](#_Toc503452453)

[ACTIVE 48](#_Toc503452454)

[CLEAN 48](#_Toc503452455)

[DEGRADED 48](#_Toc503452456)

[RECOVERING 49](#_Toc503452457)

[BACK FILLING 49](#_Toc503452458)

[REMAPPED 50](#_Toc503452459)

[STALE 50](#_Toc503452460)

[4.5 找出故障归置组 50](#_Toc503452461)

[4.6 定位对象 51](#_Toc503452462)

[5. 用户管理（▲中文文档未翻译） 52](#_Toc503452463)

[5.1 背景 53](#_Toc503452464)

[5.1.1 用户 53](#_Toc503452465)

[5.1.2 授权（能力） 54](#_Toc503452466)

[5.1.3 池存储 56](#_Toc503452467)

[5.1.4 应用标签（APPLICATION TAGS） 56](#_Toc503452468)

[5.1.5 空间命名（NAMESPACE） 56](#_Toc503452469)

[5.2 管理用户 57](#_Toc503452470)

[5.2.1 罗列用户 57](#_Toc503452471)

[5.2.2 获取用户 58](#_Toc503452472)

[5.2.3 添加用户 58](#_Toc503452473)

[5.2.4 修改用户能力 59](#_Toc503452474)

[5.2.5 删除用户 60](#_Toc503452475)

[5.2.6 打印用户密钥 60](#_Toc503452476)

[5.2.7 导入用户 61](#_Toc503452477)

[5.2密钥环管理 61](#_Toc503452478)

[5.2.1 创建密钥环 62](#_Toc503452479)

[5.2.2 把用户加入密钥环 62](#_Toc503452480)

[5.2.3 创建用户 63](#_Toc503452481)

[5.2.4 修改用户属性 63](#_Toc503452482)

[5.3命令行用法 64](#_Toc503452483)

[5.4 限制 64](#_Toc503452484)

[6. 修复PG不一致（▲英文文档只有标题，内容为空） 65](#_Toc503452485)

[7. 数据归置（DATA PLACEMENT）概览 65](#_Toc503452486)

[8. 存储池（★英文文档改动） 66](#_Toc503452487)

[8.1 列表池 66](#_Toc503452488)

[8.2 创建一个池 67](#_Toc503452489)

[8.3 关联池到应用程序（★英文文档新增） 68](#_Toc503452490)

[8.4 设置存储池配额 69](#_Toc503452491)

[8.5 删除存储池 69](#_Toc503452492)

[8.6 重命名存储池 70](#_Toc503452493)

[8.7 查看存储池统计信息 70](#_Toc503452494)

[8.8 拍下存储池快照 70](#_Toc503452495)

[8.9 删除存储池快照 71](#_Toc503452496)

[8.10 设置存储池选项值 71](#_Toc503452497)

[8.11 获取存储池选项值 75](#_Toc503452498)

[8.12 设置对象副本数 76](#_Toc503452499)

[8.13 获取对象副本数 77](#_Toc503452500)

[9. 纠删码（▲中文文档未翻译） 77](#_Toc503452501)

[9.1 创建示例纠删码存储池 77](#_Toc503452502)

[9.2 纠错编码配置文件 78](#_Toc503452503)

[9.3 重写纠删码（★英文文档新增） 79](#_Toc503452504)

[9.4 纠删码存储池与缓存分级 80](#_Toc503452505)

[9.5 术语 80](#_Toc503452506)

[9.6 内容列表 81](#_Toc503452507)

[9.6.1 纠删码配置 81](#_Toc503452508)

[9.6.2 JERASURE 纠删码插件 83](#_Toc503452509)

[9.6.3 ISA 纠删码插件 85](#_Toc503452510)

[9.6.4 局部自修复纠删码插件（★英文文档改动） 87](#_Toc503452511)

[10. 分级缓存 95](#_Toc503452512)

[10.1 一个忠告（★英文文档新增） 96](#_Toc503452513)

[10.1.1 已知良好的工作负载 96](#_Toc503452514)

[10.1.2 已知的不良工作负载 97](#_Toc503452515)

[10.2 配置存储池 97](#_Toc503452516)

[10.2.1 配置后端存储池 97](#_Toc503452517)

[10.2.2 配置缓存池 97](#_Toc503452518)

[10.3 创建缓存层 98](#_Toc503452519)

[10.4 配置缓存层 98](#_Toc503452520)

[10.4.1 目标尺寸和类型 99](#_Toc503452521)

[10.4.2 缓存空间消长 100](#_Toc503452522)

[10.4.3 缓存时长 102](#_Toc503452523)

[10.5 拆除缓存层 102](#_Toc503452524)

[10.5.1 拆除只读缓存 102](#_Toc503452525)

[10.5.2 拆除回写缓存 103](#_Toc503452526)

[11. 归置组（pg）（★英文文档改动） 104](#_Toc503452527)

[11.1 预定义 PG\_NUM 104](#_Toc503452528)

[11.2 PG是如何使用的？（▲中文文档未翻译） 104](#_Toc503452529)

[11.3 PG的权衡（▲中文文档未翻译） 105](#_Toc503452530)

[11.3.1 数据持久性 106](#_Toc503452531)

[11.3.2 对象在一个池中的分配 107](#_Toc503452532)

[11.3.3 内存，CPU和网络使用情况 107](#_Toc503452533)

[11.4 选择PG的数量（▲中文文档未翻译） 108](#_Toc503452534)

[11.1 设置归置组数量 109](#_Toc503452535)

[11.2 获取归置组数量 109](#_Toc503452536)

[11.3 获取归置组统计信息 109](#_Toc503452537)

[11.4 获取卡住的归置组统计信息 109](#_Toc503452538)

[11.5 获取一归置组运行图 110](#_Toc503452539)

[11.6 获取一 PG 的统计信息 110](#_Toc503452540)

[11.7 清理归置组 110](#_Toc503452541)

[11.8 优先考虑安置组的回填/恢复（★英文文档新增） 111](#_Toc503452542)

[11.9 恢复丢失的 111](#_Toc503452543)

[12. 使用PG-UPMAP（★英文文档新增） 112](#_Toc503452544)

[12.1 启用 112](#_Toc503452545)

[12.2 一个忠告 113](#_Toc503452546)

[12.3 离线优化 113](#_Toc503452547)

[13. CRUSH 图 114](#_Toc503452548)

[13.1 CRUSH 位置（★英文文档改动） 114](#_Toc503452549)

[定制位置挂钩 115](#_Toc503452550)

[13.2 CRUSH结构（★英文文档改动） 115](#_Toc503452551)

[13.2.1 设备 115](#_Toc503452552)

[13.2.2 类型和桶 115](#_Toc503452553)

[13.2.3 规则 117](#_Toc503452554)

[13.2.4 设备类 118](#_Toc503452555)

[13.2.5 权重集 119](#_Toc503452556)

[13.3 修改CRUSH MAP 120](#_Toc503452557)

[13.3.1 添加/移动OSD 120](#_Toc503452558)

[13.3.2 调整OSD的CRUSH权重 121](#_Toc503452559)

[13.3.3 删除OSD 121](#_Toc503452560)

[13.3.4 添加桶 121](#_Toc503452561)

[13.3.5 移动桶 122](#_Toc503452562)

[13.3.6 删除桶 122](#_Toc503452563)

[13.3.7 创建compat 权重集（★英文文档新增） 123](#_Toc503452564)

[13.3.8 创建per-pool 权重集（★英文文档新增） 123](#_Toc503452565)

[13.3.9 为副本池创建规则（★英文文档新增） 124](#_Toc503452566)

[13.3.10 为纠删码池创建规则（★英文文档新增） 125](#_Toc503452567)

[13.3.10 删除规则（★英文文档新增） 126](#_Toc503452568)

[13.4 可调参数 126](#_Toc503452569)

[13.4.1 可调选项非最优时发出警告 126](#_Toc503452570)

[13.4.2 一些要点 127](#_Toc503452571)

[13.4.3 调整CRUSH 127](#_Toc503452572)

[13.5 主亲和性 128](#_Toc503452573)

[14. 手动编辑CRUSH映射（★英文文档新增） 128](#_Toc503452574)

[14.1 获取CRUSH映射 129](#_Toc503452575)

[14.2 反编译CRUSH映射 129](#_Toc503452576)

[14.3 组成 129](#_Toc503452577)

[14.4 CRUSH映射设备 129](#_Toc503452578)

[14.5 CRUSH映射桶类型 130](#_Toc503452579)

[14.6 CRUSH地图桶层次 131](#_Toc503452580)

[14.7 CRUSH映射规则 134](#_Toc503452581)

[15. 在不同的OSDS上放置不同的池（★英文文档新增） 136](#_Toc503452582)

[15.1 调整CRUSH，困难的方式 139](#_Toc503452583)

[15.2 传统值（LEGACY VALUES） 140](#_Toc503452584)

[16. 增加/删除 OSD 140](#_Toc503452585)

[16.1 增加OSD 140](#_Toc503452586)

[16.1.1 部署硬件 140](#_Toc503452587)

[16.1.2 安装必要软件 141](#_Toc503452588)

[16.1.3 增加 OSD （手动） 141](#_Toc503452589)

[16.1.4 替换一个OSD（★英文文档新增） 143](#_Toc503452590)

[16.1.5 启动 OSD 144](#_Toc503452591)

[16.1.6 观察数据迁移 144](#_Toc503452592)

[16.2 删除OSD（手动） 145](#_Toc503452593)

[16.2.1 把 OSD 踢出集群 145](#_Toc503452594)

[16.2.2 观察数据迁移 145](#_Toc503452595)

[16.2.3 停止 OSD 146](#_Toc503452596)

[16.2.4 删除 OSD（★英文文档改动） 146](#_Toc503452597)

[17. 增加/删除监视器 147](#_Toc503452598)

[17.1 增加监视器 147](#_Toc503452599)

[17.1.1 部署硬件 148](#_Toc503452600)

[17.1.2 安装必要软件 148](#_Toc503452601)

[17.1.3 增加监视器（手动） 148](#_Toc503452602)

[17.2 删除监视器 150](#_Toc503452603)

[17.2.1 删除监视器（手动） 150](#_Toc503452604)

[17.2.2 从不健康集群删除监视器 150](#_Toc503452605)

[17.3 更改监视器的 IP 地址 151](#_Toc503452606)

[17.3.1 一致性要求 152](#_Toc503452607)

[17.3.2 更改监视器 IP 地址（正确方法） 152](#_Toc503452608)

[17.3.3 更改监视器 IP 地址（凌乱方法） 153](#_Toc503452609)

[18. BLUESTORE迁移（★英文文档新增） 155](#_Toc503452610)

[18.1 部署带BLUESTORE新OSD 156](#_Toc503452611)

[18.2 转换现有的OSD 156](#_Toc503452612)

[18.2.1 标记并替换 156](#_Toc503452613)

[18.2.2 整个主机替换 158](#_Toc503452614)

[18.2.3 每个OSD设备拷贝 162](#_Toc503452615)

[19. 命令参考 162](#_Toc503452616)

[19.1 监视器命令 162](#_Toc503452617)

[19.2 系统命令 163](#_Toc503452618)

[19.2.1显示集群状态 163](#_Toc503452619)

[19.2.2显示集群状态的运行摘要、及主要事件 163](#_Toc503452620)

[19.2.3显示监视器法定人数状态，包括哪些监视器参与者、哪个是首领。 163](#_Toc503452621)

[19.2.4 查询单个监视器状态，包括是否在法定人数里 163](#_Toc503452622)

[19.3 认证子系统 163](#_Toc503452623)

[19.3.1 要添加一个 OSD 的密钥环 163](#_Toc503452624)

[19.3.2 要列出集群的密钥及其能力 164](#_Toc503452625)

[19.4 归置组（pg）子系统 164](#_Toc503452626)

[19.4.1 显示所有归置组的统计信息 164](#_Toc503452627)

[19.4.2 显示卡在某状态的所有归置组 164](#_Toc503452628)

[19.4.3 删除“丢失”对象，或者恢复到其先前状态 165](#_Toc503452629)

[19.5 OSD子系统 165](#_Toc503452630)

[19.5.1 查询 OSD 子系统状态 165](#_Toc503452631)

[19.5.2 把最新的 OSD 运行图拷贝到一个文件 165](#_Toc503452632)

[19.5.3 从最新 OSD 运行图拷出 CRUSH 图 165](#_Toc503452633)

[19.5.4 转储 OSD 运行图 166](#_Toc503452634)

[19.5.5 把 OSD 运行图转储为树 166](#_Toc503452635)

[19.5.6 找出某对象在哪里或应该在哪里 166](#_Toc503452636)

[19.5.7 增加或挪动一个新 OSD 条目 166](#_Toc503452637)

[19.5.8 从现有 CRUSH 图删除存在的条目（ OSD ） 166](#_Toc503452638)

[19.5.9 从现有 CRUSH 图删除存在的空桶 166](#_Toc503452639)

[19.5.10 把有效的桶从分级结构里的一个位置挪到另一个 167](#_Toc503452640)

[19.5.11 设置 {name} 所指条目的权重为 {weight} 167](#_Toc503452641)

[19.5.12 把 OSD 标记为丢失 167](#_Toc503452642)

[19.5.13 创建新 OSD 。如果未指定 ID ，有可能的话将自动分配个新 ID 。 167](#_Toc503452643)

[19.5.14 删除指定 OSD 。 167](#_Toc503452644)

[19.5.15 查询 OSD 运行图里的 max\_osd 参数。 167](#_Toc503452645)

[19.5.16 导入指定 CRUSH 图。 168](#_Toc503452646)

[19.5.17 设置 OSD 运行图的 max\_osd 参数，扩展存储集群时有必要。 168](#_Toc503452647)

[19.5.18 把 ID 为 {osd-num} 的 OSD 标记为 down 。 168](#_Toc503452648)

[19.5.19 把 OSD {osd-num} 标记为数据分布之外（即不给分配数据）。 168](#_Toc503452649)

[19.5.20 把 OSD {osd-num} 标记为数据分布之内（即分配了数据）。 168](#_Toc503452650)

[19.5.21 设置或清空 OSD 运行图里的暂停标记。 168](#_Toc503452651)

[19.5.22 把 {osd-num} 的权重设置为 {weight} 169](#_Toc503452652)

[19.5.23 重设所有滥用 OSD 的权重 169](#_Toc503452653)

[19.5.24 描述reweight-by-utilization会做什么 169](#_Toc503452654)

[19.5.26 增加、删除黑名单里的地址。 170](#_Toc503452655)

[19.5.27 创建/删除存储池快照。 170](#_Toc503452656)

[19.5.28 创建/删除/重命名存储池。 170](#_Toc503452657)

[19.5.29 更改存储池设置。 170](#_Toc503452658)

[19.5.30 获取存储池配置值。 171](#_Toc503452659)

[19.5.31 向 OSD {osd-num} 下达一个洗刷命令，用通配符 \* 把命令下达到所有 OSD 。 171](#_Toc503452660)

[19.5.32 向 osdN 下达修复命令，用 \* 下达到所有 OSD 。 171](#_Toc503452661)

[19.5.33 在 osdN 上进行个简单的吞吐量测试 171](#_Toc503452662)

[19.6 MDS子系统 172](#_Toc503452663)

[19.6.1 更改在运行 mds 的参数 172](#_Toc503452664)

[19.6.2 显示所有元数据服务器状态。 172](#_Toc503452665)

[19.6.3 活跃 MDS 失败时进行标记，如果有候补此命令会触发故障转移。 172](#_Toc503452666)

[19.7 监视器子系统 172](#_Toc503452667)

[19.7.1 查看监视器状态 172](#_Toc503452668)

[20. 监视器故障排除 175](#_Toc503452669)

[20.1 使用监视器的管理套接字 175](#_Toc503452670)

[20.2 理解 MON\_STATUS 176](#_Toc503452671)

[20.3 常见最监视器的问题 177](#_Toc503452672)

[20.3.1 有quorum但至少有一个监视器状态是down 177](#_Toc503452673)

[20.3.2 恢复监视器损坏的MONMAP 178](#_Toc503452674)

[20.3.3 偏移时钟 180](#_Toc503452675)

[20.3.4 客户端不能连接或挂载 181](#_Toc503452676)

[20.4 监视存储失败（★英文文档新增） 181](#_Toc503452677)

[20.4.1 存储损坏（STORE CORRUPTION）的症状 181](#_Toc503452678)

[20.4.2 使用健康的监视器进行恢复 182](#_Toc503452679)

[20.4.3 使用OSD进行恢复 182](#_Toc503452680)

[20.5 所有尝试都失败了，怎么办？ 183](#_Toc503452681)

[20.5.1 到外面寻求帮助 183](#_Toc503452682)

[20.5.2 收集所需日志 183](#_Toc503452683)

[20.5.3 我需要重启监视器来更改调试级别吗？ 184](#_Toc503452684)

[20.5.4 在某个调试级别下重现了问题，然后呢？ 185](#_Toc503452685)

[21. OSD故障排除 185](#_Toc503452686)

[21.1 收集OSD数据 185](#_Toc503452687)

[21.1.1 CEPH 日志 185](#_Toc503452688)

[21.1.2 管理套接字 185](#_Toc503452689)

[21.1.3 显示剩余空间 186](#_Toc503452690)

[21.1.4 I/O 统计信息 186](#_Toc503452691)

[21.1.5 诊断消息 186](#_Toc503452692)

[21.2 停止自动重均衡 187](#_Toc503452693)

[21.3 OSD 没运行 187](#_Toc503452694)

[21.3.1 OSD 起不来 187](#_Toc503452695)

[21.3.2 OSD 失败 188](#_Toc503452696)

[21.3.3 硬盘没剩余空间 189](#_Toc503452697)

[21.4 OSD 龟速或无响应 190](#_Toc503452698)

[21.4.1 网络问题 190](#_Toc503452699)

[21.4.2 驱动器配置 191](#_Toc503452700)

[21.4.3 坏扇区和碎片化硬盘 191](#_Toc503452701)

[21.4.4 监视器和 OSD 蜗居 191](#_Toc503452702)

[21.4.5 进程蜗居 191](#_Toc503452703)

[21.4.6 日志记录级别 192](#_Toc503452704)

[21.4.7 恢复节流 192](#_Toc503452705)

[21.4.8 内核版本 192](#_Toc503452706)

[21.4.9 内核与 SYNCFS 问题 192](#_Toc503452707)

[21.4.10 文件系统问题（★英文文档改动） 192](#_Toc503452708)

[21.4.11 内存不足 192](#_Toc503452709)

[21.4.12 old request或slow request 193](#_Toc503452710)

[21.4.13 调试慢请求（★英文文档新增） 194](#_Toc503452711)

[21.5 FLAPPING OSDS 195](#_Toc503452712)

[22. 归置组（pg）故障排除（▲中文文档未完全翻译） 196](#_Toc503452713)

[22.1 归置组总不整洁 196](#_Toc503452714)

[22.1.1 一个节点集群 196](#_Toc503452715)

[22.1.2 OSD少于副本数 197](#_Toc503452716)

[22.1.3 池大小=1 197](#_Toc503452717)

[22.1.4 CRUSH图错误 197](#_Toc503452718)

[22.2 卡住的归置组 197](#_Toc503452719)

[22.3 归置组挂了——互联失败 198](#_Toc503452720)

[22.4 未找到的对象 199](#_Toc503452721)

[22.5 无根归置组 201](#_Toc503452722)

[22.6 只有几个 OSD 接收数据 202](#_Toc503452723)

[22.7 不能写入数据 202](#_Toc503452724)

[22.8 归置组不一致（★英文文档新增详述） 202](#_Toc503452725)

[22.9 纠删编码的归置组不是 ACTIVE+CLEAN 205](#_Toc503452726)

[22.9.1 OSD 不够多 206](#_Toc503452727)

[22.9.2 CRUSH 条件不能满足 206](#_Toc503452728)

[22.9.3 CRUSH 过早中止 207](#_Toc503452729)

[23. 日志记录和调试（★英文文档命令有改动） 210](#_Toc503452730)

[23.1 运行时 210](#_Toc503452731)

[23.2 启动时 211](#_Toc503452732)

[23.3 加快日志更迭 212](#_Toc503452733)

[23.4 VALGRIND 213](#_Toc503452734)

[23.5 子系统、日志和调试选项 213](#_Toc503452735)

[23.5.1 CEPH 子系统概览（★英文文档内存日志级别有改动） 213](#_Toc503452736)

[23.5.2 日志记录选项 215](#_Toc503452737)

[23.5.3 OSD 218](#_Toc503452738)

[23.5.4 FILESTORE 219](#_Toc503452739)

[23.5.5 MDS 220](#_Toc503452740)

[23.5.6 RADOS 网关 221](#_Toc503452741)

[24. CPU剖析 222](#_Toc503452742)

[24.1 初始化 OPROFILE 222](#_Toc503452743)

[24.2 启动 OPROFILE 222](#_Toc503452744)

[24.3 停止 OPROFILE 223](#_Toc503452745)

[24.4 查看 OPROFILE 运行结果 223](#_Toc503452746)

[24.5 重置 OPROFILE 223](#_Toc503452747)

[25. 内存剖析 224](#_Toc503452748)

[25.1 启动剖析器 225](#_Toc503452749)

[25.2 打印统计信息 225](#_Toc503452750)

[25.3 转储堆栈信息 226](#_Toc503452751)

[25.4 释放内存 226](#_Toc503452752)

[25.5 停止剖析器 227](#_Toc503452753)

# 序：内容导览

## 高级运维

高级集群操作主要包括用 ceph 服务管理脚本启动、停止、重启集群，和集群健康状态检查、监控和操作集群。

* [操纵集群](#_1._操纵集群（★英文文档改动）)
* [健康检查](#_2._健康检查（★英文文档新增）)
* [监控集群](#_3._监控集群)
* [监控 OSD 和归置组](#_4._监控_OSD)
* [用户管理](#_5._用户管理（▲中文文档未翻译）)
* [修复PG不一致](#_6._修复PG不一致（▲英文文档只有标题，内容为空）)

## 数据归置

你的集群开始运行后，就可以尝试数据归置了。 Ceph 是 PB 级数据存储集群，它用 CRUSH 算法、靠存储池和归置组在集群内分布数据。

* [数据归置概览](#_7._数据归置（DATA_PLACEMENT）概览)
* [存储池](#_8._存储池（★英文文档改动）)
* [纠删码](#_9._纠删码（▲中文文档未翻译）)
* [分级缓存](#_10._分级缓存)
* [归置组](#_11._归置组（pg）（★英文文档改动）)
* [使用PG-UPMAP](#_12._使用PG-UPMAP（★英文文档新增）)
* [CRUSH 图](#_13._CRUSH_图)
* [手动编辑CRUSH映射](#_14._手动编辑CRUSH映射（★英文文档新增）)
* [在不同的OSD上放置不同的池](#_15._在不同的OSDS上放置不同的池（★英文文档新增）)

## 低级运维

低级集群运维包括启动、停止、重启集群内的某个具体守护进程；更改某守护进程或子系统配置；增加或拆除守护进程。低级运维还经常遇到扩展、缩减 Ceph 集群，以及更换老旧、或损坏的硬件。

* [增加/删除 OSD](#_16._增加/删除_OSD)
* [增加/删除监视器](#_17._增加/删除监视器)
* [BlueStore迁移](#_18._BLUESTORE迁移（★英文文档新增）)
* [命令参考](#_19._命令参考)

## 故障排除

Ceph 仍在积极开发中，所以你可能碰到一些问题，需要评估 Ceph 配置文件、并修改日志和调试选项来纠正它。

* Ceph 社区
* [监视器故障排除](#_20._监视器故障排除)
* [OSD 故障排除](#_21._OSD故障排除)
* [归置组故障排除](#_22._归置组（pg）故障排除（▲中文文档未完全翻译）)
* [日志记录和调试](#_23._日志记录和调试（★英文文档命令有改动）)
* [CPU 剖析](#_24._CPU剖析)
* [内存剖析](#_25._内存剖析)

# 1. 操纵集群（★英文文档改动）

## 1.1 用systemd运行ceph

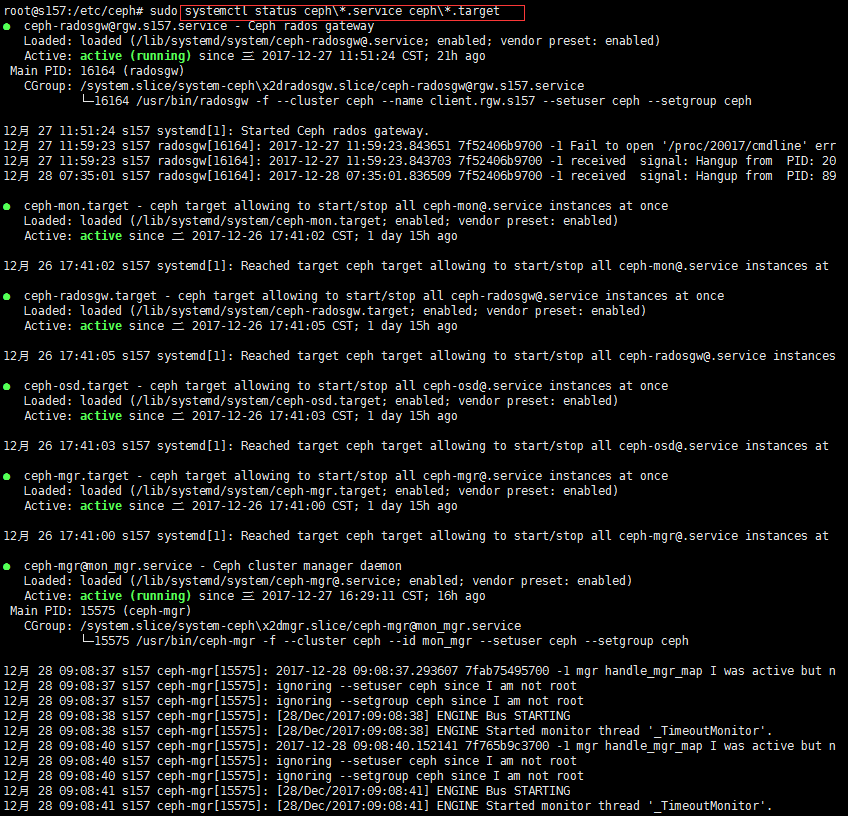
对于所有支持systemd（CentOS 7，Fedora，Debian Jessie 8及更高版本，SUSE）的发行版，现在Ceph使用本地systemd文件而不是旧版sysvinit脚本来管理ceph守护进程。例如：

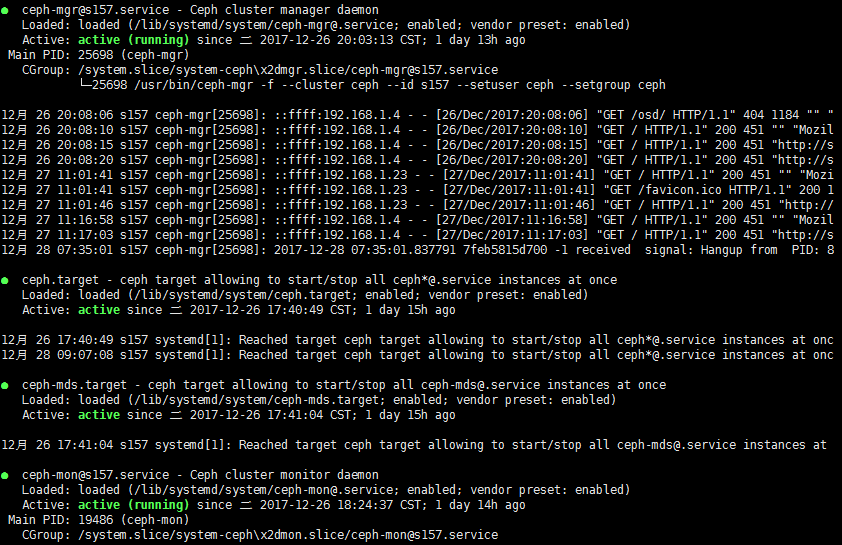
sudo systemctl start ceph.target *# start all daemons*

sudo systemctl status ceph-osd@12 *# check status of osd.12*

### 1.1.1 在节点上列出ceph systemd单元

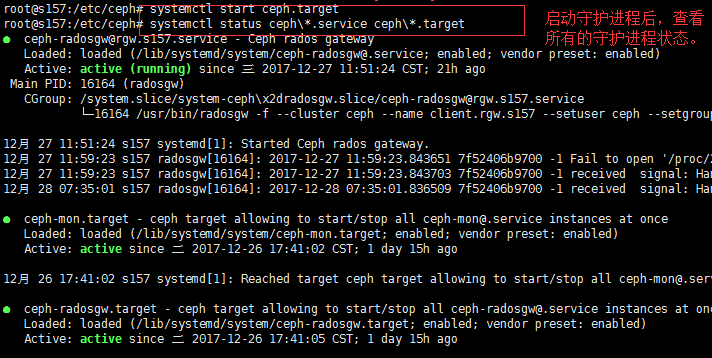
sudo systemctl status ceph\\*.service ceph\\*.target





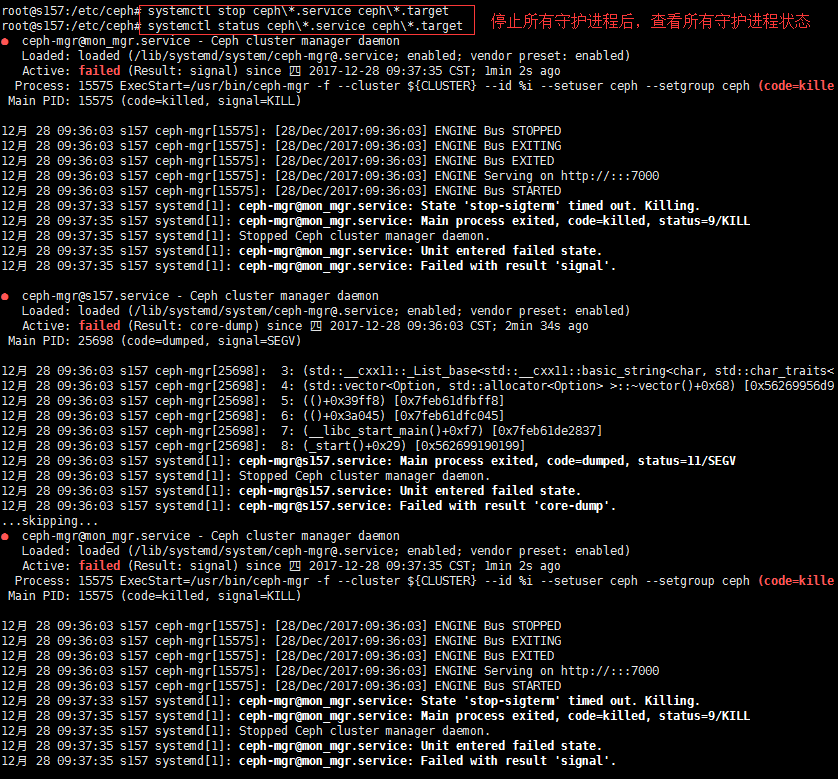
### 1.1.2 启动Ceph节点上的所有守护进程（不管类型）

sudo systemctl start ceph.target



### 1.1.3 停止Ceph节点上的所有守护进程（不管类型）

sudo systemctl stop ceph\\*.service ceph\\*.target



### 1.1.4 按类型启动守护进程

要启动Ceph节点上特定类型的所有守护进程，请执行以下操作之一：

sudo systemctl start ceph-osd.target

sudo systemctl start ceph-mon.target

sudo systemctl start ceph-mds.target

### 1.1.5 按类型停止守护进程

要停止Ceph节点上的所有特定类型的守护进程，请执行以下操作之一：

sudo systemctl stop ceph-mon\\*.service ceph-mon.target

sudo systemctl stop ceph-osd\\*.service ceph-osd.target

sudo systemctl stop ceph-mds\\*.service ceph-mds.target

### 1.1.6 启动单个进程

要启动某节点上一指定守护进程例程，用下列命令之一：

sudo systemctl start ceph-osd@{id}

sudo systemctl start ceph-mon@{hostname}

sudo systemctl start ceph-mds@{hostname}

例如：

sudo systemctl start ceph-osd**@1**

sudo systemctl start ceph-mon**@ceph**-server

sudo systemctl start ceph-mds**@ceph**-server

### 1.1.7 停止单个进程

要停止某节点上一指定守护进程例程，用下列命令之一：

sudo systemctl stop ceph-osd@{id}

sudo systemctl stop ceph-mon@{hostname}

sudo systemctl stop ceph-mds@{hostname}

例如：

sudo systemctl stop ceph-osd**@1**

sudo systemctl stop ceph-mon**@ceph**-server

sudo systemctl stop ceph-mds**@ceph**-server

## 1.2 用upstart运行ceph

**|----------------------------------\*----------------------------------\*----------------------------------\*----------------------------------|**

**编者注**：用upstart也可进行ceph运维，但经实际测试：



执行该命令出错。

这是因为在ubuntu中，upstart逐步被Systemd取代，在本文档中，仍列出以下用ceph upstart的运维命令，按君所好取用。

**|----------------------------------\*----------------------------------\*----------------------------------\*----------------------------------|**

ceph-deploy在Ubuntu Trusty上部署Ceph时，可以使用基于事件的[Upstart](http://upstart.ubuntu.com/index.html)在[Ceph节点](http://docs.ceph.com/docs/master/glossary/#term-ceph-node)上启动和停止Ceph守护进程。Upstart不要求您在Ceph配置文件中定义守护进程实例。

要在节点上列出Ceph Upstart作业和实例，请执行：

sudo initctl list | grep ceph

有关更多详细信息，请参阅[initctl](http://manpages.ubuntu.com/manpages/raring/en/man8/initctl.8.html)。

### 1.2.1 启动所有守护程序

要启动Ceph节点上的所有守护进程（不管类型），请执行以下操作：

sudo start ceph-all

### 1.2.2 停止所有的守护程序

要停止Ceph节点上的所有守护进程（不管类型），请执行以下操作：

sudo stop ceph-all

### 1.2.3 按类型启动所有守护进程

要启动Ceph节点上特定类型的所有守护进程，请执行以下操作之一：

sudo start ceph-osd-all

sudo start ceph-mon-all

sudo start ceph-mds-all

### 1.2.4 按类型停止所有守护进程

要停止Ceph节点上的所有特定类型的守护进程，请执行以下操作之一：

sudo stop ceph-osd-all

sudo stop ceph-mon-all

sudo stop ceph-mds-all

### 1.2.5 启动守护进程

要在Ceph节点上启动特定的守护进程实例，请执行以下操作之一：

sudo start ceph-osd id={id}

sudo start ceph-mon id={hostname}

sudo start ceph-mds id={hostname}

例如：

sudo start ceph-osd id=1

sudo start ceph-mon id=ceph-server

sudo start ceph-mds id=ceph-server

### 1.2.6 停止守护程序

要停止Ceph节点上的特定守护程序实例，请执行以下操作之一：

sudo stop ceph-osd id={id}

sudo stop ceph-mon id={hostname}

sudo stop ceph-mds id={hostname}

例如：

sudo stop ceph-osd id=1

sudo start ceph-mon id=ceph-server

sudo start ceph-mds id=ceph-server

## 1.2 运行ceph

每次用命令启动、重启、停止Ceph 守护进程（或整个集群）时，你必须指定至少一个选项和一个命令，还可能要指定守护进程类型或具体例程。

{commandline} [options] [commands] [daemons]

ceph 选项包括：

|  |  |  |
| --- | --- | --- |
| **选项** | **简写** | **描述** |
| --verbose | -v | 详细的日志。 |
| --valgrind | N/A | （只适合开发者和质检人员）用 Valgrind 调试。 |
| --allhosts | -a | 在 ceph.conf 里配置的所有主机上执行，否 则它只在本机执行。 |
| --restart | N/A | 核心转储后自动重启。 |
| --norestart | N/A | 核心转储后不自动重启。 |
| --conf | -c | 使用另外一个配置文件。 |

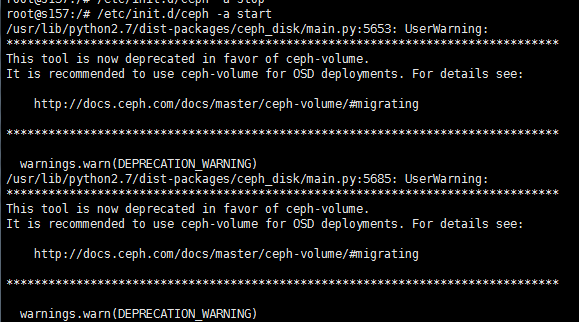
ceph 命令包括：

|  |  |
| --- | --- |
| **选项** | **描述** |
| start | 启动守护进程。 |
| stop | 停止守护进程。 |
| forcestop | 暴力停止守护进程，等价于 kill -9 |
| killall | 杀死某一类守护进程。 |
| cleanlogs | 清理掉日志目录。 |
| cleanalllogs | 清理掉日志目录内的所有文件。 |

至于子系统操作， ceph 服务能指定守护进程类型，在 [daemons] 处指定守护进程类型就行了，守护进程类型包括：

* mon
* osd
* mds

在ceph中文文档中，这里有一部分弃用的命令：



可使用ceph-volume：<http://docs.ceph.com/docs/master/ceph-volume/#migrating>

# 2. 健康检查（★英文文档新增）

## 2.1 概述

Ceph集群产生的健康信息可能有限，这些信息被定义为具有唯一标识符的健康检查。

标识符是一个简洁的伪人类可读（例如：就像一个变量名称）字符串。它的目的是使工具（如用户界面）具有健康检查的意义，并以反映其含义的方式呈现。

此页面列出了由监视器和管理守护进程引发的健康检查。除此之外，您还可以看到源自MDS守护进程（元数据守护进程）的健康检查（请参阅[CephFS健康信息](http://docs.ceph.com/docs/master/cephfs/health-messages/#cephfs-health-messages)）以及由ceph-mgr python模块定义的健康状况检查。

## 2.2 定义

### 2.2.1 OSDS

#### 2.2.1.1 OSD\_DOWN

一个或多个OSD被标记为down。ceph-osd守护程序可能已经停止，或者对应的OSD可能无法通过网络到达OSD。常见原因包括停止或崩溃的守护程序，停机主机或网络中断。

验证主机是否正常，守护程序已启动，网络是否正常工作。如果守护进程已经崩溃，守护进程日志文件（/var/log/ceph/ceph-osd.\*）可能包含调试信息。

#### 2.2.1.2 OSD\_<CRUSH TYPE>\_DOWN

（例：OSD\_HOST\_DOWN, OSD\_ROOT\_DOWN）

特定CRUSH子树内的所有OSD都被标记为DOWN，例如同一个主机上的所有OSD。

#### 2.2.1.3 OSD\_ORPHAN

OSD在CRUSH地图层次结构中被引用，但其实不存在。

OSD可以用如下命令从CRUSH层次结构中删除：

ceph osd crush rm osd.<id>

#### 2.2.1.4 OSD\_OUT\_OF\_ORDER\_FULL

backfillfull， nearfull， full， and/or failsafe\_full 的利用率阈值不是升序，特别是我们希望大小关系为backfillfull < nearfull， nearfull < full， and full < failsafe\_full

阈值可以用以下命令进行调整：

ceph osd set-backfillfull-ratio <ratio>

ceph osd set-nearfull-ratio <ratio>

ceph osd set-full-ratio <ratio>

#### 2.2.1.5 OSD\_FULL

一个或多个OSD已超出完整阈值，并阻止集群为写入服务。

可以使用以下方法检查池的使用情况：

ceph df

目前定义的完整比例可以用如下命令看到：

ceph osd dump | grep full\_ratio

恢复写入可用性的短期解决方法是将整个阈值提高一小部分：

ceph osd set-full-ratio <ratio>

新的存储应该通过部署更多的OSD来添加到集群中，或者应该删除现有的数据以释放空间。

#### 2.2.1.6 OSD\_BACKFILLFULL

一个或多个OSD已超出回填阈值（backfillfull threshold），这将防止数据被允许重新平衡到此设备。这是一个早期的警告，即重新平衡可能无法完成，而且该集群容量正在接近满值。

可以使用以下方法检查池的使用情况：

ceph df

#### 2.2.1.7 OSD\_NEARFULL

一个或多个OSD已经超过了近乎满的阈值（nearfull threshold）。这是一个早期的警告，说明集群容量正在接近满值。

可以使用以下方法检查池的使用情况：

ceph df

#### 2.2.1.8 OSDMAP\_FLAGS

已经设置了一个或多个集群的标志，这些标志包括：

* full – 集群被标记为已满，不能写入服务
* pauserd, pausewr – 暂停读取或写入
* noup – OSD不允许启动
* nodown – 忽略OSD故障报告，使得监视器不会将OSD标记为down
* noin – 之前被标记为out的OSD在它们启动时不会被标记回in
* noout – 在配置的时间间隔之后，状态为down的OSD不会被自动标记为out
* nobackfill, norecover, norebalance –暂停恢复数据或数据重平衡
* noscrub, nodeep\_scrub – 禁用清理
* notieragent – 暂停缓存分层活动

除了full状态，这些标志可以设置或清除：

ceph osd set <flag>

ceph osd unset <flag>

#### 2.2.1.9 OSD\_FLAGS

One or more OSDs has a per-OSD flag of interest set. These flags include:

* noup: OSD 不允许启动
* nodown: 忽略该OSD的故障报告
* noin:如果这个OSD之前在发生故障时被自动标记为down，当它启动时将不会再被标记为in
* noout: 在配置的时间间隔之后，状态为down的OSD不会被自动标记为out

每个标志可以设置或清除：

ceph osd add-<flag> <osd-id>

ceph osd rm-<flag> <osd-id>

例如：

ceph osd rm-nodown osd.123

#### 2.2.1.10 OLD\_CRUSH\_TUNABLES

CRUSH映射在使用非常旧的设置，应该更新。由mon\_crush\_min\_required\_versionconfig选项确定不会触发此健康警告的可以使用的最旧的可调参数（即可以连接到集群的最旧的客户端版本）。有关更多信息，请参阅[可调参数](http://docs.ceph.com/docs/master/rados/operations/crush-map/#crush-map-tunables)。

#### 2.2.1.11 OLD\_CRUSH\_STRAW\_CALC\_VERSION

CRUSH图使用较老的非最佳方法来计算straw桶的中间权重值。

应该更新CRUSH映射以使用更新的方法（straw\_calc\_version=1）。有关更多信息，请参阅 [可调参数](http://docs.ceph.com/docs/master/rados/operations/crush-map/#crush-map-tunables)。

#### 2.2.1.12 CACHE\_POOL\_NO\_HIT\_SET

一个或多个高速缓存池（cache pools）没有配置*hit set*来跟踪利用率，这将阻止分层代理识别冷对象以便从高速缓存中清除并逐出。

*hit set*可以在缓存池中配置：

ceph osd pool set <poolname> hit\_set\_type <type>

ceph osd pool set <poolname> hit\_set\_period <period-in-seconds>

ceph osd pool set <poolname> hit\_set\_count <number-of-hitsets>

ceph osd pool set <poolname> hit\_set\_fpp <target-false-positive-rate>

#### 2.2.1.13 OSD\_NO\_SORTBITWISE

没有pre-luminous v12.y.z版本的 OSDs在运行，但是sortbitwise标志还没有设置。

sortbitwise标志必须在luminous v12.y.z或更新的OSD可以启动之前设置，您可以用以下命令安全地设置标志：

ceph osd set sortbitwise

#### 2.2.1.14 POOL\_FULL

一个或多个池已达到配额，不再允许写入。

池配额和利用率可以用以下命令看出：

ceph df detail

您可以通过以下方式提高池配额：

ceph osd pool set-quota <poolname> max\_objects <num-objects>

ceph osd pool set-quota <poolname> max\_bytes <num-bytes>

或删除一些现有数据以减少利用率。

### 2.2.2 数据健康 (池 & PGs)

#### PG\_AVAILABILITY

数据可用性降低，这意味着集群无法为集群中的某些数据提供潜在的读取或写入请求。具体而言，一个或多个PG处于不允许IO服务请求的状态。有问题的PG状态包括*peering*, *stale*, *incomplete*,，缺乏*active*（如果这些条件不能很快清除）。

有关哪些PG受到影响的详细信息可从以下命令获得：

ceph health detail

在大多数情况下，根本原因是一个或多个OSD正在关闭；可以看2.2.1.1关于OSD\_DOWN的讨论。

具体PG的状态可以用如下命令查询：

ceph tell <pgid> query

#### PG\_DEGRADED

对于一些数据来说，冗余减少了，这意味着集群没有针对所有数据（对于复制的池）或纠删码片段（对于纠删码池）所需数量的副本(副本数不够)。具体而言，一个或多个PG：

* 已经设置了降级（degraded ）标志或缩小（undersized ）标志，这意味着该集群中没有足够的该PG实例。
* 有一段时间没有clean的标志。

有关哪些PG受到影响的详细信息可从以下命令获得：

ceph health detail

在大多数情况下，根本原因是一个或多个OSD正在关闭；可以看2.2.1.1关于OSD\_DOWN的讨论。

具体PG的状态可以用如下命令查询：

ceph tell <pgid> query

#### PG\_DEGRADED\_FULL

由于集群中缺少空闲空间，有些数据可能存在数据冗余或减少的风险。具体而言，一个或多个PG具有 backfill\_toofull或recovery\_toofull标志，这意味着该集群无法迁移或恢复数据，因为一个或多个OSD高于回填（*backfillfull*）阈值。

请参阅上面对OSD\_BACKFILLFULL或OSD\_FULL的讨论，了解解决此问题的步骤。

#### PG\_DAMAGED

数据清理已经发现了集群中数据一致性的一些问题。具体而言，一个或多个的PGs具有*不一致（inconsistent）*或被标记为 *snaptrim\_error*，表示较早的清洗操作中发现了问题；或者说，若标记为*修复（repair）*，意味着目前正在进行一个用于这种不一致的修复。

请参阅[修复PG不一致问题](http://docs.ceph.com/docs/master/rados/operations/pg-repair/)以获取更多信息。

#### OSD\_SCRUB\_ERRORS

最近的OSD清洗已经发现了不一致之处。这个错误通常与*PG\_DAMANGED*配对（见上文）。

请参阅[修复PG不一致问题](http://docs.ceph.com/docs/master/rados/operations/pg-repair/)以获取更多信息。

#### CACHE\_POOL\_NEAR\_FULL

缓存层存储池已满。这是由缓存池中的target\_max\_bytes和target\_max\_objects属性决定的。一旦池达到目标阈值，刷新数据并从缓存中释放时，向池中写入请求可能会阻塞，这种状态通常会导致非常高的延迟和较差性能。

缓存池的目标大小可以通过以下方式调整：

ceph osd pool set <cache-pool-name> target\_max\_bytes <bytes>

ceph osd pool set <cache-pool-name> target\_max\_objects <objects>

由于基础层的可用性或性能的降低或整体集群负载的降低，也可能限制正常缓存刷新和逐出活动。

#### TOO\_FEW\_PGS

集群中使用的PG的数量低于每个OSD PGs的mon\_pg\_warn\_min\_per\_osd可配置阈值。这可能导致整个集群中OSD的数据分布不均匀和不平衡，同样会降低总体性能。

如果数据池尚未创建，这可能是一个期望的条件。

可以增加现有池的PG计数，也可以创建新的池。请参阅[选择展示位置组的数量以](http://docs.ceph.com/docs/master/rados/operations/placement-groups/#choosing-number-of-placement-groups)获取更多信息。

#### TOO\_MANY\_PGS

集群中使用的PG数量高于每个OSD PGs的mon\_max\_pg\_per\_osd可配置阈值。如果超过此阈值，集群将不允许创建新的池，增加池的pg\_num，增加池副本（其中任何一个都会导致集群有更多的PG）。大量的PG可能会导致OSD守护进程的内存利用率提高，集群状态更改（如OSD重新启动，添加或删除）后同步更慢以及Manager和Monitor守护进程负载更高。

减轻问题最简单的方法是通过增加更多的硬件来增加集群中OSD的数量。请注意，用于此健康检查的OSD计数是“in”OSD的数量，因此将“out”OSDs标记为“in”也会有所帮助：

ceph osd in <osd id(s)>

请参阅[选择展示位置组的数量以](http://docs.ceph.com/docs/master/rados/operations/placement-groups/#choosing-number-of-placement-groups)获取更多信息。

#### SMALLER\_PGP\_NUM

一个或多个池的pgp\_num值小于pg\_num。这通常表示PG计数增加而不增加放置行为。

当PG数量从数据迁移中进行调整时，这有时是有意识地将*split*步骤分离出来。

This is sometimes done deliberately to separate out the *split* step when the PG count is adjusted from the data migration that is needed when pgp\_num is changed.

这通常通过设置pgp\_num匹配pg\_num，触发数据迁移来解决：

ceph osd pool set <pool> pgp\_num <pg-num-value>

#### MANY\_OBJECTS\_PER\_PG

一个或多个池中，每个PG的平均对象数明显高于整个集群的平均值。具体的阈值由mon\_pg\_warn\_max\_object\_skew 配置值控制。

这通常表示集群中包含大部分数据的池的PG数量太少，并且/或包含没那么多数据的其他池的PG数量太多。请参阅上面关于TOO\_MANY\_PGS的讨论 。

可以通过调整监视器上mon\_pg\_warn\_max\_object\_skew的配置选项来提高阈值以使健康警告消失。

#### POOL\_APP\_NOT\_ENABLED

存在包含一个或多个对象的池，但尚未为特定应用程序标记供其使用。

通过标记池以供应用程序使用来解决此警告。例如，如果池被RBD使用，则：

rbd pool init <poolname>

如果池正在被自定义应用程序“foo”使用，您还可以通过底层命令进行标注：

ceph osd pool application enable foo

有关更多信息，请参阅将[池关联到应用程序](http://docs.ceph.com/docs/master/rados/operations/pools/#associate-pool-to-application)。

#### POOL\_FULL

一个或多个池已达到（或接近达到）其配额。触发此错误条件的阈值由mon\_pool\_quota\_crit\_threshold配置选项控制。

池配额可以调整上或下（或删除）：

ceph osd pool set-quota <pool> max\_bytes <bytes>

ceph osd pool set-quota <pool> max\_objects <objects>

将配额值设置为0将禁用配额。

#### POOL\_NEAR\_FULL

一个或多个池正在接近配额。触发此警告条件的阈值由mon\_pool\_quota\_warn\_threshold配置选项控制 。

池配额可以调整上或下（或删除）：

ceph osd pool set-quota <pool> max\_bytes <bytes>

ceph osd pool set-quota <pool> max\_objects <objects>

将配额值设置为0将禁用配额。

#### OBJECT\_MISPLACED

集群中的一个或多个对象不存储在集群希望存储的节点上。这表明由于最近的集群变更导致的数据迁移尚未完成。

放错位置的数据本身并不是一个危险的状况；数据一致性永远不会受到威胁，并且直到所需数量的新副本（在所需的位置）出现为止，永远不会删除旧的对象副本。

#### OBJECT\_UNFOUND

无法找到集群中的一个或多个对象。具体而言，OSD知道一个对象新的或更新的副本应该存在，但是在当前在线的OSD上没有找到该对象版本的副本。

解除对象的读取或写入请求将被阻止。

理想情况下，可以重新联机一个状态为down的OSD，其中包含未找到的对象的最近副本。候选OSD可以从负责未发现对象的PG的对等状态中识别出来（Candidate OSDs can be identified from the peering state for the PG(s) responsible for the unfound object）：

ceph tell <pgid> query

如果对象的最新副本不可用，则可以通知集群回滚到该对象的先前版本。有关更多信息，请参阅未[找到的对象](http://docs.ceph.com/docs/master/rados/troubleshooting/troubleshooting-pg/#failures-osd-unfound)。

#### SLOW\_OPS

一个或多个OSD请求需要很长时间才能处理。这可能是极端负载，缓慢的存储设备或软件错误的指示。

OSD上的请求队列可以通过OSD主机执行以下命令查询：

ceph daemon osd.<id> ops

可以用以下命令看到最近最慢请求的摘要：

ceph daemon osd.<id> dump\_historic\_ops

OSD的位置可以通过以下方式找到：

ceph osd find osd.<id>

#### PG\_NOT\_SCRUBBED

一个或多个PGs最近还没有被清洗过。PGs通常每隔mon\_scrub\_interval秒清洗一次，而这个警告会在过了mon\_warn\_not\_scrubbed这几秒后还没有清洗时触发。

如果PG没有被标记为clean，PG将不会清洗；这可能会在放错地方或降级（如上面的PG\_AVAILABILITY和 PG\_DEGRADED）时发生。

您可以手动开始清洗一个PG：

ceph pg scrub <pgid>

#### PG\_NOT\_DEEP\_SCRUBBED

一个或多个PGs最近没有被深度清洗过。PGs通常每隔osd\_deep\_mon\_scrub\_interval秒清洗一次，而这个警告会在过了mon\_warn\_not\_deep\_scrubbed这几秒后还没有清洗时触发。

如果PG没有被标记为clean，则PG将不会（深度）清洗；这可能会在它们放错位置或降级（参见上面的PG\_AVAILABILITY和 PG\_DEGRADED）时发生。

您可以手动开始清洗一个干净的PG：

ceph pg deep-scrub <pgid>

# 3. 监控集群

监视集群通常涉及检查OSD状态，监视状态，放置组状态和元数据服务器状态。

## 3.1 使用命令行

### 3.1.1 交互模式

在交互模式下运行 ceph ，不带参数运行 ceph ，例如：

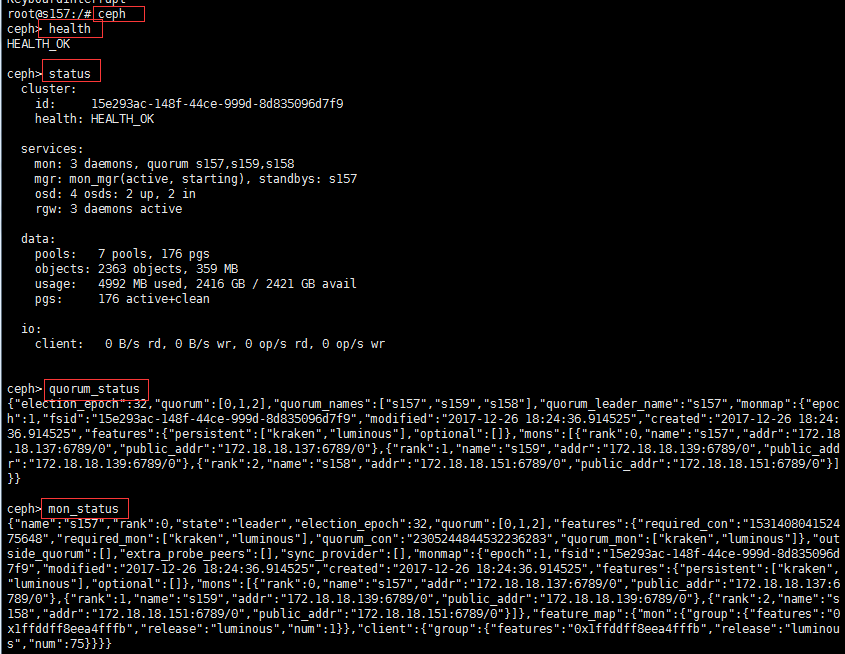
ceph

ceph> health

ceph> status

ceph> quorum\_status

ceph> mon\_status



### 3.1.2 非默认路径

如果您为配置或密钥环指定了非默认位置，则可以指定其位置：

ceph -c /path/to/conf -k /path/to/keyring health

## 3.2 检查集群的状态（★英文文档新增）

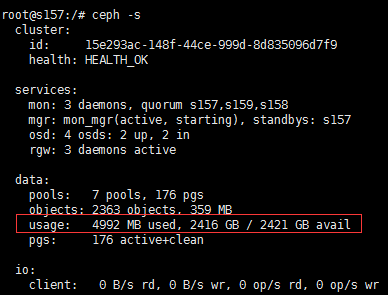
在启动集群之后，在开始读取和/或写入数据之前，请首先检查集群的状态。

要检查集群的状态，请执行以下操作：

ceph status

或：

ceph -s



注：

PGs：Placement Group 存储对象集合

Ceph如何计算数据的使用

used 值反映了确实已占用的原始存储空间。 xxx GB / xxx GB 值则是剩余空间（较小的数）与集群总容量的比较。理论数值反映了所存储数据的原始尺寸，未计算其副本、克隆、或快照空间，所以数据存储实际占用的空间通常会超过理论数值，因为 Ceph 会自动创建数据副本，另外存储空间也可能用于克隆和快照。

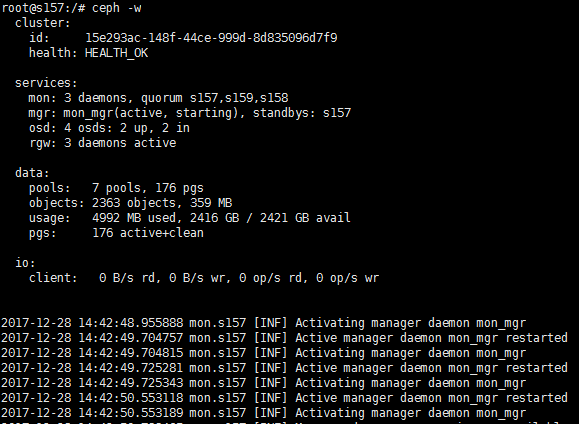
## 3.3 观察集群

除了每个守护进程的本地日志之外，Ceph集群还维护一个*集群日志*，记录整个系统的高级事件。这将记录到监视器服务器上的磁盘（默认为/var/log/ceph/ceph.log），但也可以通过命令行进行监视。

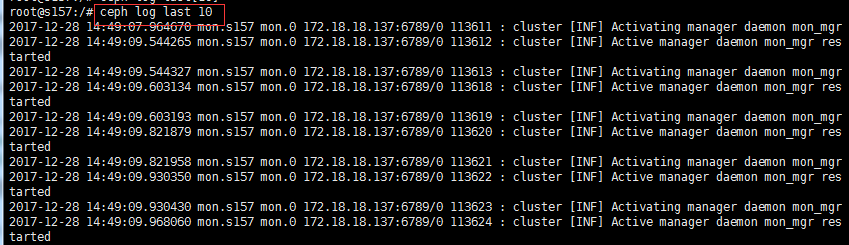
要跟踪集群日志，请使用以下命令：

ceph -w

Ceph将打印系统的状态，然后打印每个日志消息。 例如：



除了使用ceph -w发送日志行之外，还可以使用ceph log last [n]查看集群日志中最近的n行。



## 3.4 检测健康检查

Ceph持续对自己的状态进行各种*健康检查*。当健康检查失败时，这反映在ceph status（或 ceph health）的输出中。此外，还会将消息发送到集群日志，以指示检查何时失败以及集群何时恢复。

例如，当OSD下降时，health状态输出部分可以更新如下：

health: HEALTH\_WARN

1 osds down

Degraded data redundancy: 21/63 objects degraded (33.333%), 16 pgs unclean, 16 pgs degraded

此时还会发出集群日志消息，以记录运行状况检查的失败情况：

2017-07-25 10:08:58.265945 mon.a mon.0 172.21.9.34:6789/0 91 : cluster [WRN] Health check failed: 1 osds down (OSD\_DOWN)

2017-07-25 10:09:01.302624 mon.a mon.0 172.21.9.34:6789/0 94 : cluster [WRN] Health check failed: Degraded data redundancy: 21/63 objects degraded (33.333%), 16 pgs unclean, 16 pgs degraded (PG\_DEGRADED)

当OSD重新联机时，集群日志记录集群返回到健康状态：

2017-07-25 10:11:11.526841 mon.a mon.0 172.21.9.34:6789/0 109 : cluster [WRN] Health check update: Degraded data redundancy: 2 pgs unclean, 2 pgs degraded, 2 pgs undersized (PG\_DEGRADED)

2017-07-25 10:11:13.535493 mon.a mon.0 172.21.9.34:6789/0 110 : cluster [INF] Health check cleared: PG\_DEGRADED (was: Degraded data redundancy: 2 pgs unclean, 2 pgs degraded, 2 pgs undersized)

2017-07-25 10:11:13.535577 mon.a mon.0 172.21.9.34:6789/0 111 : cluster [INF] Cluster **is** now healthy

## 3.5 检测配置问题

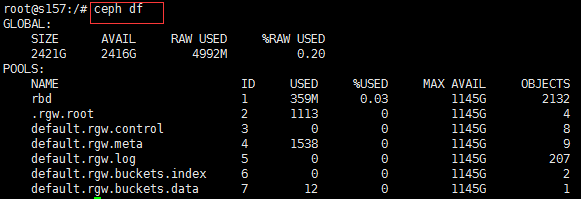
除了Ceph以自身状态持续运行的健康检查之外，还有一些配置问题只能通过外部工具检测到。

使用[ceph-medic](http://docs.ceph.com/ceph-medic/master/)工具在Ceph集群的配置上运行这些额外的检查。

## 3.6 检查集群的使用情况统计

要检查集群的数据使用情况和数据池之间的分配情况，可以使用该df选项。它类似于Linux df。执行以下操作：

ceph df



输出的 **GLOBAL** 段展示了数据所占用集群存储空间的概要。

* **SIZE:** 集群的总容量；
* **AVAIL:** 集群的空闲空间总量；
* **RAW USED:** 已用存储空间总量；
* **% RAW USED:** 已用存储空间比率。用此值参照 full ratio 和 near full \ ratio 来确保不会用尽集群空间。详情见[存储容量](http://docs.ceph.org.cn/rados/configuration/mon-config-ref#storage-capacity)。

输出的 **POOLS** 段展示了存储池列表及各存储池的大致使用率。本段**没有**展示副本、克隆品和快照占用情况。例如，如果你把 1MB 的数据存储为对象，理论使用率将是 1MB ，但考虑到副本数、克隆数、和快照数，实际使用率可能是 2MB 或更多。

* **NAME:** 存储池名字；
* **ID:** 存储池唯一标识符；
* **USED:** 大概数据量，单位为 KB 、 MB 或 GB ；
* **%USED:** 各存储池的大概使用率；
* **Objects:** 各存储池内的大概对象数。

**Note**

**POOLS** 段内的数字是理论值，它们不包含副本、快照或克隆。因此，它与 **USED** 和 **%USED** 数量之和不会达到**GLOBAL** 段中的 **RAW USED** 和 **%RAW USED** 数量。

## 3.7 检查OSD状态（★英文文档改动）

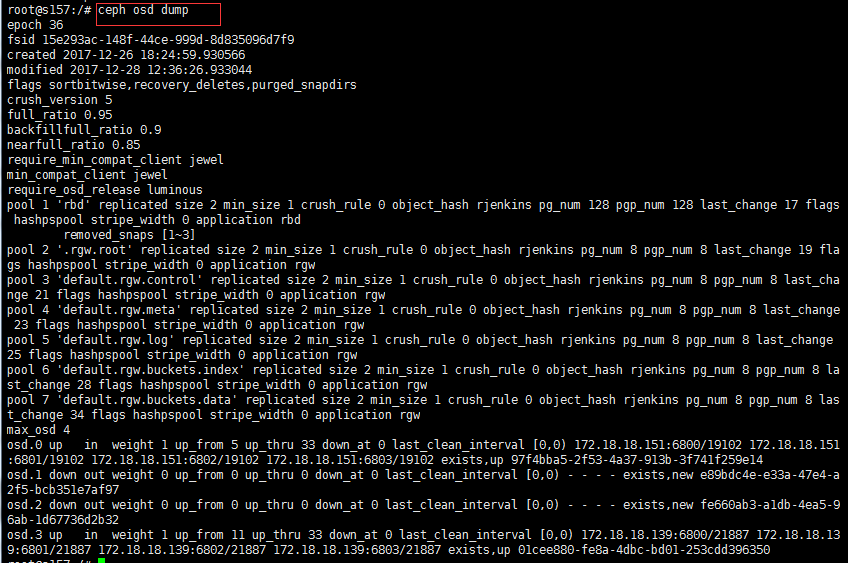
可以执行下列命令来确定 OSD 状态为 up 且 in ：

ceph osd stat



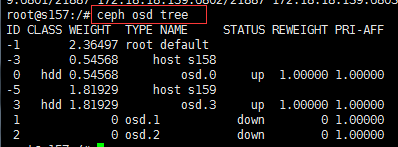
或：

ceph osd dump



也可以根据 OSD 在 CRUSH 图里的位置来查看：

ceph osd tree



详见4. 监控 OSD 和归置组。

## 3.8 检查监视器状态

如果你有多个监视器（很可能），你启动集群后、读写数据前应该检查监视器法定人数状态。运行着多个监视器时必须形成法定人数，最好周期性地检查监视器状态来确定它们在运行。

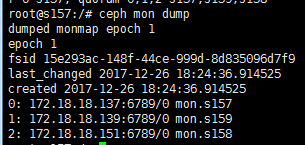
要查看监视器图，执行下面的命令：

ceph mon stat



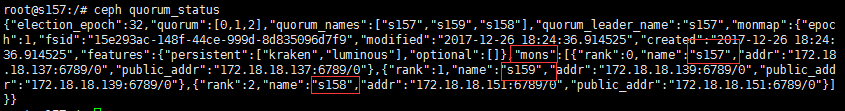
或者：

ceph mon dump



要检查监视器的法定人数状态，执行下面的命令：

ceph quorum\_status



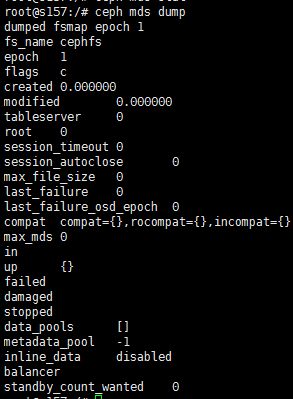
## 3.9 检查MDS状态

元数据服务器为 Ceph 文件系统提供元数据服务，元数据服务器有两种状态： up | down 和 active | inactive ，执行下面的命令查看元数据服务器状态为 up 且 active ：

ceph mds stat

要展示元数据集群的详细状态，执行下面的命令：

ceph mds dump



## 3.10 检查归置组状态

归置组把对象映射到 OSD 。监控归置组时，我们希望它们的状态是 active 且 clean 。个中详情见4.监控 OSD 和归置组。

## 3.11 使用管理套接字（ADMIN SOCKET）

Ceph 管理套接字允许你通过套接字接口查询守护进程，它们默认存在于 /var/run/ceph 下。要通过管理套接字访问某个守护进程，先登录它所在的主机、再执行下列命令：

ceph daemon {daemon-name}

ceph daemon {path-to-socket-file}

比如，这是下面这两种用法是等价的：

ceph daemon osd.0 foo

ceph daemon /var/run/ceph/ceph-osd.0.asok foo

用下列命令查看可用的管理套接字命令：

ceph daemon {daemon-name} help

管理套接字命令允许你在运行时查看和修改配置，见[查看运行时配置](http://docs.ceph.org.cn/rados/configuration/ceph-conf#ceph-runtime-config)。

另外，你可以在运行时直接修改配置选项（也就是说管理套接字会绕过监视器，不要求你直接登录宿主主机，不像 ceph {daemon-type} tell {id} injectargs 依赖监视器。

# 4. 监控 OSD 和归置组

高可用性和高可靠性要求容错方法来管理软硬件。 Ceph 没有单故障点，并且能在“降级”模式下继续提供服务。其[数据归置](http://docs.ceph.org.cn/rados/operations/data-placement)引进了一个间接层，它可保证数据不会直接绑死到某一个特定 OSD 地址，这也意味着追踪系统错误的根源得深入[归置组](http://docs.ceph.org.cn/rados/operations/placement-groups)及底层的 OSD 。

**Tip**

集群某一部分失效可能导致不能访问某个对象，但不会牵连其他对象。碰到这种问题时无需恐慌，只需按步骤检查 OSD 和归置组，然后排除故障。

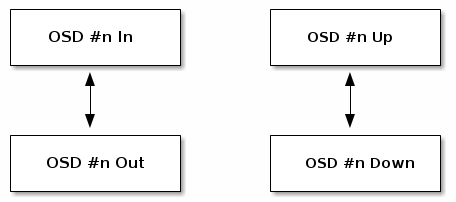
Ceph 通常能自己康复，然而如果故障持续存在，监控 OSD 和归置组有助于找出问题所在。

## 4.1 监控OSD

某 OSD 的状态可以是在集群内（ in ）或集群外（ out ）、也可以是活着且在运行（ up ）或挂了且不在运行（ down ）。如果一个 OSD 活着，它也可以是 in （你可以读写数据）或者 out 集群。如果它以前是 in 但最近 out 了， Ceph 会把其归置组迁移到其他 OSD 。如果一 OSD out 了， CRUSH 就不会再分配归置组给它。如果它挂了（ down ）其状态也应该是 out 。

**Note**

如果一 OSD 状态为 down 且 in ，必定有问题，而且集群处于非健康状态。



**|----------------------------------\*----------------------------------\*----------------------------------\*----------------------------------|**

**编者注**：OSD状态的描述分为两个维度：up或者down（表明OSD是否正常工作），in或者out（表明OSD是否在至少一个PG中）。因此，对于任意一个OSD，共有四种可能的状态：

—— **Up且in**：说明该OSD正常运行，且已经承载至少一个PG的数据。这是一个OSD的标准工作状态；

—— **Up且out**：说明该OSD正常运行，但并未承载任何PG，其中也没有数据。一个新的OSD刚刚被加入Ceph集群后，便会处于这一状态。而一个出现故障的OSD被修复后，重新加入Ceph集群时，也是处于这一状态；

—— **Down且in**：说明该OSD发生异常，但仍然承载着至少一个PG，其中仍然存储着数据。这种状态下的OSD刚刚被发现存在异常，可能仍能恢复正常，也可能会彻底无法工作；

—— **Down且out**：说明该OSD已经彻底发生故障，且已经不再承载任何PG。

**|----------------------------------\*----------------------------------\*----------------------------------\*----------------------------------|**

如果你执行过这些命令，如 ceph health 、 ceph -s 、或 ceph -w ，也许注意到了，集群并非一直返回 HEALTH OK ，别紧张。就 OSD 而言你应该明确，在一些情况下集群不会返回 HEALTH OK ：

1. 你还没启动集群（它不会响应的）。
2. 你刚刚启动或重启完集群，而且它还没准备好，因为归置组正被创建、 OSD 们正在相互建立连接。
3. 你刚刚增加或拆除一个 OSD 。
4. 你刚刚修改完集群运行图。

OSD 监控的一个重要事情是，当集群启动并运行时，所有 OSD 也应该是启动（ up ）并在集群内（ in ）运行。用下列命令查看：

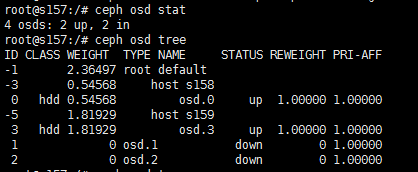
ceph osd stat

其结果会告诉你运行图版本（ eNNNN ）、 总共有 x 个 OSD 、 y 个是 up 的、 z 个是 in 的。

*eNNNN: x osds: y up, z in*



如果处于 in 状态的 OSD 多于 up 的，用下列命令看看哪些 ceph-osd 守护进程没在运行：



**Tip**

精心设计的 CRUSH 分级结构可以帮你更快的定位到物理位置、加快故障排除。

若一个 OSD 状态为 down ，启动它：

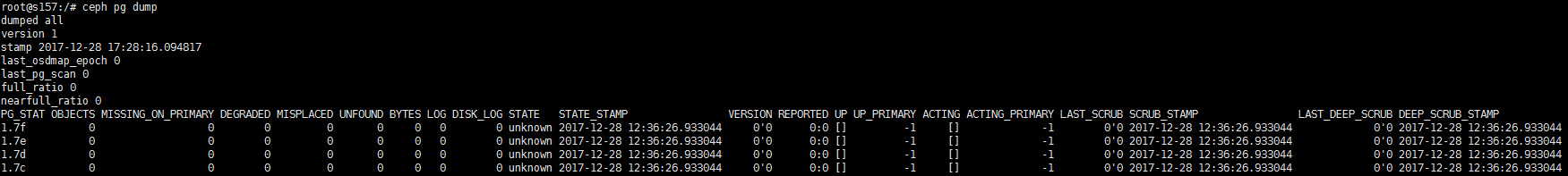
sudo /etc/init.d/ceph -a start osd.1

和 OSD 没运行或不启动相关的问题请看 [OSD 没运行](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-osd#osd-not-running)。

## 4.2 归置组集（PG SETS）

获取归置组列表：

ceph pg dump



要根据指定归置组号查看哪些 OSD 位于 Acting Set 或 Up Set 里，执行：

ceph pg map {pg-num}

其结果会告诉你 osdmap 版本（ eNNN ）、归置组号（ {pg-num} ）、 Up Set 内的 OSD （ up[] ）、和 Acting Set 内的 OSD （ acting[] ）。

osdmap eNNN pg {pg-num} -> up [0,1,2] acting [0,1,2]

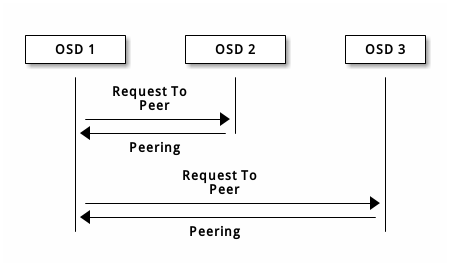
例如：

**Note**

如果 Up Set 和 Acting Set 不一致，这可能表明集群内部在重均衡或者有潜在问题。

## 4.3 节点互联

写入数据前，归置组必须处于 active 、而且**应该**是 clean 状态。假设一存储池的归置组有 3 个副本，为让 Ceph 确定归置组的当前状态，一归置组的主 OSD （即 acting set 内的第一个 OSD ）会与第二和第三 OSD 建立连接、并就归置组的当前状态达成一致意见。



OSD 们也向监视器报告自己的状态，详情见[监视器与 OSD 交互的配置](http://docs.ceph.org.cn/rados/configuration/mon-osd-interaction/)。要排除连接建立问题，参见[互联失败](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-pg#failures-osd-peering)。

## 4.4 监控归置组状态

归置组监控的一件重要事情是保证集群起来并运行着，所有归置组都处于 active 状态、并且最好是 clean 状态。用下列命令查看所有归置组状态：

ceph pg stat

其结果会告诉你归置组运行图的版本号（ vNNNNNN ）、归置组总数 x 、有多少归置组处于某种特定状态，如 active+clean （ y ）。

vNNNNNN: x pgs: y active+clean; z bytes data, aa MB used, bb GB / cc GB avail

**Note**

Ceph 同时报告出多种状态是正常的。

例如：

除了归置组状态之外， Ceph 也会报告数据占据的空间（ aa ）、剩余空间（ bb ）和归置组总容量。这些数字在某些情况下是很重要的：

- 快达到“near full ratio”或“full ratio”时；

- 由于 CRUSH 配置错误致使你的数据没能在集群内分布时。

**归置组唯一标识符**

归置组 ID 由存储池号（不是存储池名字）、后面跟一个点（ . ）、然后是归置组 ID ，它是一个十六进制数字。用 ceph osdlspools 可查看存储池号及其名字，例如，默认存储池 rbd 对应的存储池号是 0 。完整的归置组 ID 格式如下：

{pool-num}.{pg-id}

典型长相：

0.1f

用以下命令获取归置组列表：

ceph pg dump

也可以让它输出到 JSON 格式，并保存到文件：

ceph pg dump -o {filename} --format=json

要查询某个归置组，用下列命令：

ceph pg {poolnum}.{pg-id} query

Ceph 会输出成 JSON 格式。



以下详述了都有哪些常见的状态：

### CREATEING

存储池在建中。

创建存储池时，它会创建指定数量的归置组。 Ceph 在创建一或多个归置组时会显示 creating ；创建完后，在其归置组的 Acting Set 里的 OSD 将建立互联；一旦互联完成，归置组状态应该变为 active+clean ，意思是 Ceph 客户端可以向归置组写入数据了。



### PEERING

互联建立中。

Ceph 为归置组建立互联时，会让存储归置组副本的 OSD 之间就其中的对象和元数据状态达成一致。 Ceph 完成了互联，也就意味着存储着归置组的 OSD 就其当前状态达成了一致。然而，互联过程的完成并不能表明各副本都有了数据的最新版本。

**权威历史**

Ceph **不会**向客户端确认写操作，直到 acting set 里的所有 OSD 都完成了写操作。这样处理保证了从上次成功互联起， acting set 中至少有一个成员确认了每个写操作。

有了各个已确认写操作的精确记录， Ceph 就可以构建和散布一个新的归置组权威历史——一个完整、完全有序的操作集，如果被采用，就能把一个 OSD 的归置组副本更新到最新。

### ACTIVE

Ceph 完成互联后，一归置组状态会变为 active 。 active 状态意味着数据已完好地保存到了主归置组和副本归置组。

### CLEAN

某一归置组处于 clean 状态时，主 OSD 和副本 OSD 已成功互联，并且没有偏离的归置组。 Ceph 已把归置组中的对象复制了规定次数。

### DEGRADED

已降级。

当客户端向主 OSD 写入数据时，由主 OSD 负责把数据副本写入其余副本 OSD 。主 OSD 把对象写入存储后，在副本 OSD 创建完对象副本并报告给主 OSD 之前，主 OSD 会一直停留在 degraded 状态。

归置组状态可以处于 active+degraded 状态，原因在于一 OSD 即使尚未持有所有对象也可以处于 active 状态。如果一 OSD 挂了， Ceph 会把分配到此 OSD 的归置组都标记为 degraded ；那个 OSD 重生后，它们必须重新互联。然而，客户端仍可以向处于 degraded 状态的归置组写入新对象，只要它还在 active 状态。

如果一 OSD 挂了，且老是处于 degraded 状态， Ceph 会把 down 的 OSD 标记为在集群外（ out ）、并把那个 down 掉的 OSD 上的数据重映射到其它 OSD 。从标记为 down 到 out 的时间间隔由 mon osd down out interval 控制，默认是 300 秒。

归置组也会被降级（ degraded ），因为 Ceph 找不到本应存在于此归置组中的一或多个对象，这时，你不能读写找不到的对象，但仍能访问位于降级归置组中的其它对象。

### RECOVERING

Ceph 被设计为可容错，可抵御一定规模的软、硬件问题。当某 OSD 挂了（ down ）时，其内的归置组会落后于别的归置组副本；此 OSD 重生（ up ）时，归置组内容必须更新到当前状态；在此期间， OSD 处于 recovering 状态。

恢复并非总是这些小事，因为一次硬件失败可能牵连多个 OSD 。比如一个机柜或房间的网络交换机失败了，这会导致多个主机上的 OSD 落后于集群的当前状态，故障恢复后每一个 OSD 都必须恢复。

Ceph 提供了几个选项来均衡资源竞争，如新服务请求、恢复数据对象和恢复归置组到当前状态。 osd recovery delay start 选项允许一 OSD 在开始恢复进程前，先重启、重建互联、甚至处理一些重放请求；osd recovery threads 选项限制恢复进程的线程数，默认为 1 线程； osd recovery thread timeout 设置线程超时，因为多个 OSD 可能交替失败、重启和重建互联； osdrecovery max active 选项限制一 OSD 最多同时接受多少请求，以防它压力过大而不能正常服务； osd recovery max chunk 选项限制恢复数据块尺寸，以防网络拥塞。

### BACK FILLING

回填中。

有新 OSD 加入集群时， CRUSH 会把现有集群内的部分归置组重分配给它。强制新 OSD 立即接受重分配的归置组会使之过载，用归置组回填可使这个过程在后台开始。只要回填顺利完成，新 OSD 就可以对外服务了。

在回填运转期间，你可能见到以下几种状态之一： backfill\_wait 表明一回填操作在等待时机，尚未开始； backfill 表明一回填操作正在进行； backfill\_too\_full 表明需要进行回填，但是因存储空间不足而不能完成。某归置组不能回填时，其状态应该是 incomplete 。

Ceph 提供了多个选项来解决重分配归置组给一 OSD （特别是新 OSD ）时相关的负载问题。默认， osd\_max\_backfills 把双向的回填并发量都设置为 10 ； osd backfill full \ ratio 可让一 OSD 在接近占满率（默认 85% ）时拒绝回填请求，如果一 OSD 拒绝了回填请求，在 osd backfill retry interval 间隔之后将重试（默认 10 秒）； OSD 也能用 osd backfill scanmin 和 osd backfill scan max 来管理扫描间隔（默认 64 和 512 ）。

### REMAPPED

被重映射。

负责维护某一归置组的 Acting Set 变更时，数据要从旧集合迁移到新的。新的主 OSD 要花费一些时间才能提供服务，所以老的主 OSD 还要持续提供服务、直到归置组迁移完。数据迁移完后，运行图会包含新 acting set 里的主 OSD 。

### STALE

过时、陈旧。

虽然 Ceph 用心跳来保证主机和守护进程在运行，但是 ceph-osd 仍有可能进入 stuck 状态，它们没有按时报告其状态（如网络瞬断）。默认， OSD 守护进程每半秒（ 0.5 ）会一次报告其归置组、出流量、引导和失败统计状态，此频率高于心跳阀值。如果一归置组的**主 OSD** 所在的 acting set 没能向监视器报告、或者其它监视器已经报告了那个主 OSD 已 down ，监视器们就会把此归置组标记为 stale 。

启动集群时，会经常看到 stale 状态，直到互联完成。集群运行一阵后，如果还能看到有归置组位于 stale 状态，就说明那些归置组的主 OSD 挂了（ down ）、或没在向监视器报告统计信息。

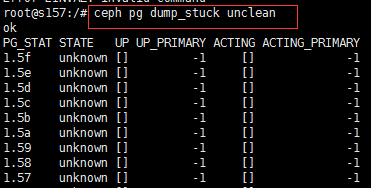
## 4.5 找出故障归置组

如前所述，一个归置组状态不是 active+clean 时未必有问题。一般来说，归置组卡住时 Ceph 的自修复功能往往无能为力，卡住的状态细分为：

* **Unclean**: 归置组里有些对象的副本数未达到期望次数，它们应该在恢复中；
* **Inactive**: 归置组不能处理读写请求，因为它们在等着一个持有最新数据的 OSD 回到 up 状态；
* **Stale**: 归置组们处于一种未知状态，因为存储它们的 OSD 有一阵子没向监视器报告了（由 mon osd report timeout 配置）。

为找出卡住的归置组，执行：

ceph pg dump\_stuck [unclean|inactive|stale|undersized|degraded]



详情见[归置组子系统](http://docs.ceph.org.cn/rados/operations/control#placement-group-subsystem)，关于排除卡住的归置组见[排除归置组错误](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-pg#troubleshooting-pg-errors)。

## 4.6 定位对象

要把对象数据存入 Ceph 对象存储，一 Ceph 客户端必须：

1. 设置对象名
2. 指定一[存储池](http://docs.ceph.org.cn/rados/operations/pools)

Ceph 客户端索取最新集群运行图、并用 CRUSH 算法计算对象到[归置组](http://docs.ceph.org.cn/rados/operations/placement-groups)的映射，然后计算如何动态地把归置组映射到 OSD 。要定位对象，只需要知道对象名和存储池名字，例如：

ceph osd map {poolname} {object-name}

**练习：定位一个对象**

反正是练习，我们先创建一个对象。给 rados put 命令指定一对象名、一个包含数据的测试文件路径、和一个存储池名字，例如：

rados put {object-name} {file-path} --pool=data

rados put test-object-1 testfile.txt --pool=data

用下列命令确认 Ceph 对象存储已经包含此对象：

rados -p data ls

现在可以定位对象了：

ceph osd map {pool-name} {object-name}

ceph osd map data test-object-1

Ceph 应该输出对象的位置，例如：

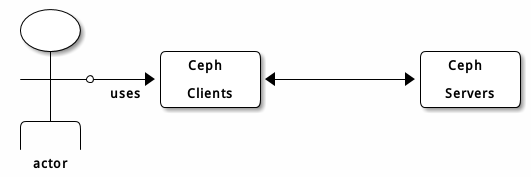
osdmap e537 pool 'data' (0) object 'test-object-1' -> pg 0.d1743484 (0.4) -> up [1,0] acting [1,0]

要删除测试对象，用 rados rm 即可，如：

rados rm test-object-1 --pool=data

# 5. 用户管理（▲中文文档未翻译）

本节叙述了[Ceph客户端](http://docs.ceph.org.cn/glossary/#term-67)的用户身份，及其与[Ceph存储集群](http://docs.ceph.org.cn/glossary/#term-21)的认证和授权。用户可以是个人或系统角色（像应用程序），它们使用Ceph客户端和Ceph服务器守护进程交互。



当Ceph运行身份验证和授权（默认启用）时，您必须指定用户名和包含指定用户的密钥（通常通过命令行）的密钥环。如果您不指定用户名，Ceph将使用client.admin作为默认用户名。如果你没有指定密钥环，Ceph将通过Ceph配置中的密钥环设置来查找密钥环。例如，如果您执行ceph health 命令而不指定用户或密钥环：

ceph health

Ceph会这样解析命令：

ceph -n client.admin --keyring=/etc/ceph/ceph.client.admin.keyring health

## 5.1 背景

不管Ceph客户端的类型（例如块设备，对象存储，文件系统，本地API等），Ceph将所有数据作为对象存储在[池中](http://docs.ceph.org.cn/rados/operations/pools)。Ceph用户必须有权访问池才能读取和写入数据。另外，Ceph用户必须拥有执行权限才能使用Ceph的管理命令。以下概念将帮助您了解Ceph用户管理。

### 5.1.1 用户

用户是个人或系统角色，如应用程序。通过创建用户，您可以控制谁（或什么）可以访问Ceph存储集群，其池以及池中的数据。

Ceph有一种用户类型的概念。为了用户管理的目的，类型将始终是客户端。Ceph以句点（.）分隔的形式标识用户，包括用户类型和用户标识：例如 TYPE.ID，client.admin或client.user1。用户键入的原因是Ceph监视器，OSD和元数据服务器也使用Cephx协议，但它们不是客户端。区分用户类型有助于区分客户端用户和其他用户，从而简化访问控制，用户监控和可追溯性。

有时候Ceph的用户类型可能会让人感到困惑，因为Ceph命令行允许你指定一个有或没有类型的用户，具体取决于你的命令行用法。如果指定--user或--id，则可以省略该类型。所以client.user1可以简单的输入为user1。如果指定 --name或-n，则必须指定类型和名称，例如 client.user1。我们建议尽可能使用类型和名称作为最佳实践。

**注意**

Ceph存储集群用户不同于Ceph对象存储用户或Ceph文件系统用户。Ceph对象网关使用Ceph存储集群用户在网关守护进程和存储集群之间进行通信，但网关对最终用户有自己的用户管理功能。Ceph文件系统使用POSIX语义。与Ceph文件系统相关的用户空间与Ceph存储集群用户不一样。

### 5.1.2 授权（能力）

Ceph用能力（capabilities，caps）这个术语来描述给认证用户的授权，这样才能使用监视器，OSD和元数据服务器的功能。能力也用于限制一个存储池内的数据或某个名字空间的访问。Ceph的管理用户可以在创建或更新某个用户时赋予他能力。

能力的语法符合下面的形式：

{daemon-type} '{cap-spec}[, {cap-spec} ...]'

* **监视器能力**包括r，w，x和allow profile {cap}，例如：

mon 'allow {access-spec}'

mon 'profile *{name}*'

{access-spec} 的语法如下：

\* | all | [r][w][x]

* **OSD能力：** OSD能够包括r，w，x，class-read，class-write和profile osd。另外，OSD能够支持存储池和命名空间的配置。

osd 'allow {access-spec} [{match-spec}]'

osd 'profile *{name}* [pool={pool-name} [namespace={namespace-name}]]'

{access-spec} 的语法如下：

\* | all | [r][w][x] [class-read] [class-write]

class {class name} [{method name}]

{match-spec} 的语法如下：

pool={pool-name} [namespace={namespace-name}] [object\_prefix {prefix}]

[namespace={namespace-name}] tag {application} {key}={value}

* **元数据服务器能力：**对于管理员，请使用allow \*。对于所有其他用户，如CephFS客户端，请参阅[CephFS客户端功能](http://docs.ceph.com/docs/master/cephfs/client-auth/)。

**注意**

Ceph的对象网关守护进程（radosgw）是CEPH存储集群的一种客户端，所以它没被表示成一种独立的CEPH存储集群守护进程类型。

下面描述了各种能力。

**allow**

|  |  |
| --- | --- |
| 描述： | 优先守护进程的访问设置。意味着rw 只有MDS。 |

**r**

|  |  |
| --- | --- |
| 描述： | 给用户读取权限。监视器需要检索CRUSH映射。 |

**w**

|  |  |
| --- | --- |
| 描述： | 给用户写入对象的权限。 |

**x**

|  |  |
| --- | --- |
| 描述： | 给予用户调用类方法（即读和写）和auth 在监视器上进行操作的能力。 |

**class-read**

|  |  |
| --- | --- |
| 描述： | 给用户提供调用类读取方法的能力。x的子集。 |

**class-write**

|  |  |
| --- | --- |
| 描述： | 赋予用户调用类写入方法的能力。x的子集。 |

**\*， all**

|  |  |
| --- | --- |
| 描述： | 为用户提供对特定守护进程/池的读取，写入和执行权限，以及执行管理命令的能力。 |

以下条目描述有效的功能配置文件：

**profile osd （仅限监视器）**

|  |  |
| --- | --- |
| 描述： | 给予用户作为OSD连接到其他OSD或监视器的权限。赋予OSD以使OSD能够处理复制心跳流量和状态报告。 |

**profile mds （仅限监视器）**

|  |  |
| --- | --- |
| 描述： | 授予用户作为MDS连接到其他MDS或监视器的权限。 |

**profile bootstrap-osd （仅限监视器）**

|  |  |
| --- | --- |
| 描述： | 为用户提供启动OSD的权限。赋予部署工具，如ceph-volume，ceph-deploy等，使他们有权自举一个OSD时添加键等。 |

**profile bootstrap-mds （仅限监视器）**

|  |  |
| --- | --- |
| 描述： | 授予用户引导元数据服务器的权限。赋予部署工具，比如ceph-deploy等，以便在引导元数据服务器时有权添加密钥等。 |

**profile rbd （监视器和OSD）**

|  |  |
| --- | --- |
| 描述： | 给予用户操作RBD图像的权限。当用作监视器上限时，它提供了RBD客户端应用程序所需的最小权限。当用作OSD上限时，它提供对RBD客户端应用程序的读写访问。 |

**profile rbd-read-only （仅限OSD）**

|  |  |
| --- | --- |
| 描述： | 为用户提供对RBD图像的只读权限。 |

### 5.1.3 池存储

存储池是用户存储数据的逻辑分区。在Ceph部署中，经常创建存储池作为逻辑分区，用以归类相似的数据。例如，用Ceph作为OpenStack的后端时，典型的部署通常会创建多个存储池，分别用于存储卷宗，映像，备份和虚拟机，以及用户（如client.glance，client.cinder等）。

### 5.1.4 应用标签（APPLICATION TAGS）

根据应用程序元数据的定义，访问可能被限制在特定的池中。\*通配符可以用于key参数， value参数，或两者同时。all是\*的同义词。

### 5.1.5 空间命名（NAMESPACE）

池中的对象可以关联到一个名称空间 - 池中对象的逻辑组。用户对池的访问可以与命名空间相关联，使得用户的读写仅在命名空间内进行。写入池中名称空间的对象只能由有权访问名称空间的用户访问。

**注意**

目前（即luminous），命名空间只对写在librados之上的应用程序有用。Ceph客户端，如块设备，对象存储和文件系统目前不支持这个功能。

命名空间的基本原理是，为了授权单独的用户集合，池可能是计算上昂贵的分离数据集的方法。例如，一个游泳池每个OSD应该有~100个放置组。因此，具有1000个OSD的示例性集群对于一个池将具有100,000个放置组。每个池将在示例性集群中创建另外100,000个放置组。相比之下，将对象写入名称空间只是将名称空间与对象名称关联起来，而不用单独池的计算开销。您可以使用名称空间，而不是为用户或用户组创建单独的池。**注意：此时**只能使用librados。

## 5.2 管理用户

### 5.2.1 罗列用户

列出集群中的用户，请执行以下操作：

ceph auth ls



请注意，用户的TYPE.ID表示法适用于：osd.0是类型为osd的用户，其ID是0，client.admin是类型为client的用户， 其标识是admin（即默认的client.admin用户）。还要注意，每个条目都有一个key：<value>条目和一个或多个 caps：条目。

您可以使用ceph auth 列表中的-o {filename}选项将输出保存到文件中。

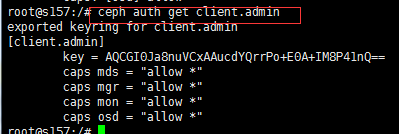
### 5.2.2 获取用户

要检索特定用户，密钥和功能，请执行以下操作：

ceph auth get {TYPE.ID}

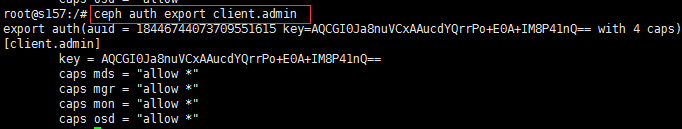
例如：

ceph auth get client.admin



您也可以使用-o {filename}选项与ceph auth get将输出保存到文件中。开发人员也可以执行以下操作：

ceph auth export {TYPE.ID}



该auth export的命令与auth get相同，但会打印出的内部AUID，这是最终的用户无关。

### 5.2.3 添加用户

添加用户将创建一个用户名（即TYPE.ID），一个密钥以及用于创建用户的命令中包含的所有功能。

用户的密钥使用户能够使用Ceph存储集群进行身份验证。用户的权限授权用户在Ceph监视器（mon），Ceph OSD（osd）或Ceph元数据服务器（mds）上读取，写入或执行。

有几种添加用户的方法：

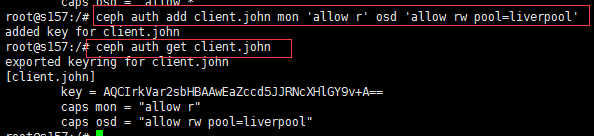
* ceph auth add：此命令是添加用户的标准方式。它将创建用户，生成一个密钥并添加任何指定的功能。
* ceph auth get-or-create：这个命令通常是创建用户最方便的方法，因为它返回一个带有用户名（括号内）和密钥的密钥文件格式。如果用户已经存在，则该命令仅以密钥文件格式返回用户名和密钥。您可以使用该 选项将输出保存到文件。-o{filename}
* ceph auth get-or-create-key：这个命令是创建用户并返回用户密钥（仅）的一种便捷方式。这对于仅需要密钥的客户端是有用的（例如，libvirt）。如果用户已经存在，则该命令只是返回密钥。您可以使用该选项将输出保存到文件。-o{filename}

ceph auth add client.john mon 'allow r' osd 'allow rw pool=liverpool'

ceph auth get-or-create client.paul mon 'allow r' osd 'allow rw pool=liverpool'

ceph auth get-or-create client.george mon 'allow r' osd 'allow rw pool=liverpool' -o george.keyring

ceph auth get-or-create-key client.ringo mon 'allow r' osd 'allow rw pool=liverpool' -o ringo.key



**重要**

如果您向OSD提供用户功能，但不限制访问特定池，则用户将有权访问集群中的所有池！

### 5.2.4 修改用户能力

ceph auth caps 命令可以用来修改指定用户的能力。设置新能力时会覆盖当前能力。查看用户当前的能力可以用 ceph auth getUSERTYPE.USERID ；增加能力时应该加上当前已经有的能力，命令格式如下：

ceph auth caps USERTYPE.USERID {daemon} 'allow [r|w|x|\*|...] [pool={pool-name}] [namespace={namespace-name}]' [{daemon} 'allow [r|w|x|\*|...] [pool={pool-name}] [namespace={namespace-name}]']

例如：

ceph auth get client.john

ceph auth caps client.john mon 'allow r' osd 'allow rw pool=liverpool'

ceph auth caps client.paul mon 'allow rw' osd 'allow rwx pool=liverpool'

ceph auth caps client.brian-manager mon 'allow \*' osd 'allow \*'

要删除功能，您可以重置功能。如果您希望用户无法访问之前设置的特定守护程序，请指定一个空字符串。例如：

ceph auth caps client.ringo mon ' ' osd ' '

关于能力请参考[授权（能力）](http://docs.ceph.org.cn/rados/operations/user-management/#id4)。

### 5.2.5 删除用户

要删除用户，请使用：ceph auth del

ceph auth del {TYPE}.{ID}

其中 {TYPE} 是 client 、 osd 、 mon 或 mds 之一， {ID} 是用户名或守护进程的 ID 。

### 5.2.6 打印用户密钥

要将用户的身份验证密钥输出到标准输出，请执行以下操作：

ceph auth print-key {TYPE}.{ID}

其中 {TYPE} 是 client 、 osd 、 mon 或 mds 之一， {ID} 是用户名或守护进程的 ID 。

当您需要使用用户密钥（例如，libvirt）填充客户端软件时，打印用户密钥很有用。

mount -t ceph serverhost:/ mountpoint -o name=client.user,secret=`ceph auth print-key client.user`

### 5.2.7 导入用户

要导入一个或多个用户，请使用并指定一个密钥环。

ceph auth import -i /path/to/keyring

例如：

sudo ceph auth import -i /etc/ceph/ceph.keyring

**注意**

ceph存储集群将添加新用户的密钥和能力，并将更新现有的用户的密钥和能力。

## 5.2密钥环管理

当您通过Ceph客户端访问Ceph时，Ceph客户端将寻找本地密钥环。Ceph keyring默认预设了以下四个密钥环名称的设置，所以您不必在Ceph配置文件中设置它们，除非您想覆盖默认设置（不推荐）：

* /etc/ceph/$cluster.$name.keyring
* /etc/ceph/$cluster.keyring
* /etc/ceph/keyring
* /etc/ceph/keyring.bin

元$cluster变量是由Ceph配置文件的名称定义的Ceph集群名称（即，ceph.conf表示集群名称是ceph;因此ceph.keyring）。元$name变量是用户类型和用户ID（例如，client.admin因此ceph.client.admin.keyring）。

**注意**

执行读取或写入命令时，/etc/ceph可能需要使用sudo执行命令root。

在创建用户（例如client.ringo）之后，您必须获得密钥并将其添加到Ceph客户机上的密钥环，以便用户可以访问Ceph存储集群。

该[用户管理](http://docs.ceph.com/docs/master/rados/operations/user-management/#user-management)部分详细介绍了如何列出，获取，添加，修改，直接在Ceph的存储集群删除用户。但是，Ceph还提供了 ceph-authtool实用程序，允许您从Ceph客户端管理密钥环。

### 5.2.1 创建密钥环

当您使用[Managing Users](http://docs.ceph.com/docs/master/rados/operations/user-management/#managing-users)部分中的过程来创建用户时，您需要为Ceph客户端提供用户密钥，以便Ceph客户端可以检索指定用户的密钥并使用Ceph存储集群进行身份验证。Ceph客户端访问密钥环以查找用户名和检索用户的密钥。

该ceph-authtool实用程序允许您创建一个钥匙圈。要创建一个空的钥匙环，请使用--create-keyring或-C。例如：

ceph-authtool --create-keyring /path/to/keyring

当创建多个用户的密钥环时，我们建议使用$cluster.keyring密钥环文件名的集群名称（例如），并将其保存在 /etc/ceph目录中，以便keyring配置默认设置将选择文件名，而不需要在本地副本中指定你的Ceph配置文件。例如，ceph.keyring通过执行以下操作来创建：

sudo ceph-authtool -C /etc/ceph/ceph.keyring

使用单个用户创建密钥环时，建议使用集群名称，用户类型和用户名，并将其保存在/etc/ceph目录中。例如，为client.admin用户创建ceph.client.admin.keyring。

要在/etc/ceph中创建一个钥匙圈，你必须用root用户操作。这意味着该文件将只有root用户的rw权限，root当密钥环包含管理员密钥时，这是适当的。但是，如果您打算为特定用户或用户组使用密钥环，请确保执行chown或chmod建立适当的密钥环所有权和访问权限。

### 5.2.2 把用户加入密钥环

当您 [添加用户](http://docs.ceph.com/docs/master/rados/operations/user-management/#add-a-user)到Ceph的存储集群，您可以用[获取用户](http://docs.ceph.com/docs/master/rados/operations/user-management/#get-a-user)程序检索用户，按键和功能，并且用户保存到钥匙圈。

如果您只想为每个密钥环使用一个用户，则使用该选项的“ [获取用户”](http://docs.ceph.com/docs/master/rados/operations/user-management/#get-a-user)过程-o将以密钥环文件格式保存输出。要为client.admin用户创建密钥环，请执行：

sudo ceph auth get client.admin -o /etc/ceph/ceph.client.admin.keyring

请注意，我们为个别用户使用推荐的文件格式。

要将用户导入密钥环，则可以使用ceph-authtool 指定目标密钥环和源密钥环。例如：

sudo ceph-authtool /etc/ceph/ceph.keyring --import-keyring /etc/ceph/ceph.client.admin.keyring

### 5.2.3 创建用户

Ceph提供了[添加用户](http://docs.ceph.com/docs/master/rados/operations/user-management/#add-a-user)功能，直接在Ceph存储集群中创建用户。但是，您也可以直接在Ceph客户端密钥环上创建用户，密钥和功能。然后，您可以将用户导入Ceph存储集群。例如：

sudo ceph-authtool -n client.ringo --cap osd 'allow rwx' --cap mon 'allow rwx' /etc/ceph/ceph.keyring

也可以创建一个密钥环，并同时向密钥环添加一个新用户：

sudo ceph-authtool -C /etc/ceph/ceph.keyring -n client.ringo --cap osd 'allow rwx' --cap mon 'allow rwx' --gen-key

在上述场景下，新用户client.ringo只在密钥环中。要将新用户添加到Ceph存储集群，您仍然必须将新用户添加到Ceph存储集群。

sudo ceph auth add client.ringo -i /etc/ceph/ceph.keyring

### 5.2.4 修改用户属性

要修改密钥环中用户记录的功能，请指定密钥环，然后指定用户以及功能。例如：

sudo ceph-authtool /etc/ceph/ceph.keyring -n client.ringo --cap osd 'allow rwx' --cap mon 'allow rwx'

要将用户更新到Ceph存储集群，您必须将密钥环中的用户更新为Ceph存储集群中的用户条目。

sudo ceph auth import -i /etc/ceph/ceph.keyring

有关从密钥环更新Ceph Storage Cluster用户的详细信息，请参阅[导入用户](http://docs.ceph.com/docs/master/rados/operations/user-management/#import-a-user-s)。

您也可以直接在集群中[修改用户权限](http://docs.ceph.com/docs/master/rados/operations/user-management/#id1)，将结果存储到密钥环文件中; 然后，将密钥环导入到主 ceph.keyring文件中。

## 5.3命令行用法

Ceph 支持用户名和密钥的下列用法：

--id | --user

描述：

Ceph 用一个类型和 ID（ 如 TYPE.ID 或 client.admin 、 client.user1 ）来标识用户， id 、 name 、和 -n 选项可用于指定用户名（如 admin 、 user1 、 foo 等）的 ID 部分，你可以用 --id 指定用户并忽略类型，例如可用下列命令指定 client.foo 用户：

ceph --id foo --keyring /path/to/keyring health

ceph --user foo --keyring /path/to/keyring health

--name | -n

描述：

Ceph 用一个类型和 ID （如 TYPE.ID 或 client.admin 、 client.user1 ）来标识用户， --name 和 -n 选项可用于指定完整的用户名，但必须指定用户类型（一般是 client ）和用户 ID ，例如：

ceph --name client.foo --keyring /path/to/keyring health

ceph -n client.foo --keyring /path/to/keyring health

--keyring

描述：

包含一或多个用户名、密钥的密钥环路径。 --secret 选项提供了相同功能，但它不能用于 RADOS 网关，其 --secret 另有用途。你可以用 ceph auth get-or-create 获取密钥环并保存在本地，然后您就可以改用其他用户而无需重指定密钥环路径了。

sudo rbd map --id foo --keyring /path/to/keyring mypool/myimage

## 5.4 限制

cephx协议让Ceph的客户端和服务器进行彼此身份认证。它不打算处理人类用户或代表他们运行的应用程序的身份验证。如果需要这种效果来处理访问控制需求，则必须具有其他机制，这可能是特定于用于访问Ceph对象存储的前端的机制。这个其他机制的作用是确保只有可以接受的用户和程序能够在Ceph允许访问其对象存储的机器上运行。

用于验证Ceph客户端和服务器的密钥通常以具有适当权限的纯文本文件存储在可信主机中。

**重要**

将密钥存储在纯文本文件中存在安全缺陷，但鉴于Ceph在后台使用的基本身份验证方法，它们难以避免。那些设置Ceph系统的人应该意识到这些缺点。

尤其是，任意的用户机器，尤其是便携式机器，不应该被配置为直接与Ceph交互，因为这种使用模式需要在不安全的机器上存储明文认证密钥。任何偷走那台机器或者偷偷摸摸地进入机器的人都可以获得允许他们向Ceph验证自己的机器的密钥。

不要让潜在的不安全的机器直接访问Ceph对象存储，而应该要求用户使用为您的目的提供足够安全性的方法登录到环境中的受信任机器。该可信计算机将为人类用户存储明文的Ceph密钥。未来版本的Ceph可能会更全面地解决这些特定的身份验证问题。

目前，Ceph身份验证协议没有提供传输中消息的保密性。因此，窃听者可以听到和理解在Ceph中的客户端和服务器之间发送的所有数据，即使它不能创建或修改它们。此外，Ceph不包括加密对象存储中的用户数据的选项。当然，用户可以手动加密并将自己的数据存储在Ceph对象存储中，但Ceph不提供自身执行对象加密的功能。那些在Ceph中存储敏感数据的人应该在将数据提供给Ceph系统之前考虑加密他们的数据。

# 6. 修复PG不一致（▲英文文档只有标题，内容为空）

# 7. 数据归置（DATA PLACEMENT）概览

Ceph 通过 RADOS 集群动态地存储、复制和重新均衡数据对象。很多不同用户因不同目的把对象存储在不同的存储池里，而它们都坐落于无数的 OSD 之上，所以 Ceph 的运营需要些数据归置计划。 Ceph 的数据归置计划概念主要有：

* **存储池（ Pool ）：** Ceph 在存储池内存储数据，它是对象存储的逻辑组；存储池管理着归置组数量、副本数量、和存储池规则集。要往存储池里存数据，用户必须通过认证、且权限合适，存储池可做快照。详情参见8.[存储池](http://docs.ceph.org.cn/rados/operations/pools)。
* **归置组（ Placement Group ）：** Ceph 把对象映射到归置组（ PG ），归置组是一逻辑对象池的片段，这些对象组团后再存入到 OSD 。归置组减少了各对象存入对应 OSD 时的元数据数量，更多的归置组（如每 OSD 100 个）使得均衡更好。详情见11.[归置组](http://docs.ceph.org.cn/rados/operations/placement-groups)。
* **CRUSH 图（ CRUSH Map ）：** CRUSH 是重要组件，它使 Ceph 能伸缩自如而没有性能瓶颈、没有扩展限制、没有单点故障，它为 CRUSH 算法提供集群的物理拓扑，以此确定一个对象的数据及它的副本应该在哪里、怎样跨故障域存储，以提升数据安全。详情见 13.[CRUSH 图](http://docs.ceph.org.cn/rados/operations/crush-map)。

起初安装测试集群的时候，可以使用默认值。但开始规划一个大型 Ceph 集群，做数据归置操作的时候会涉及存储池、归置组、和 CRUSH 。

# 8. 存储池（★英文文档改动）

如果你开始部署集群时没有创建存储池， Ceph 会用默认存储池存数据。存储池提供的功能：

* **自恢复力：** 你可以设置在不丢数据的前提下允许多少 OSD 失效，对多副本存储池来说，此值是一对象应达到的副本数。典型配置存储一个对象和它的一个副本（即 size = 2 ），但你可以更改副本数；对[纠删编码的存储池](http://docs.ceph.org.cn/rados/operations/erasure-code)来说，此值是编码块数（即**纠删码配置**里的 m=2 ）。
* **归置组（PGs）：** 你可以设置一个存储池的归置组数量。典型配置给每个 OSD 分配大约 100 个归置组，这样，不用过多计算资源就能得到较优的均衡。配置了多个存储池时，要考虑到这些存储池和整个集群的归置组数量要合理。
* **CRUSH 规则：** 当你在存储池里存数据的时候，与此存储池相关联的 CRUSH 规则集可控制 CRUSH 算法，并以此操纵集群内对象及其副本的复制（或纠删码编码的存储池里的数据块）。你可以自定义存储池的 CRUSH 规则。
* **快照：** 用 ceph osd pool mksnap 创建快照的时候，实际上创建了某一特定存储池的快照。

要把数据组织到存储池里，你可以列出、创建、删除存储池，也可以查看每个存储池的利用率。

## 8.1 列出池

列出集群的池，执行：

ceph osd lspools



## 8.2 创建一个池

创建存储池前先看看[存储池、归置组和 CRUSH 配置参考](http://docs.ceph.org.cn/rados/configuration/pool-pg-config-ref)。你最好在配置文件里重置默认归置组数量，因为默认值并不理想。关于归置组数量请参考[设置归置组数量](http://docs.ceph.org.cn/rados/operations/placement-groups#set-the-number-of-placement-groups)。

**注意**

从Luminous开始，所有池需要使用池关联到应用程序。有关更多信息，请参阅下面的[关联池到应用](http://docs.ceph.com/docs/master/rados/operations/pools/#id1)

例如：

osd pool default pg num = 100

osd pool default pgp num = 100

创建一个池，执行如下命令：

ceph osd pool create {pool-name} {pg-num} [{pgp-num}] [replicated] \

[crush-rule-name] [expected-num-objects]

ceph osd pool create {pool-name} {pg-num} {pgp-num} erasure \

[erasure-code-profile] [crush-rule-name] [expected\_num\_objects]

参数含义如下：

**{pool-name}**

描述: 存储池名称，必须唯一。

类型: String

是否必需: 必需。

**{pg-num}**

描述: 存储池拥有的归置组总数。关于如何计算合适的数值，请参见归置组。默认值 8 对大多数系统都不合适。

类型: 整数

是否必需: Yes

默认值: 8

**{pgp-num}**

描述: 用于归置的归置组总数。此值应该等于归置组总数，归置组分割的情况下除外。

类型: 整数

是否必需: 没指定的话读取默认值、或 Ceph 配置文件里的值。

默认值: 8

**{replicated|erasure}**

描述: 存储池类型，可以是副本（保存多份对象副本，以便从丢失的 OSD 恢复）或纠删（获得类似 RAID5 的功能）。多副本存储池需更多原始存储空间，但已实现所有 Ceph 操作；纠删存储池所需原始存储空间较少，但目前仅实现了部分 Ceph 操作。

类型: String

是否必需: No.

默认值: replicated

**[crush-rule-name]** （★原本为：crush-ruleset-name）

描述: 此存储池所用的 CRUSH 规则名称。指定的规则必须存在。

类型: String

是否必需: No.

默认值: 对于多副本（ replicated ）存储池来说，其默认规则集由 osd pool default crush replicated ruleset 配置决定，此规则集必须存在。 对于用 erasure-code 编码的纠删码（ erasure ）存储池来说，不同的 {pool-name} 所使用的默认（ default ）纠删码配置是不同的，如果它不存在的话，会显式地创建它。

**[erasure-code-profile=profile]**

描述:仅用于**纠删**存储池。指定[纠删码配置](http://docs.ceph.org.cn/rados/operations/erasure-code-profile)框架，此配置必须已由 osd erasure-code-profile set 定义。

类型: String

是否必需: No.

当你创建存储池时，要设置一个合理的归置组数量（如 100 ）。也要考虑到每 OSD 的归置组总数，因为归置组很耗计算资源，所以很多存储池和很多归置组（如 50 个存储池，各包含 100 归置组）会导致性能下降。收益递减点取决于 OSD 主机的强大。

如何为存储池计算合适的归置组数量请参见[归置组](http://docs.ceph.org.cn/rados/operations/placement-groups)。

**[expected-num-objects]**

描述: 为这个存储池预估的对象数。设置此值（要同时把 filestore merge threshold 设置为负数）后，在创建存储池时就会拆分 PG 文件夹，以免运行时拆分文件夹导致延时增大。

类型: Integer

是否必需: No.

默认值: 0 ，创建存储池时不拆分目录。

## 8.3 关联池到应用程序（★英文文档新增）

在使用之前，池需要与应用程序相关联。将与CephFS一起使用的池或由RGW自动创建的池会自动关联。用于RBD的池应该使用该rbd工具进行初始化（请参阅[阻止设备命令](http://docs.ceph.com/docs/master/rbd/rados-rbd-cmds/#create-a-block-device-pool)了解更多信息）。

对于其他情况，您可以手动将自由格式的应用程序名称关联到池：

ceph osd pool application enable {pool-name} {application-name}

**注意**

CephFS使用应用程序名称cephfs，RBD使用应用程序名称rbd，而RGW使用应用程序名称rgw。

## 8.4 设置存储池配额

存储池配额可设置最大字节数、和/或每存储池最大对象数。

ceph osd pool set-quota {pool-name} [max\_objects {obj-count}] [max\_bytes {bytes}]

例如：

ceph osd pool set-quota data max\_objects 10000

要取消配额，设置为 0 。

## 8.5 删除存储池

要删除一存储池，执行：

ceph osd pool delete {pool-name} [{pool-name} --yes-i-really-really-mean-it]

要除去池，必须在监视器的配置中将mon\_allow\_pool\_delete标志设置为true。否则，将拒绝移除一个池。

请参阅[监视器配置](http://docs.ceph.com/docs/master/rados/configuration/mon-config-ref)以获取更多信息。

如果您为自己创建的池创建了自己的规则，那么当您不再需要池时，应该考虑将其移除：

ceph osd pool get {pool-name} crush\_rule

例如，如果规则是“123”，则可以像这样检查其他池：

ceph osd dump | grep "^pool" | grep "crush\_rule 123"

如果没有其他池使用该自定义规则，那么从集群中删除该规则是安全的。

如果您为不再存在的池创建过具有权限的用户，则应该考虑删除这些用户：

ceph auth ls | grep -C 5 {pool-name}

ceph auth del {user}

## 8.6 重命名存储池

要重命名一个存储池，执行：

ceph osd pool rename {current-pool-name} {new-pool-name}

如果重命名了一个存储池，且认证用户有对每个存储池都有权限，那你必须用新存储池名字更新用户的权限（例如 caps ）。

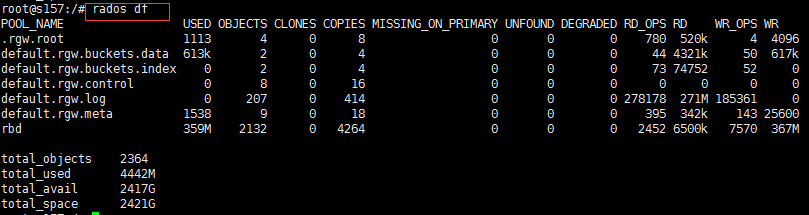
**Note**

适用 0.48 Argonaut 及以上。

## 8.7 查看存储池统计信息

要查看某存储池的使用统计信息，执行命令：

rados df



## 8.8 拍下存储池快照

要拍下某存储池的快照，执行命令：

ceph osd pool mksnap {pool-name} {snap-name}

**Note**

适用 0.48 Argonaut 及以上。

## 8.9 删除存储池快照

要删除某存储池的一个快照，执行命令：

ceph osd pool rmsnap {pool-name} {snap-name}

**Note**

适用 0.48 Argonaut 及以上。

## 8.10 设置存储池选项值

要设置一个存储池的选项值，执行命令：

ceph osd pool set {pool-name} {key} {value}

你可以设置下列key的值：

**compression\_algorithm**（★英文文档新增）

描述： 设置用于底层BlueStore的内联压缩算法。此设置将覆盖[全局设置](http://docs.ceph.com/docs/master/rados/operations/pools/rados/configuration/bluestore-config-ref/#inline-compression)的bluestore compression algorithm。

类型： 串

有效设置： lz4，snappy，zlib，zstd

**compression\_mode**（★英文文档新增）

描述： 设置底层BlueStore的内联压缩算法的策略。此设置将覆盖[全局设置](http://docs.ceph.com/docs/master/rados/operations/pools/rados/configuration/bluestore-config-ref/#inline-compression)的bluestore compression mode。

类型： 串

有效设置： none，passive，aggressive，force

**compression\_min\_blob\_size**（★英文文档新增）

描述： 比这更小的块不会被压缩。此设置将覆盖[全局设置](http://docs.ceph.com/docs/master/rados/operations/pools/rados/configuration/bluestore-config-ref/#inline-compression)的bluestore compression min blob \*。

类型： 无符号整数

**compression\_max\_blob\_size**（★英文文档新增）

描述： 比这个值更大的块在被压缩之前被分成更小的大小为compression\_max\_blob\_size的块。

类型： 无符号整数

**size**

描述： 设置存储池中的对象副本数，详情参见[设置对象副本数](http://docs.ceph.com/docs/master/rados/operations/pools/#set-the-number-of-object-replicas)。仅适用于副本存储池。

类型： 整数

**min\_size**

描述： 设置 I/O 需要的最小副本数，详情参见[设置对象副本数](http://docs.ceph.com/docs/master/rados/operations/pools/#set-the-number-of-object-replicas)。仅适用于副本存储池。

类型： 整数

适用版本： 0.54 及以上。

**pg\_num**（★英文文档新增）

描述： 计算数据归置时要用到的归置组的有效数量。

类型： 整数

有效范围：优于pg\_num当前值。

**pgp\_num**

描述：计算数据归置时要用到的用于放置的归置组的有效数量。

类型： 整数

有效范围： 等于或小于 pg\_num 。

**crush\_rule**（★英文文档改动）

描述： 集群内映射对象归置时使用的规则。

类型： 整数

**allow\_ec\_overwrites**（★英文文档新增）

描述：无论写入纠删码池是否可以更新对象的一部分，cephfs和rbd都可以使用它。有关更多详细信息，请参阅 [重写纠删码](http://docs.ceph.com/docs/master/rados/operations/erasure-code#erasure-coding-with-overwrites)。

类型：布尔

版本： 12.2.0 以上

**hashpspool**

描述： 给指定存储池设置/取消 HASHPSPOOL 标志。

类型： 整数

有效范围： 1 开启， 0 取消

适用版本： 0.48 及以上。

**nodelete**

描述： 给指定存储池设置/取消 NODELETE 标志。

类型： 整数

有效范围： 1 开启， 0 取消

适用版本： Version FIXME

**nopgchange**

描述： 给指定存储池设置/取消 NOPGCHANGE 标志。

类型： 整数

有效范围： 1 开启， 0 取消

适用版本： Version FIXME

**nosizechange**

描述： 给指定存储池设置/取消 NOSIZECHANGE 标志。

类型： 整数

有效范围： 1 开启， 0 取消

适用版本： Version FIXME

**write\_fadvise\_dontneed**

描述： 设置或取消指定存储池的 WRITE\_FADVISE\_DONTNEED 标志。

类型： Integer

有效范围： 1 开启， 0 取消

**noscrub**

描述： 设置或取消指定存储池的 NOSCRUB 标志。

类型： Integer

有效范围： 1 设置， 0 取消

**nodeep-scrub**

描述： 设置或取消指定存储池的 NODEEP\_SCRUB 标志。

类型： Integer

有效范围： 1 开启， 0 取消

**hit\_set\_type**

描述： 启用缓存存储池的命中集跟踪，详情见 Bloom 过滤器。

类型： String

有效值： bloom, explicit\_hash, explicit\_object

默认值： bloom ，其它是用于测试的。

**hit\_set\_count**

描述： 为缓存存储池保留的命中集数量。此值越高， ceph-osd 守护进程消耗的内存越多。

类型： 整数

有效范围： 1. Agent doesn’t handle > 1 yet.

**hit\_set\_period**

描述： 为缓存存储池保留的命中集有效期。此值越高， ceph-osd 消耗的内存越多。

类型： 整数

实例： 3600 1hr

**hit\_set\_fpp**

描述： bloom 命中集类型的假阳性概率。详情见 Bloom 过滤器。

类型： Double

有效范围： 0.0 - 1.0

默认值： 0.05

**cache\_target\_dirty\_ratio**

描述： 缓存存储池包含的脏对象达到多少比例时就把它们回写到后端的存储池。

类型： Double

默认值： .4

**cache\_target\_dirty\_high\_ratio**

描述： 缓存存储池内包含的已修改（脏的）对象达到此比例时，缓存层代理就会更快地把脏对象刷回到后端存储池。

类型： Double

默认值： .6

**cache\_target\_full\_ratio**

描述： 缓存存储池包含的干净对象达到多少比例时，缓存代理就把它们赶出缓存存储池。

类型： Double

默认值： .8

**target\_max\_bytes**

描述： 达到 max\_bytes 阀值时 Ceph 就回写或赶出对象。

类型： 整数

实例： 1000000000000 #1-TB

**target\_max\_objects**

描述： 达到 max\_objects 阀值时 Ceph 就回写或赶出对象。

类型： 整数

实例： 1000000 #1M objects

**hit\_set\_grade\_decay\_rate**

描述： 在两个连续 hit\_sets 间的热度衰退速率。

类型： Integer

有效范围： 0 - 100

默认值： 20

**hit\_set\_grade\_search\_last\_n**

描述： 计算热度时，在 hit\_sets 里最多计数 N 次。

类型： Integer

有效范围： 0 - hit\_set\_count

默认值： 1

**cache\_min\_flush\_age**

描述： 达到此时间（单位为秒）时，缓存代理就把某些对象从缓存存储池刷回到存储池。

类型： 整数

实例： 600 10min

**cache\_min\_evict\_age**

描述： 达到此时间（单位为秒）时，缓存代理就把某些对象从缓存存储池赶出。

类型： 整数

实例： 1800 30min

**fast\_read**

描述： 在纠删码存储池上，如果打开了这个标志，读请求会向所有分片发送子操作读，然后等着，直到收到的分片足以解码给客户端。对 jerasure 和 isa 纠删码插件来说，只要前 K 个请求返回，就能立即解码、并先把这些数据发给客户端。这样有助于资源折衷，以提升性能。当前，这些标志还只能用于纠删码存储池。

类型： Boolean

默认值： 0

**scrub\_min\_interval**

描述： 在负载低时，洗刷存储池的最大间隔秒数。如果是 0 ，就按照配置文件里的 osd\_scrub\_min\_interval 。

类型： Double

默认值： 0

**scrub\_max\_interval**

描述： 不管集群负载如何，都要洗刷存储池的最大间隔秒数。如果是 0 ，就按照配置文件里的 osd\_scrub\_max\_interval 。

类型： Double

默认值： 0

**deep\_scrub\_interval**

描述： “深度”洗刷存储池的间隔秒数。如果是 0 ，就按照配置文件里的 osd\_deep\_scrub\_interval 。

类型： Double

默认值： 0

## 8.11 获取存储池选项值

要获取一个存储池的选项值，执行命令：

ceph osd pool get {pool-name} {key}

你可以获取到下列选项的值：

具体选项的含义见8.10节。

size

min\_size

pg\_num

pgp\_num

crush\_rule

hit\_set\_type

hit\_set\_count

hit\_set\_period

hit\_set\_fpp

cache\_target\_dirty\_ratio

cache\_target\_dirty\_high\_ratio

cache\_target\_full\_ratio

target\_max\_bytes

target\_max\_objects

cache\_min\_flush\_age

cache\_min\_evict\_age

fast\_read

scrub\_min\_interval

scrub\_max\_interval

deep\_scrub\_interval

## 8.12 设置对象副本数

要设置多副本存储池的对象副本数，执行命令：

ceph osd pool set {poolname} size {num-replicas}

**Important：** {num-replicas} 包括对象自身，如果你想要对象自身及其两份拷贝共计三份，指定 3 。例如：

ceph osd pool set data size 3

你可以在每个存储池上执行这个命令。**注意：**一个处于降级模式的对象其副本数小于规定值 pool size ，但仍可接受 I/O 请求。为保证 I/O 正常，可用 min\_size 选项为其设置个最低副本数。例如：

ceph osd pool set data min\_size 2

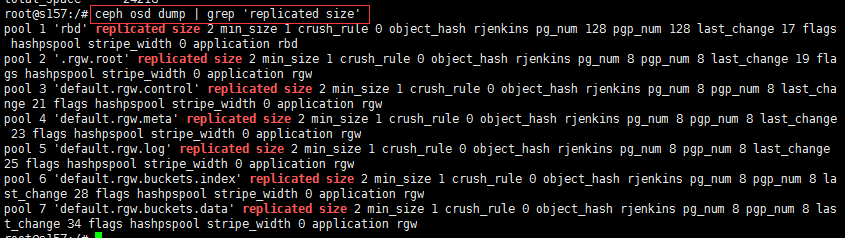
这确保数据存储池里任何副本数小于 min\_size 的对象都不会收到 I/O 了。

## 8.13 获取对象副本数

要获取对象副本数，执行命令：

ceph osd dump | grep 'replicated size'

Ceph 会列出存储池，且高亮 replicated size 属性。默认情况下， Ceph 会创建一对象的两个副本（一共三个副本，或 size 值为 3 ）。



# 9. 纠删码（▲中文文档未翻译）

一个Ceph池被关联到一个类型来维持一个OSD的丢失（即一个磁盘，因为大多数情况下每个磁盘有一个OSD）。创建池时的默认选择将被复制，这意味着每个对象都被复制到多个磁盘上。可以使用“纠删码”池类型来节省空间。

## 9.1 创建示例纠删码存储池

最简单的纠删码池相当于RAID5，并且至少需要三台主机：

$ ceph osd pool create ecpool 12 12 erasure

pool 'ecpool' created

$ echo ABCDEFGHI | rados --pool ecpool put NYAN -

$ rados --pool ecpool get NYAN -

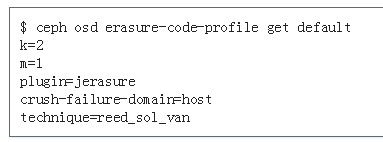
ABCDEFGHI

注意：12代表放置组的数量。

## 9.2 纠错编码配置文件

默认的擦除码配置文件可以维持单个OSD的丢失。它相当于一个大小为2的复制池，但需要1.5TB而不是2TB来存储1TB的数据。默认配置文件可以显示：

$ ceph osd erasure-code-profile get default



选择正确的配置文件非常重要，因为在创建池之后无法修改该配置文件：需要创建具有不同配置文件的新池，并将前一个池中的所有对象都移动到新的池中。

配置文件中最重要的参数是K，M和 crush-failure-domain，因为它们定义了存储开销和数据持久性。例如，如果所需的体系结构必须承受两个机架的损失而存储开销为40％的开销，则可以定义以下简档：

$ ceph osd erasure-code-profile set myprofile \

k=3 \

m=2 \

ruleset-failure-domain=rack

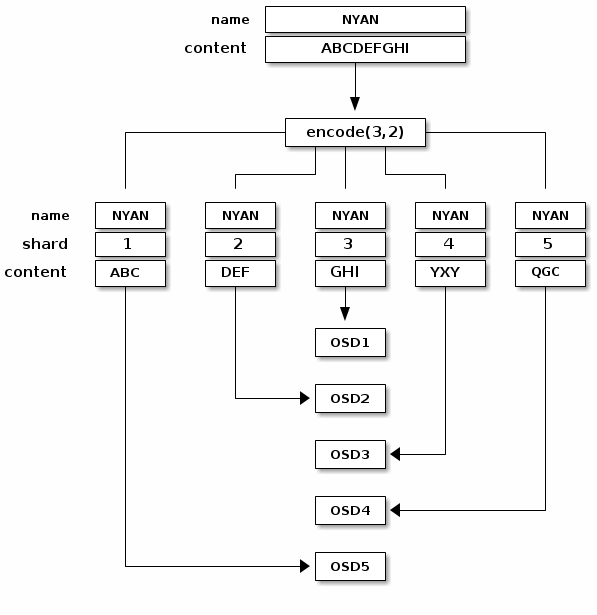
$ ceph osd pool create ecpool 12 12 erasure myprofile

$ echo ABCDEFGHI | rados --pool ecpool put NYAN -

$ rados --pool ecpool get NYAN -

ABCDEFGHI

该NYAN对象将被划分为三个（K = 3），且两个附加的块将被创建（M = 2）。M的值定义了可以同时丢失多少OSD而不丢失任何数据。crush-failure-domain=rack 将创建一个规则：确保没有两个块存储在同一个机架上。



更多信息可以在[纠删码配置](http://docs.ceph.com/docs/master/rados/operations/erasure-code-profile)文件中找到。

## 9.3 重写纠删码（★英文文档新增）

默认情况下，纠删码池只能像RGW那样执行完全对象写入和追加。

Luminous之后，可以使用每个池的设置来启用纠删码池的部分写入。这使得RBD和Cephfs可以将数据存储在纠删码池中：

ceph osd pool set ec\_pool allow\_ec\_overwrites true

这只能在位于bluestore OSD上的池上启用，因为bluestore的校验和用于在深度擦洗期间检测bitrot或其他损坏。除了不安全之外，使用ec覆盖的filestore与bluestore相比性能较差。

纠删码池不支持omap，因此要将它们与RBD和Cephfs一起使用，必须指示它们将数据存储在ec池中，并将其元数据存储在复制池中。对于RBD，这意味着使用纠删码池作为--data-pool创建期间的图像：

rbd create --size 1G --data-pool ec\_pool replicated\_pool/image\_name

对于Cephfs，使用擦除编码池意味着在[文件布局中](http://docs.ceph.com/docs/master/cephfs/file-layouts)设置该池。

## 9.4 纠删码存储池与缓存分级

纠删码池需要比复制池更多的资源，并且缺少一些功能，如omap。为了克服这些限制，可以在纠删码池之前设置一个[缓存层](http://docs.ceph.com/docs/master/rados/operations/cache-tiering)。

例如，如果池的 *hot-storage*由快速存储组成：

$ ceph osd tier add ecpool hot-storage

$ ceph osd tier cache-mode hot-storage writeback

$ ceph osd tier set-overlay ecpool hot-storage

会将*热存储*池作为*ecpool*层以*回写* 模式*放置*，这样每个对*ecpool的*写入和读取实际上都使用 *hot-storage，*并从其灵活性和速度中受益。

更多信息可以在分级[缓存](http://docs.ceph.com/docs/master/rados/operations/cache-tiering)中找到。

## 9.5 术语

*chunk*

当调用编码函数时，它将返回相同大小的块。数据块可以连接起来重建原始对象和编码块，可以用来重建丢失的块。

*K*

数据*块*的数量，即原始对象被划分成的*块*的数量。例如，如果*K* = 2，一个10KB的对象将被分成5KB的*K个*对象。

*M*

编码*块（coding chunks）*的数量，即 由编码函数计算的附加*块*的数量。如果有2个编码*块*，则意味着2个OSD可以丢失而不丢失数据。

## 9.6 内容列表

### 9.6.1 纠删码配置

纠删码由**配置**定义，在创建纠删码存储池及其相关的 CRUSH 规则集时用到。

创建 Ceph 集群时初始化的、名为 **default** 的纠删码配置可提供与两副本相同的冗余水平，却能节省 25% 的磁盘空间。在此配置中 **k=2** 和 **m=1** ，其含义为数据分布于 3 个 OSD （ k+m==3 ）且允许一个失效。

为了在不增加原始存储空间需求的前提下提升冗余性，你可以新建配置。例如，一个 **k=10** 且 **m=4** 的配置可容忍 4 个 OSD 失效，它会把一对象分布到 14 个（ k+m=14 ） OSD 上。此对象先被分割为 **10** 块（若对象为 10MB ，那每块就是 1MB ）、并计算出 **4**个用于恢复的编码块（各编码块尺寸等于数据块，即 1MB ）；这样，原始空间仅多占用 10% 就可容忍 4 个 OSD 同时失效、且不丢失数据。

#### 9.6.1.1 OSD ERASURE-CODE-PROFILE SET（★英文文档改动）

要新建纠删码配置：

ceph osd erasure-code-profile set {name} \

[{directory=directory}] \

[{plugin=plugin}] \

[{stripe\_unit=stripe\_unit}] \

[{key=value} ...] \

[--force]

其中：

**{directory=directory}**

|  |  |
| --- | --- |
| 描述： | 设置纠删码插件的路径，需是目录。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | /usr/lib/ceph/erasure-code |

**{plugin=plugin}**

|  |  |
| --- | --- |
| 描述： | 指定纠删码插件来计算编码块、及恢复丢失块。详见可用插件列表。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | jerasure |

**{stripe\_unit=stripe\_unit}**（★英文文档新增）

|  |  |
| --- | --- |
| 描述： | 数据块中每条数据的数据量。例如，具有2个数据块和stripe\_unit = 4K的配置文件会将块0中的0-4K，块1中的4K-8K以及块0中的8K-12K中的范围再次置入。对于最佳性能，这应该是4K的倍数。创建池时，默认值是从监视器配置的选项osd\_pool\_erasure\_code\_stripe\_unit中获取的。使用此配置文件的池的stripe\_width将是数据块的数量乘以此stripe\_unit。 |
| 类型： | String |
| 是否必需： | No. |

**{key=value}**

|  |  |
| --- | --- |
| 描述： | 纠删码插件所定义的键/值对含义。 |
| 类型： | String |
| 是否必需： | No. |

**--force**

|  |  |
| --- | --- |
| 描述： | 覆盖同名配置。 |
| 类型： | String |
| 是否必需： | No. |

#### 9.6.1.2 OSD ERASURE-CODE-PROFILE RM

要删除纠删码配置：

ceph osd erasure-code-profile rm {name}

若此配置还被某个存储池使用着，则删除会失败。

#### 9.6.1.3 OSD ERASURE-CODE-PROFILE GET

要查看一纠删码配置：

ceph osd erasure-code-profile get {name}

#### 9.6.1.4 OSD ERASURE-CODE-PROFILE LS

列出所有纠删码配置：

ceph osd erasure-code-profile ls

### 9.6.2 JERASURE 纠删码插件

jerasure 插件是最通用、最灵活的插件，也是 Ceph 纠删码存储池的默认插件。

jerasure 插件封装了 [Jerasure](https://bitbucket.org/jimplank/jerasure/) 库，要想更好地理解各参数，建议您仔细阅读 jerasure 文档。

#### 9.6.2.1 创建 JERASURE 配置（★英文文档改动）

要新建 jerasure 纠删码配置：

ceph osd erasure-code-profile set {name} \

plugin=jerasure \

k={data-chunks} \

m={coding-chunks} \

technique={reed\_sol\_van|reed\_sol\_r6\_op|cauchy\_orig|cauchy\_good|liberation|blaum\_roth|liber8tion} \

[crush-root={root}] \

[crush-failure-domain={bucket-type}] \

[crush-device-class={device-class}] \

[directory={directory}] \

[--force]

其中：

**k={data chunks}**

|  |  |
| --- | --- |
| 描述： | 各对象都被分割为数据块，分别存储于不同 OSD 。 |
| 类型： | Integer |
| 是否必需： | Yes. |
| 实例： | 4 |

**m={coding-chunks}**

|  |  |
| --- | --- |
| 描述： | 计算各对象的编码块、并存储于不同 OSD 。编码块的数量等同于在不丢数据的前提下允许同时失效的 OSD 数量。 |
| 类型： | Integer |
| 是否必需： | Yes. |
| 实例： | 2 |

**technique={reed\_sol\_van|reed\_sol\_r6\_op|cauchy\_orig|cauchy\_good|liberation|blaum\_roth|liber8tion}**

|  |  |
| --- | --- |
| 描述： | reed\_sol\_van 技术更灵活：它足以设置 k 和 m 值。 cauchy\_good 技术更快，但你得谨慎地选择 packetsize 值。 reed\_sol\_r6\_op 、 liberation 、 blaum\_roth 、 liber8tion 都是与 RAID6 等价的技术，它们只能配置为 m=2 。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | reed\_sol\_van |

**packetsize={bytes}**

|  |  |
| --- | --- |
| 描述： | 以 bytes 大小的包为单位进行编码。确定合适的包尺寸很难， jerasure 文档对此有很详细的描述。 |
| 类型： | Integer |
| 是否必需： | No. |
| 默认值： | 2048 |

**crush-root={root}**

|  |  |
| --- | --- |
| 描述： | 用于CRUSH规则的第一步的粉碎桶的名称。对于intance 步骤采取默认。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | default |

**crush-failure-domain={bucket-type}**

|  |  |
| --- | --- |
| 描述： | 确保没有两个块位于具有相同故障域的存储桶中。例如，如果故障域是 主机，则不会在同一个主机上存储两个块。它用于创建CRUSH规则步骤，如步骤选择主机。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | host |

**crush-device-class={device-class}**

|  |  |
| --- | --- |
| 描述： | 使用CRUSH贴图中的粉碎设备类名限制放置到特定类（例如， ssd或hdd）的设备。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： |  |

**directory={directory}**

|  |  |
| --- | --- |
| 描述： | 设置纠删码插件的路径，需是目录。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | /usr/lib/ceph/erasure-code |

**--force**

|  |  |
| --- | --- |
| 描述： | 用相同的名称覆盖现有的配置文件。 |
| 类型： | String |
| 是否必需： | No. |

### 9.6.3 ISA 纠删码插件

isa 插件封装了 [ISA](https://01.org/intel%C2%AE-storage-acceleration-library-open-source-version/) 库。它只能运行在 Intel 处理器上。

#### 9.6.3.1 创建 ISA 配置（★英文文档改动）

要新建 isa 纠删码配置：

ceph osd erasure-code-profile set {name} \

plugin=isa \

technique={reed\_sol\_van|cauchy} \

[k={data-chunks}] \

[m={coding-chunks}] \

[crush-root={root}] \

[crush-failure-domain={bucket-type}] \

[crush-device-class={device-class}] \

[directory={directory}] \

[--force]

其中：

**k={data chunks}**

|  |  |
| --- | --- |
| 描述： | 各对象都被分割为数据块，分别存储于不同 OSD 。 |
| 类型： | Integer |
| 是否必需： | No. |
| 默认值： | 7 |

**m={coding-chunks}**

|  |  |
| --- | --- |
| 描述： | 计算各对象的编码块、并存储于不同 OSD 。编码块的数量等同于在不丢数据的前提下允许同时失效的 OSD 数量。 |
| 类型： | Integer |
| 是否必需： | No. |
| 默认值： | 3 |

**technique={reed\_sol\_van|cauchy}**

|  |  |
| --- | --- |
| 描述： | ISA 插件包含两种 Reed Solomon 编码形式。设置为 reed\_sol\_van 表示用 Vandermonde 算法，设置为 cauchy 表示用 Cauchy 算法。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | reed\_sol\_van |

**crush-root={root}**

|  |  |
| --- | --- |
| 描述： | 用于CRUSH规则的第一步的粉碎桶的名称。对于intance 步骤采取默认。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | default |

**crush-failure-domain={bucket-type}**

|  |  |
| --- | --- |
| 描述： | 确保没有两个块位于具有相同故障域的存储桶中。例如，如果故障域是 主机，则不会在同一个主机上存储两个块。它用于创建CRUSH规则步骤，如步骤选择主机。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | host |

**crush-device-class={device-class}**

|  |  |
| --- | --- |
| 描述： | 使用CRUSH贴图中的粉碎设备类名限制放置到特定类（例如， ssd或hdd）的设备。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： |  |

**directory={directory}**

|  |  |
| --- | --- |
| 描述： | 设置纠删码插件的路径，需是目录。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | /usr/lib/ceph/erasure-code |

**--force**

|  |  |
| --- | --- |
| 描述： | 用相同的名称覆盖现有的配置文件。 |
| 类型： | String |
| 是否必需： | No. |

### 9.6.4 局部自修复纠删码插件（★英文文档改动）

用 jerasure 插件时，纠删码编码的对象存储在多个 OSD 上，丢失一个 OSD 的恢复过程需读取所有其他的 OSD 。比如 jerasure 的配置为 k=8 且 m=4 ，丢失一个 OSD 后需读取其他 11 个 OSD 才能恢复。

lrc 纠删码插件创建的是局部校验块，这样只需较少的 OSD 即可恢复。比如 lrc 的配置为 k=8 、 m=4 且 l=4 ，它将为每四个 OSD 创建额外的校验块，当一个 OSD 丢失时，它只需四个 OSD 即可恢复，而不需要十一个。

#### 9.6.4.1 纠删码配置实例

##### 降低主机间的恢复带宽

虽然当所有主机都接入同一交换机时，这不会是诱人的用法，但是带宽利用率确实降低了。

$ ceph osd erasure-code-profile set LRCprofile \

plugin=lrc \

k=4 m=2 l=3 \

crush-failure-domain=host

$ ceph osd pool create lrcpool 12 12 erasure LRCprofile

##### 降低机架间的恢复带宽

在 Firefly 版中，只有主 OSD 与丢失块位于同一机架时所需带宽才能降低。

$ ceph osd erasure-code-profile set LRCprofile \

plugin=lrc \

k=4 m=2 l=3 \

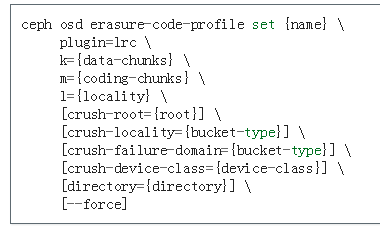
crush-locality=rack \

crush-failure-domain=host

$ ceph osd pool create lrcpool 12 12 erasure LRCprofile

#### 9.6.4.2 创建 LRC 配置

要新建 lrc 纠删码配置：



其中：

**k={data chunks}**

|  |  |
| --- | --- |
| 描述： | 各对象都被分割为数据块，分别存储于不同 OSD 。 |
| 类型： | Integer |
| 是否必需： | Yes. |
| 实例： | 4 |

**m={coding-chunks}**

|  |  |
| --- | --- |
| 描述： | 计算各对象的编码块、并存储于不同 OSD 。编码块的数量等同于在不丢数据的前提下允许同时失效的 OSD 数量。 |
| 类型： | Integer |
| 是否必需： | Yes. |
| 实例： | 2 |

**l={locality}**

|  |  |
| --- | --- |
| 描述： | 把编码块和数据块分组为大小为 locality 的集合。比如， k=4 且 m=2 时，若设置 locality=3 ，将会分组为大小为三的两组，这样各组都能自行恢复，无需从另一组读数据块。 |
| 类型： | Integer |
| 是否必需： | Yes. |
| 实例： | 3 |

**crush-root={root}**

|  |  |
| --- | --- |
| 描述： | 用于CRUSH规则的第一步的粉碎桶的名称。对于intance 步骤采取默认。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | default |

**crush-failure-domain={bucket-type}**

|  |  |
| --- | --- |
| 描述： | 确保没有两个块位于具有相同故障域的存储桶中。例如，如果故障域是 主机，则不会在同一个主机上存储两个块。它用于创建CRUSH规则步骤，如步骤选择主机。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | host |

**crush-device-class={device-class}**

|  |  |
| --- | --- |
| 描述： | 使用CRUSH贴图中的粉碎设备类名限制放置到特定类（例如， ssd或hdd）的设备。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： |  |

**directory={directory}**

|  |  |
| --- | --- |
| 描述： | 设置纠删码插件的路径，需是目录。 |
| 类型： | String |
| 是否必需： | No. |
| 默认值： | /usr/lib/ceph/erasure-code |

**--force**

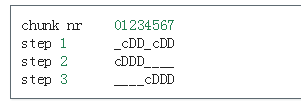
|  |  |
| --- | --- |
| 描述： | 用相同的名称覆盖现有的配置文件。 |
| 类型： | String |
| 是否必需： | No. |

#### 9.6.4.3 低级插件配置

**k** 与 **m** 之和必须是 **l** 参数的整数倍。低级配置参数没有强加这样的限制，并且在某些场合下更方便。因此有可能配置两个组，一组 4 块、另一组 3 块；也有可能递归地定义局部集合，如数据中心和机架再组合为数据中心。 **k/m/l** 可通过生成低级配置来实现。

lrc 纠删码插件递归地使用纠删码技术，这样一些块丢失的恢复大多只需少部分数据块的子集。

比如，三步编码描述为如下：

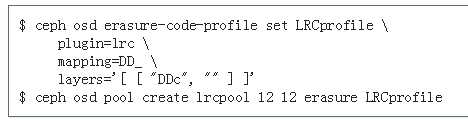


其中， c 是从数据块 D 计算出的编码块，块 7 丢失后能从后四个块恢复，块 2 丢失后能从前四个块恢复。

#### 9.6.4.4 使用低级配置的纠删码配置实例

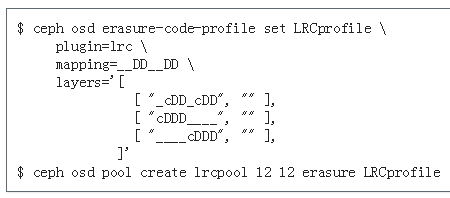
##### 最小测试

此例其实等价于默认纠删码配置， DD 其实就是 K=2 、 c 就是 m=1 并且默认使用 jerasure 插件。



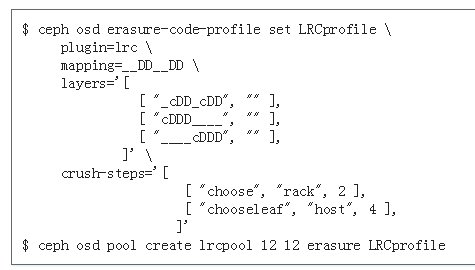
##### 降低主机间的恢复带宽

虽然当所有主机都接入同一交换机时，这不会是诱人的用法，但是带宽利用率确实降低了。它等价于 **k=4** 、 **m=2** 且 **l=3** ，尽管数据块的布局不同：



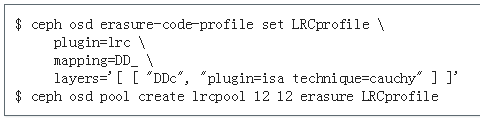
##### 降低机架间的恢复带宽

在 Firefly 版中，只有主 OSD 与丢失块位于同一机架时所需带宽才能降低。

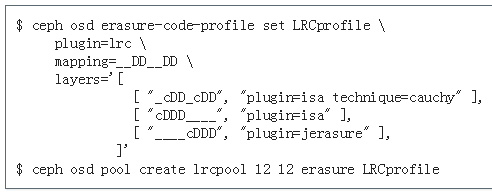


##### 不同纠删码后端测试

LRC 当前用 jerasure 作为默认 EC 后端。使用低级配置时，你可以为每一级分别指定 EC 后端、算法。 layers=’[ [ “DDc”, “” ] ]’ 里的第二个参数其实是用于本级的纠删码配置。下面的例子为 lrcpool 存储池配置了 cauchy 技术的 ISA 后端。

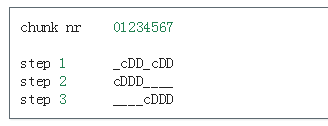


你也可以为各级分别使用不同的纠删码配置。



#### 9.6.4.5 纠删编码和解码算法

在层描述中找出的步骤：

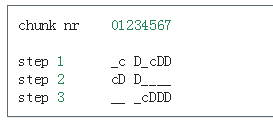


将被依次应用。比如一个 4K 的对象要被编码，它要先通过 **step 1** 被分割为四个 1K 的块（四个大写的 D ），分别依次存储于 2 、 3 、 6 和 7 。这些数据产生了两个编码块（两个小写 c ），它们分别存储于1 和 5 。

step 2 以相似的方式重用 step 1 创建的内容，并把单个编码块 c 存储于位置 0 。最后四个下划线（ \_ ）标记是为提高可读性的，被忽略了。

step 3 把单个编码块存储到了位置 4 ， step 1 创建的三个块被用于计算此编码块，也就是 step 1 产生的编码块成了 step 3 的数据块。

如果 2 块丢失了：



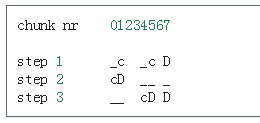
将通过解码来恢复它，反向依次执行： step 3 然后 step 2 最后是 step 1 。

step 3 对 2 一无所知（即它是下划线），所以跳过此步。

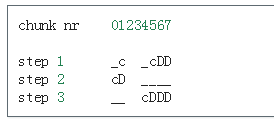
step 2 里的编码块存储在 0 块中，可用来恢复 2 块的内容。没有需要恢复的数据块了，不再考虑 step 1 ，进程终止。

恢复块 2 需读取块 0, 1, 3 并写回块 2 。

如果块 2, 3, 6 丢失：

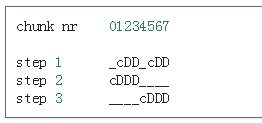


step 3 可恢复块 6 的内容：



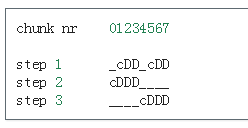
step 2 未能恢复被跳过了，因为丢失了两块（ 2, 3 ），它只能恢复一个块的丢失。

step 1 中的编码块位于块 1, 5 ，因此能恢复块 2, 3 的内容。



#### 9.6.4.6 CRUSH 归置的控制

默认的 CRUSH 规则集会选择位于不同主机的 OSD ，例如：



需要整整 8 个 OSD ，分别存储 8 个块。如果这些主机分别位于相邻的机架，前四块可放到第一个机架，后四块可放到第二个机架，这样丢失单个 OSD 恢复时就不会用到机架间的带宽。

例如：

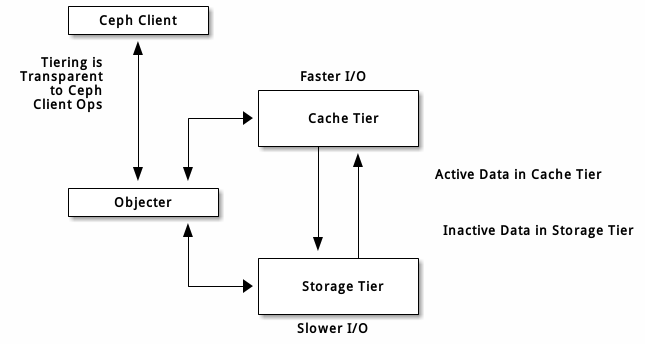


此配置会创建这样的规则集，选定类型为 rack 的两个 crush 桶、并在各桶中再选四个 OSD ，这几个 OSD 分别位于类型为 host 的不同桶中。

此规则集还可以手工雕琢一下，使其更精细。

# 10. 分级缓存

分级缓存可提升后端存储内某些（热点）数据的 I/O 性能。分级缓存需创建一个由高速而昂贵存储设备（如 SSD ）组成的存储池、作为缓存层，以及一个相对低速/廉价设备组成的后端存储池（或纠删码编码的）、作为经济存储层。 Ceph 的对象处理器决定往哪里存储对象，分级代理决定何时把缓存内的对象刷回后端存储层；所以缓存层和后端存储层对 Ceph 客户端来说是完全透明的。



缓存层代理自动处理缓存层和后端存储之间的数据迁移。然而，管理员仍可干预此迁移规则，主要有两种场景：

* **回写模式：** 管理员把缓存层配置为 writeback 模式时， Ceph 客户端们会把数据写入缓存层、并收到缓存层发来的 ACK ；写入缓存层的数据会被迁移到存储层、然后从缓存层刷掉。直观地看，缓存层位于后端存储层的“前面”，当 Ceph 客户端要读取的数据位于存储层时，缓存层代理会把这些数据迁移到缓存层，然后再发往 Ceph 客户端。从此， Ceph 客户端将与缓存层进行 I/O 操作，直到数据不再被读写。此模式对于易变数据来说较理想（如照片/视频编辑、事务数据等）。
* **只读模式：** 管理员把缓存层配置为 readonly 模式时， Ceph 直接把数据写入后端。读取时， Ceph 把相应对象从后端复制到缓存层，根据已定义策略、脏对象会被缓存层踢出。此模式适合不变数据（如社交网络上展示的图片/视频、 DNA 数据、 X-Ray 照片等），因为从缓存层读出的数据可能包含过期数据，即一致性较差。对易变数据不要用 readonly 模式。

正因为所有 Ceph 客户端都能用缓存层，所以才有提升块设备、 Ceph 对象存储、 Ceph 文件系统和原生绑定的 I/O 性能的潜力。

## 10.1 一个忠告（★英文文档新增）

高速缓存分层会降低大多数工作负载的性能。使用此功能之前，用户应该非常小心。

* 依赖于工作负载：缓存是否会提高性能高度依赖于工作负载。因为将对象移入或移出缓存相关的代价是相关的，所以只有在数据集中的访问模式存在较大的偏差时才能生效，这样大多数请求就会触及少量对象。缓存池应足够大以捕获工作负载的工作集以避免发生颠簸。
* 很难进行基准测试：用户运行以测量性能的大多数基准测试将显示使用高速缓存分层的糟糕表现，部分原因是由于极少数基准将请求偏向一小组对象，高速缓存需要很长时间才能“预热” “而且因为热身成本可能会很高。
* 通常较慢：对于不是缓存分层的工作负载，如果没有启用缓存分层，性能往往比普通的RADOS池慢。
* librados对象枚举：librados级别的对象枚举API并不意味着在案件的情况下是连贯的。如果您的应用程序直接使用librados并依赖于对象枚举，则缓存分层可能无法按预期工作。（这对于RGW，RBD或CephFS不是问题。）
* 复杂性：启用缓存分层意味着正在使用RADOS集群内的许多其他机制和复杂性。这会增加您在系统中遇到其他用户尚未遇到的错误的可能性，并会使您的部署处于更高级别的风险中。

### 10.1.1 已知良好的工作负载

* RGW时间倾斜：如果RGW工作负载是几乎所有读取操作都针对最近写入的对象，则可以使用一个简单的缓存分层配置，将最近写入的对象从可配置的时间段下载到缓存到基层。

### 10.1.2 已知的不良工作负载

已知以下配置与缓存分层效果不佳。

* 具有复制高速缓存和擦除编码基础的RBD：这是一个常见的请求，但通常效果不佳。即使合理偏差的工作负载仍然会向冷对象发送一些小的写操作，并且由于擦除编码池尚不支持小写操作，因此必须将整个（通常是4 MB）对象迁移到高速缓存中以满足小型（通常为4 KB）写。只有少数用户已经成功部署了这个配置，并且只适用于他们，因为他们的数据非常冷（备份）并且对性能没有任何敏感性。
* 具有复制的缓存和基础的 RBD：具有复制的基础层的RBD比基础被擦除编码时更好，但是仍然高度依赖于工作负载中的偏斜量，并且非常难以验证。用户需要很好地理解他们的工作负载，并且需要仔细调整缓存分层参数。

## 10.2 配置存储池

要设置缓存层，你必须有两个存储池。一个作为后端存储、另一个作为缓存。

### 10.2.1 配置后端存储池

设置后端存储池通常会遇到两种场景：

* **标准存储：** 此时，Ceph存储集群内的存储池保存了一对象的多个副本；
* **纠删存储池：** 此时，存储池用纠删码高效地存储数据，性能稍有损失。

在标准存储场景中，你可以用 CRUSH 规则集来标识失败域（如 osd 、主机、机箱、机架、排等）。当规则集所涉及的所有驱动器规格、速度（转速和吞吐量）和类型相同时， OSD 守护进程运行得最优。创建规则集的详情见 [CRUSH 图](http://docs.ceph.org.cn/rados/operations/crush-map)。创建好规则集后，再创建后端存储池。

在纠删码编码情景中，创建存储池时指定好参数就会自动生成合适的规则集，详情见[创建存储池](http://docs.ceph.org.cn/rados/operations/pools#create-a-pool)。

在后续例子中，我们把 cold-storage 当作后端存储池。

### 10.2.2 配置缓存池

缓存存储池的设置步骤大致与标准存储情景相同，但仍有不同：缓存层所用的驱动器通常都是高性能的、且安装在专用服务器上、有自己的规则集。制定规则集时，要考虑到装有高性能驱动器的主机、并忽略没有的主机。详情见[给存储池指定 OSD](http://docs.ceph.org.cn/rados/operations/crush-map/#placing-different-pools-on-different-osds) 。

在后续例子中， hot-storage 作为缓存存储池、 cold-storage 作为后端存储池。

关于缓存层的配置及其默认值的详细解释请参考[存储池——调整存储池](http://docs.ceph.org.cn/rados/operations/pools#set-pool-values)。

## 10.3 创建缓存层

设置一缓存层需把缓存存储池挂接到后端存储池上：

ceph osd tier add {storagepool} {cachepool}

例如：

ceph osd tier add cold-storage hot-storage

用下列命令设置缓存模式：

ceph osd tier cache-mode {cachepool} {cache-mode}

例如：

ceph osd tier cache-mode hot-storage writeback

缓存层盖在后端存储层之上，所以要多一步：必须把所有客户端流量从存储池迁移到缓存存储池。用此命令把客户端流量指向缓存存储池：

ceph osd tier set-overlay {storagepool} {cachepool}

例如：

ceph osd tier set-overlay cold-storage hot-storage

## 10.4 配置缓存层

缓存层支持几个配置选项，可按下列语法配置：

ceph osd pool set {cachepool} {key} {value}

详情见[存储池——调整存储池](http://docs.ceph.org.cn/rados/operations/pools#set-pool-values)。

### 10.4.1 目标尺寸和类型

生产环境下，缓存层的 hit\_set\_type 还只能用 Bloom 过滤器：

ceph osd pool set {cachepool} hit\_set\_type bloom

例如：

ceph osd pool set hot-storage hit\_set\_type bloom

hit\_set\_count 和 hit\_set\_period 选项分别定义了 HitSet 覆盖的时间区间、以及保留多少个这样的 HitSet 。

ceph osd pool set {cachepool} hit\_set\_count 1

ceph osd pool set {cachepool} hit\_set\_period 3600

ceph osd pool set {cachepool} target\_max\_bytes 1000000000000

**注意**

hit\_set\_count越大，ceph-osd进程消耗的RAM越多。

保留一段时间以来的访问记录，这样 Ceph 就能判断一客户端在一段时间内访问了某对象一次、还是多次（存活期与热度）。

min\_read\_recency\_for\_promote 定义了在处理一个对象的读操作时检查多少个 HitSet ，检查结果将用于决定是否异步地提升对象。它的取值应该在 0 和 hit\_set\_count 之间，如果设置为 0 ，对象会一直被提升；如果设置为 1 ，就只检查当前 HitSet ，如果此对象在当前 HitSet 里就提升它，否则就不提升；设置为其它值时，就要挨个检查此数量的历史 HitSet ，如果此对象出现在 min\_read\_recency\_for\_promote 个 HitSet 里的任意一个，那就提升它。

还有一个相似的参数用于配置写操作，它是 min\_write\_recency\_for\_promote 。

ceph osd pool set {cachepool} min\_read\_recency\_for\_promote 1

ceph osd pool set {cachepool} min\_write\_recency\_for\_promote 1

**Note**

统计时间越长、 min\_read\_recency\_for\_promote 或 min\_write\_recency\_for\_promote 取值越高， ceph-osd 进程消耗的内存就越多，特别是代理正忙着刷回或赶出对象时，此时所有 hit\_set\_count 个 HitSet 都要载入内存。

### 10.4.2 缓存空间消长

缓存分层代理有两个主要功能：

* **刷回：** 代理找出修改过（或脏）的对象、并把它们转发给存储池做长期存储。
* **赶出：** 代理找出未修改（或干净）的对象、并把最近未用过的赶出缓存。

#### 绝对空间消长

缓存分层代理可根据总字节数或对象数量来刷回或赶出对象，用下列命令可指定最大字节数：

ceph osd pool set {cachepool} target\_max\_bytes {#bytes}

例如，用下列命令配置在达到 1TB 时刷回或赶出：

ceph osd pool set hot-storage target\_max\_bytes 1099511627776

用下列命令指定缓存对象的最大数量：

ceph osd pool set {cachepool} target\_max\_objects {#objects}

例如，用下列命令配置对象数量达到 1M 时开始刷回或赶出：

ceph osd pool set hot-storage target\_max\_objects 1000000

**注意**

Ceph无法自动确定缓存池的大小，因此在这里需要绝对大小的配置，否则flush / evict将不起作用。如果两个都配置了，缓存分层代理会按先到的阀值执行刷回或赶出。

**注意**

所有的客户请求将被阻止，只有当 target\_max\_bytes或 target\_max\_objects达成

#### 相对空间消长

缓存分层代理可根据缓存存储池相对大小刷回或赶出对象。当缓存池包含的已修改（或脏）对象达到一定比例时，缓存分层代理就把它们刷回到存储池。用下列命令设置 cache\_target\_dirty\_ratio ：

ceph osd pool set {cachepool} cache\_target\_dirty\_ratio {0.0..1.0}

例如，设置为 0.4 时，脏对象达到缓存池容量的 40% 就开始刷回：

ceph osd pool set hot-storage cache\_target\_dirty\_ratio 0.4

当脏对象达到其容量的一定比例时，要更快地刷回脏对象。用下列命令设置 cache\_target\_dirty\_high\_ratio:

ceph osd pool set {cachepool} cache\_target\_dirty\_high\_ratio {0.0..1.0}

例如，设置为 0.6 表示：脏对象达到缓存存储池容量的 60% 时，将开始更激进地刷回脏对象。显然，其值最好在 dirty\_ratio 和 full\_ratio 之间：

ceph osd pool set hot-storage cache\_target\_dirty\_high\_ratio 0.6

当缓存池利用率达到总容量的一定比例时，缓存分层代理会赶出部分对象以维持空闲空间。执行此命令设置 cache\_target\_full\_ratio ：

ceph osd pool set {cachepool} cache\_target\_full\_ratio {0.0..1.0}

例如，设置为 0.8 时，干净对象占到总容量的 80% 就开始赶出缓存池：

ceph osd pool set hot-storage cache\_target\_full\_ratio 0.8

### 10.4.3 缓存时长

你可以规定缓存层代理必须延迟多久才能把某个已修改（脏）对象刷回后端存储池：

ceph osd pool set {cachepool} cache\_min\_flush\_age {#seconds}

例如，让已修改（或脏）对象需至少延迟 10 分钟才能刷回，执行此命令：

ceph osd pool set hot-storage cache\_min\_flush\_age 600

你可以指定某对象在缓存层至少放置多长时间才能被赶出：

ceph osd pool {cache-tier} cache\_min\_evict\_age {#seconds}

例如，要规定 30 分钟后才赶出对象，执行此命令：

ceph osd pool set hot-storage cache\_min\_evict\_age 1800

## 10.5 拆除缓存层

回写缓存和只读缓存的去除过程不太一样。

### 10.5.1 拆除只读缓存

只读缓存不含变更数据，所以禁用它不会导致任何近期更改的数据丢失。

1、把缓存模式改为 none 即可禁用。

ceph osd tier cache-mode {cachepool} none

例如：

ceph osd tier cache-mode hot-storage none

2、去除后端存储池的缓存池。

ceph osd tier remove {storagepool} {cachepool}

例如：

ceph osd tier remove cold-storage hot-storage

### 10.5.2 拆除回写缓存

回写缓存可能含有更改过的数据，所以在禁用并去除前，必须采取些手段以免丢失缓存内近期更改的对象。

1、把缓存模式改为 forward ，这样新的和更改过的对象将直接刷回到后端存储池。

ceph osd tier cache-mode {cachepool} forward

例如：

ceph osd tier cache-mode hot-storage forward

2、确保缓存池已刷回，可能要等数分钟：

rados -p {cachepool} ls

3、如果缓存池还有对象，你可以手动刷回，例如：

rados -p {cachepool} cache-flush-evict-all

4、去除此盖子，这样客户端就不会被指到缓存了。

ceph osd tier remove-overlay {storagetier}

例如：

ceph osd tier remove-overlay cold-storage

5、最后，从后端存储池剥离缓存层存储池。

ceph osd tier remove {storagepool} {cachepool}

例如：

ceph osd tier remove cold-storage hot-storage

# 11. 归置组（pg）（★英文文档改动）

## 11.1 预定义 PG\_NUM

用此命令创建存储池时：

ceph osd pool create {pool-name} pg\_num

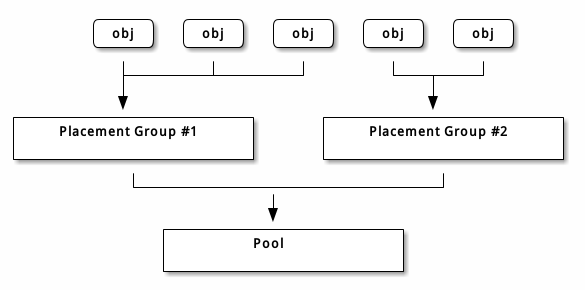
确定 pg\_num 取值是强制性的，因为不能自动计算。下面是几个常用的值：

* 少于 5 个 OSD 时可把 pg\_num 设置为 128
* OSD 数量在 5 到 10 个时，可把 pg\_num 设置为 512
* OSD 数量在 10 到 50 个时，可把 pg\_num 设置为 1024（★原本为4096）
* OSD 数量大于 50 时，你得理解权衡方法、以及如何自己计算 pg\_num 取值
* 自己计算 pg\_num 取值时可借助 [pgcalc](http://ceph.com/pgcalc/) 工具

随着 OSD 数量的增加，正确的 pg\_num 取值变得更加重要，因为它显著地影响着集群的行为、以及出错时的数据持久性（即灾难性事件导致数据丢失的概率）。

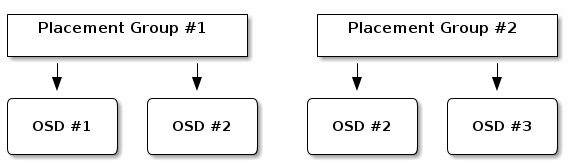
## 11.2 PG是如何使用的？（▲中文文档未翻译）

存储池内的归置组（ PG ）把对象汇聚在一起，因为跟踪每一个对象的位置及其元数据需要大量计算——即一个拥有数百万对象的系统，不可能在对象这一级追踪位置。



Ceph 客户端会计算某一对象应该位于哪个归置组里，它是这样实现的，先给对象 ID 做哈希操作，然后再根据指定存储池里的 PG 数量、存储池 ID 做一个运算。详情见 [PG 映射到 OSD](http://docs.ceph.org.cn/architecture#mapping-pgs-to-osds) 。

PG组中的对象内容存储在一组OSD中。例如，在大小为2的副本池中，每个PG组将在两个OSD上存储对象，如下所示。



如果OSD＃2失败，另一个将被分配到放置组＃1，并且将填充OSD＃1中的所有对象的副本。如果池大小从两个更改为三个，则会为PG分配一个额外的OSD，并将接收PG中的所有对象的副本。

PG不拥有OSD，他们与同一个池（甚至是其他池）中的其他PG共享OSD。如果OSD＃2失败，则PG＃2也将必须使用OSD＃3来恢复对象的副本。

当PG数量增加时，新的PG将被分配OSD。CRUSH函数的结果也将发生变化，以前的PG中的一些对象将被复制到新的PG中，并从旧PG中移除。

## 11.3 PG的权衡（▲中文文档未翻译）

数据持久性和所有OSD中的均匀分布需要更多的PG，但是为了节省CPU和内存，它们的数量应该减少到最小。

### 11.3.1 数据持久性

OSD失败后，数据丢失的风险会增加，直到其中包含的数据完全恢复。我们来想象一下，在单个PG中导致永久数据丢失的情况：

* OSD失败，包含的对象的所有副本都将丢失。对于PG中的所有对象，副本的数量突然从三个减少到两个。
* 通过选择一个新的OSD重新创建所有对象的第三个副本，Ceph开始恢复这个PG。
* 同一个PG中的另一个OSD在新OSD完全填充第三个副本之前失败。有些对象将只有一个幸存的副本。
* Ceph挑选另一个OSD，并保持复制对象以恢复所需数量的副本。
* 在恢复完成之前，同一PG中的第三个OSD失败。如果这个OSD只包含一个对象的剩余副本，它将永久丢失。

在包含10个OSD的集群中，在三个副本池中具有512个PG，CRUSH将为每个PG分配三个OSD。最后，每个OSD最终将托管（512 \* 3）/ 10 = ~150个PG。当第一个OSD失败时，上述场景将因此同时开始恢复所有150个PG。

正在恢复的150个安置小组可能会均匀分布在剩余的9个OSD上。因此，剩下的每个OSD都可能发送对象的副本给所有其他的对象，并且因为它们成为新的PG的一部分而接收一些新的要存储的对象。

完成此恢复所需的时间完全取决于Ceph集群的体系结构。假设每个OSD由一台1TB SSD托管在一台机器上，并且所有这些都连接到10Gb / s交换机，单个OSD的恢复在M分钟内完成。如果每台机器使用不带SSD日志和1Gb / s交换机的spinners的两个OSD，它至少会慢一个数量级。

在这种大小的集群中，PG的数量几乎不会影响数据的持久性。它可能是128或8192，恢复不会更慢或更快。

但是，将相同的Ceph集群扩展到20个OSD而不是10个OSD可能会加速恢复，从而显着提高数据持久性。当仅有10个OSD时，每个OSD现在仅参与〜75个PG而不是〜150，并且仍然要求所有19个剩余的OSD执行相同数量的对象副本才能恢复。但是，如果有10个OSD需要复制约100GB，他们现在必须每个复制50GB。如果网络是瓶颈，恢复将会发生两次。换句话说，当OSD的数量增加时，恢复速度会更快。

如果这个集群增长到40个OSD，他们每个只会托管~35个PG。如果一个OSD死亡，恢复将继续加快，除非被另一个瓶颈阻塞。但是，如果此集群增长到200个OSD，则每个群组将仅托管〜7个PG。如果一个OSD死亡，这些PG中的大部分〜21（7 \* 3）个OSD之间将会发生恢复：恢复需要比有40个OSD更长的时间，这意味着应该增加PG的数量。

不论恢复时间有多短，在此期间都可能有第二个 OSD 失败。在前述的有 10 个 OSD 的集群中，不管哪个失败了，都有大约 17 个归置组（即需恢复的大约 150 / 9 个归置组）将只有一份可用副本；并且假设剩余的 8 个 OSD 中任意一个失败，两个归置组中最后的对象都有可能丢失（即正在恢复的、大约 17 / 8 个仅剩一个副本的归置组）。

当集群大小变为 20 个 OSD 时， 3 个 OSD 丢失导致的归置组损坏会降低。第二个 OSD 丢失会降级大约 4 个（即需恢复的归置组约为 75 / 19 ）而不是约 17 个归置组，并且只有当第三个 OSD 恰好是包含可用副本的四分之一个 OSD 时、才会丢失数据。换句话说，假设在恢复期间丢失一个 OSD 的概率是 0.0001% ，那么，在包含 10 个 OSD 的集群中丢失 OSD 的概率是 17 \* 10 \* 0.0001% ，而在 20 个 OSD 的集群中将是 4 \* 20 \* 0.0001% 。

简而言之，更多OSD意味着更快的恢复和更低的级联失败风险，导致安置小组的永久性损失。就数据持久性而言，拥有512或4096个PG的群组大致相当于少于50个OSD的集群。

注意：添加到集群的新OSD可能需要很长时间才能填充分配给它的PG。但是，任何对象都不会降级，并且不会影响集群中包含的数据的持久性。

### 11.3.2 对象在一个池中的分配

理想情况下，对象均匀分布在每个PG中。由于CRUSH计算每个对象的PG，但实际上并不知道这个PG内的每个OSD中存储了多少数据，PG的数量与OSD的数量之间的比率可能会显着影响数据的分配。

例如，如果在三个副本池中有十个OSD的单个PG，则只有三个OSD将被使用，因为CRUSH将没有其他选择。当有更多的PG可用时，对象更可能均匀分布在其中。CRUSH还尽一切努力在所有现有的安置小组中均匀分布OSD。

只要有一个或两个数量级的PG比OSD更多，分配应该是均匀的。例如，3个OSD的300个PG，1000个OSD的1000个PG等等。

数据分布的不均衡可能是由OSD和PG之间的比率以外的因素造成的。由于CRUSH没有考虑到对象的大小，一些非常大的对象可能会造成不平衡。假设总计4GB的一百万个4K对象在10个OSD上的1000个PG之间均匀分布。他们将在每个OSD上使用4GB / 10 = 400MB。如果一个400MB的对象被添加到池中，则支持放置对象的PG的三个OSD将被填充400MB + 400MB = 800MB，而另外七个将被占用400MB。

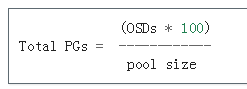
### 11.3.3 内存，CPU和网络使用情况

各个归置组、 OSD 和监视器都一直需要内存、网络、处理器，在恢复期间需求更大。为消除过载而把对象聚集成簇是归置组存在的主要原因。

最小化归置组数量可节省不少资源。

## 11.4 选择PG的数量（▲中文文档未翻译）

如果您有超过50个OSD，我们建议每个OSD大约50-100个PG，以平衡资源使用，数据耐用性和分配。如果OSD少于50个，则在上述[预选](http://docs.ceph.com/docs/master/rados/operations/placement-groups/#preselection)中进行[选择](http://docs.ceph.com/docs/master/rados/operations/placement-groups/#preselection)是最好的。对于单个对象池，可以使用以下公式来获取基准：

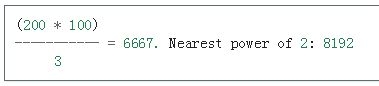


其中**池大小**是复制池的副本数或纠删码池的K + M总和（由**ceph osd erasure-code-profile get返回**）。

然后，您应该检查一下设计Ceph集群的方式是否有意义，以最大化[数据持久性](http://docs.ceph.com/docs/master/rados/operations/placement-groups/#data-durability)， [对象分布](http://docs.ceph.com/docs/master/rados/operations/placement-groups/#object-distribution)和最小化[资源使用](http://docs.ceph.com/docs/master/rados/operations/placement-groups/#resource-usage)。

其结果汇总后应该接近 2 的幂。汇总并非强制的，如果你想确保所有归置组内的对象数大致相等，最好检查下。

比如，一个配置了 200 个 OSD 且副本数为 3 的集群，你可以这样估算归置组数量：



当使用多个数据池存储对象时，您需要确保您平衡每个池的PG数量与每个OSD的PG数量，以便达到合理的PG总数，并因此使得每个 OSD 无需耗费过多系统资源或拖慢连接进程就能实现较小变迁。

例如，在10个OSD上具有512个PG的10个池的集群总共是5个120个PG，其分布在10个OSD上，即每个OSD 512个PG。这不会使用太多的资源。但是，如果1,000个池分别由512个PG创建，那么OSD将处理大约50,000个PG，每个对象需要更多的资源和时间进行对等。

您可能会发现[PGCalc](http://ceph.com/pgcalc/)工具很有帮助。

## 11.1 设置归置组数量

要设置某存储池的归置组数量，你必须在创建它时就指定好，详情见[创建存储池](http://docs.ceph.org.cn/rados/operations/pools#createpool)。一存储池的归置组数量设置好之后，还可以增加（但不可以减少），下列命令可增加归置组数量：

ceph osd pool set {pool-name} pg\_num {pg\_num}

你增加归置组数量后、还必须增加用于归置的归置组（ pgp\_num ）数量，这样才会开始重均衡。 pgp\_num 数值才是 CRUSH 算法采用的用于归置的归置组数量。虽然 pg\_num 的增加引起了归置组的分割，但是只有当用于归置的归置组（即 pgp\_num ）增加以后，数据才会被迁移到新归置组里。 pgp\_num 的数值应等于 pg\_num 。可用下列命令增加用于归置的归置组数量：

ceph osd pool set {pool-name} pgp\_num {pgp\_num}

## 11.2 获取归置组数量

要获取一个存储池的归置组数量，执行命令：

ceph osd pool get {pool-name} pg\_num

## 11.3 获取归置组统计信息

要获取集群里归置组的统计信息，执行命令：

ceph pg dump [--format {format}]

可用格式有纯文本 plain （默认）和 json 。

## 11.4 获取卡住的归置组统计信息

要获取所有卡在某状态的归置组统计信息，执行命令：

ceph pg dump\_stuck inactive|unclean|stale|undersized|degraded [--format <format>] [-t|--threshold <seconds>]

**Inactive** （不活跃）归置组不能处理读写，因为它们在等待一个有最新数据的 OSD 复活且进入集群。

**Unclean** （不干净）归置组含有复制数未达到期望数量的对象，它们应该在恢复中。

**Stale** （不新鲜）归置组处于未知状态：存储它们的 OSD 有段时间没向监视器报告了（由 mon\_osd\_report\_timeout 配置）。

可用格式有 plain （默认）和 json 。阀值定义的是，归置组被认为卡住前等待的最小时间（默认 300 秒）。

## 11.5 获取一归置组运行图

要获取一个具体归置组的归置组图，执行命令：

ceph pg map {pg-id}

例如：

ceph pg map 1.6c

Ceph 将返回归置组图、归置组、和 OSD 状态：

osdmap e13 pg 1.6c (1.6c) -> up [1,0] acting [1,0]

## 11.6 获取一 PG 的统计信息

要查看一个具体归置组的统计信息，执行命令：

ceph pg {pg-id} query

## 11.7 清理归置组

要清理归置组，请执行以下操作：

ceph pg scrub { pg - id }

Ceph检查主节点和任何副本节点，生成展示位置组中所有对象的目录，并对它们进行比较以确保没有对象丢失或不匹配，并且其内容是一致的。假设副本全部匹配，最后的语义扫描确保所有与快照有关的对象元数据是一致的。错误通过日志报告。

## 11.8 优先考虑安置组的回填/恢复（★英文文档新增）

您可能会遇到这样一种情况，即一组布局组需要恢复和/或回填，并且某些特定组保存的数据比其他组更重要（例如，这些PG可能会保存运行机器使用的图像的数据，而其他PG可能由不活动的机器使用/不太相关的数据）。在这种情况下，您可能需要优先恢复这些组，以便更早地恢复存储在这些组上的数据的性能和/或可用性。要执行此操作（在回填或恢复期间将特定展示位置组标记为优先），请执行以下操作：

ceph pg force - recovery { pg - id } [{ pg - id ＃2}] [{pg-id＃3} ...]

ceph pg force - backfill { pg - id } [{ pg - id ＃2}] [{pg-id＃3} ...]

这将导致Ceph在指定的展示位置组之前首先执行恢复或回填。这不会中断目前正在进行的回填或回收，但会导致指定的PG尽快处理。如果您改变主意或优先考虑错误的群体，请使用：

ceph pg cancel - force - recovery { pg - id } [{ pg - id ＃2}] [{pg-id＃3} ...]

ceph pg cancel - force - backfill { pg - id } [{ pg - id ＃2}] [{pg-id＃3} ...]

这将从这些PG中删除“强制”标志，并将按默认顺序进行处理。同样，这不会影响当前处理的放置组，只有那些仍在排队的组。

组的恢复或回填完成后，“强制”标志会自动清除。

## 11.9 恢复丢失的

如果集群丢了一或多个对象，而且必须放弃搜索这些数据，你就要把未找到的对象标记为丢失（ lost ）。

如果所有可能的位置都查询过了，而仍找不到这些对象，你也许得放弃它们了。这可能是罕见的失败组合导致的，集群在写入完成前，未能得知写入是否已执行。

当前只支持 revert 选项，它使得回滚到对象的前一个版本（如果它是新对象）或完全忽略它。要把 unfound 对象标记为 lost ，执行命令：

ceph pg {pg-id} mark\_unfound\_lost revert|delete

**Important**

要谨慎使用，它可能迷惑那些期望对象存在的应用程序。

# 12. 使用PG-UPMAP（★英文文档新增）

从Luminous v12.2.z开始，OSDMap中有一个新的*pg-upmap*异常表，允许集群明确地将特定的PG映射到特定的OSD。这使得集群可以微调数据分布，在大多数情况下，可以跨OSD使用完美的分布式PG。

这个新机制的关键是要求所有客户端都了解*OSDMap中*新的*pg-upmap*结构。

## 12.1 启用

要允许使用该功能，您必须用以下命令告诉集群它只需要支持luminous (和newer)的客户端：

ceph osd set-require-min-compat-client luminous

如果任何pre-luminous的客户端或守护进程连接到监视器，则此命令将失败。您可以看到正在使用哪些客户端版本：

ceph features



## 12.2 一个忠告

这是一个新功能，不是非常友好。在写这篇文章的时候，我们正在为ce​​ph-mgr 开发一个新的平衡器模块，最终自动完成所有这些功能。

直到那时，……（英文文档未写完）

## 12.3 离线优化

Upmap条目使用内置的离线优化器进行更新osdmaptool。

1. 获取osdmap的最新副本：

ceph osd getmap -o om



1. 运行优化器:

osdmaptool om --upmap out.txt [--upmap-pool <pool>] [--upmap-max <max-count>] [--upmap-deviation <max-deviation>]

强烈建议对每个池单独进行优化，或者对同类池进行优化。您可以--upmap-pool多次指定选项。“类似池”是指映射到相同设备并存储相同类型数据（例如，RBD映像池，是，RGW索引池和RGW数据池，否）的池。

该max-count值是运行中要标识的上图条目的最大数量。默认值为100，但您可能希望将其设置为较小的值，以便工具更快地完成（但工作量较少）。如果它找不到任何额外的变化，使它会提前停止（即当分配池是完美的）。

该max-deviation值默认为*.01*（即1％）。如果OSD利用率与平均值相差小于这个数值，那么它将被认为是完美的。

1. 建议的更改将写入out.txt上面的示例中的输出文件。这些是可以运行以将更改应用到集群的正常的ceph CLI命令。这可以通过以下方式完成：

source out.txt

上述步骤可以根据需要重复多次，以实现每组池的PG的完美分配。

# 13. CRUSH 图

CRUSH 算法通过计算数据存储位置来确定如何存储和检索。 CRUSH 授权 Ceph 客户端直接连接 OSD ，而非通过一个中央服务器或经纪人。数据存储、检索算法的使用，使 Ceph 避免了单点故障、性能瓶颈、和伸缩的物理限制。

CRUSH 需要一张集群的地图，且使用 CRUSH 把数据伪随机地存储、检索于整个集群的 OSD 里。 CRUSH 的讨论详情参见 [CRUSH - 可控、可伸缩、分布式地归置多副本数据](http://ceph.com/papers/weil-crush-sc06.pdf) 。

CRUSH 图包含 OSD 列表、把设备汇聚为物理位置的“桶”列表、和指示 CRUSH 如何复制存储池里的数据的规则列表。由于对所安装底层物理组织的表达， CRUSH 能模型化、并因此定位到潜在的相关失败设备源头，典型的源头有物理距离、共享电源、和共享网络，把这些信息编码到集群运行图里， CRUSH 归置策略可把对象副本分离到不同的失败域，却仍能保持期望的分布。例如，要定位同时失败的可能性，可能希望保证数据复制到的设备位于不同机架、不同托盘、不同电源、不同控制器、甚至不同物理位置。

当你写好配置文件，用 ceph-deploy 部署 Ceph 后，它生成了一个默认的 CRUSH 图，对于你的沙盒环境来说它很好。然而，部署一个大规模数据集群的时候，应该好好设计自己的 CRUSH 图，因为它帮你管理 Ceph 集群、提升性能、和保证数据安全性。

例如，如果一个 OSD 挂了， CRUSH 图可帮你定位此事件中 OSD 所在主机的物理数据中心、房间、行和机架，据此你可以请求在线支持或替换硬件。

类似地， CRUSH 可帮你更快地找出问题。例如，如果一个机架上的所有 OSD 同时挂了，问题可能在于机架的交换机或电源，而非 OSD 本身。

定制的 CRUSH 图也能在归置组降级时，帮你找出冗余副本所在主机的物理位置。

## 13.1 CRUSH 位置（★英文文档改动）

用 CRUSH 图层次结构所表示的 OSD 位置被称为“ crush 位置”，它用键/值对列表来表示。例如，一 OSD 位于某特定行、机柜、机架、和主机，且是 CRUSH 图里名为 default 树的一部分，那么其 crush 位置可表示如下：

root=default row=a rack=a2 chassis=a2a host=a2a1

注：

1、注意键（关键词）与顺序无关；

2、键名（ = 左边）必须是 CRUSH 内的合法 type ，默认情况下，它包含 root 、 datacenter 、 room 、 row 、 pod 、 pdu 、 rack 、 chassis 、和 host ，但这些类型可修改 CRUSH 图任意定义。

3、并非所有键都需指定，例如，默认情况下 Ceph 会自动把 ceph-osd 守护进程的位置设置为 root=default host=HOSTNAME（即是 hostname -s ）。

### 定制位置挂钩

自定义的位置挂钩可用于在启动时生成更完整的crush位置。示例ceph-crush-location应用程序将为给定的守护进程生成一个CRUSH位置字符串，位置是按照以下优先顺序：

1. ceph.conf中的crush location选项；
2. 使用hostname -s命令生成的主机名默认值root=default host=HOSTNAME。

这本身并没有用，因为OSD本身具有完全相同的行为。然而，可以修改脚本以提供额外的位置字段（例如机架rank或数据中心datacenter），然后通过配置选项启用挂钩：

crush location hook = /path/to/customized-ceph-crush-location

此挂钩应该接受几个参数（下述）并向标准输出打印一行 CRUSH 位置描述：

$ ceph-crush-location --cluster CLUSTER --id ID --type TYPE

其中，集群名通常是 ceph ， id 是守护进程标识符（ OSD 号），守护进程类型通常是 osd 。

## 13.2 CRUSH结构（★英文文档改动）

CRUSH图简单地说就是描述集群物理拓扑结构的层次结构，以及一组定义关于如何在这些设备上放置数据的策略的规则。层次结构ceph-osd在叶子上有设备（守护进程），内部节点对应于其他物理特征或分组：主机，机架，行，数据中心等等。这些规则描述了复制品是如何按照这种等级来安置的（例如，“不同货架上的三个复制品”）。

### 13.2.1 设备

设备是ceph-osd可以存储数据的个人守护进程。通常在这里为您的集群中的每个OSD守护进程定义一个。设备由一个id（一个非负整数）和一个名称来标识，通常osd.N在N设备id的什么地方。

设备也可能有一个与它们相关的*设备类*（例如， hdd或ssd），使它们可以方便地通过粉碎规则。

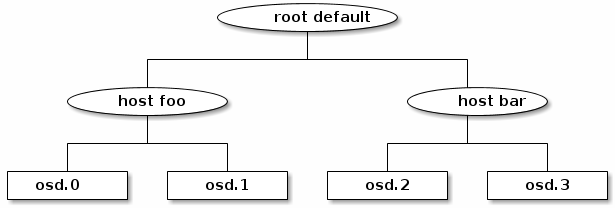
### 13.2.2 类型和桶

一个桶是层次结构中内部节点的CRUSH术语：主机，机架，行等CRUSH映射定义了一系列用于描述这些节点的类型。默认情况下，这些类型包括：

* osd（或设备）
* 主办
* 机壳
* 架
* 行
* PDU
* 荚
* 房间
* 数据中心
* 地区
* 根

大多数集群只使用这些类型中的一小部分，而其他集群可以根据需要进行定义。

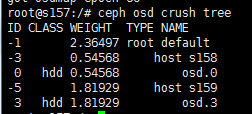
层次结构是使用osd叶子上的设备（通常是类型），非设备类型的内部节点以及类型的根节点 构建的root。例如，



层次结构中的每个节点（设备或存储桶）都有一个 与其关联的*权重*，指示设备或层次结构子树应存储的总数据的相对比例。权重设置在树叶上，指示设备的大小，并从那里自动地总结树，使得默认节点的权重将是包含在其下面的所有设备的总和。通常重量是以兆兆字节（TB）为单位的。

您可以通过以下方式简单查看集群的CRUSH层次结构（包括权重）：

ceph osd crush tree



### 13.2.3 规则

规则定义了有关数据如何分布在层次结构中的设备上的策略。

CRUSH规则定义了放置和复制策略或分发策略，允许您指定CRUSH如何放置对象副本。例如，您可以创建一个规则，为2路镜像选择一对目标，在两个不同的数据中心选择三个目标进行3路镜像的另一个规则，以及六个存储设备上的纠删码的另一个规则。有关CRUSH规则的详细讨论，请参阅[CRUSH - 复制数据的受控，可扩展，分散放置](https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf)，更具体地说，参见3.2节。

在几乎所有情况下，都可以通过CLI创建CRUSH规则，方法是指定将用于（复制或​​纠删码）的池类型，失败域以及可选的设备类。在极少数情况下，必须通过手动编辑CRUSH图来手动编写规则。

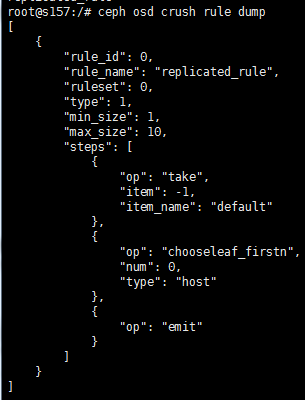
查看集群的CRUSH规则：

ceph osd crush rule ls



查看规则的内容：

ceph osd crush rule dump



### 13.2.4 设备类

每个设备可以有一个与之关联的*类*。默认情况下，OSD在启动时自动将其类设置为 *hdd*，*ssd*或*nvme，*这取决于它们所支持的设备类型。

一个或多个OSD的设备类可以显式设置：

ceph osd crush set-device-class <class> <osd-name> [...]

一旦设置了一个设备类，直到旧类未被设置，它就不能被改变为另一个类。

ceph osd crush rm-device-class <osd-name> [...]

这允许管理员设置设备类，而不必在OSD重启或其他脚本中更改类。

针对特定设备类的展示位置规则可以通过以下方式创建：

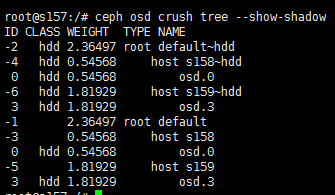
ceph osd crush rule create - replicated < rule - name > < root > < failure - domain > < class >

然后可以将池更改为使用新规则：

ceph osd pool set <pool-name> crush\_rule <rule-name>

设备类是通过为每个正在使用的设备类创建一个“阴影”CRUSH层次结构来实现的，该设备类只包含那个类的设备。然后，规则可以在影子层次结构上分配数据。这种方法的一个好处是，它完全向后兼容旧的Ceph客户端。您可以使用以下查看具有阴影项目的CRUSH层次结构：

ceph osd crush tree --show-shadow



### 13.2.5 权重集

权重集是一种权重计算数据放置时使用的替代集。与CRUSH图中每个设备相关的正常权重是根据设备大小设置的，并指出我们应该在多少数据中存储数据。然而，因为CRUSH是基于伪随机的放置过程，所以这个理想的分布总会有一些变化，就像掷骰子六十次一样，也不会导致十个十六进制。权重集允许集群根据集群（层次结构，池等）的具体情况进行数值优化，以实现均衡分布。

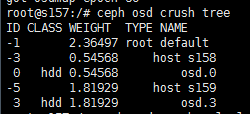
有两种类型的权重集支持：

1、compat权重集为集群中的每个设备和节点的权重的单替代集。这并不适用于纠正所有异常情况（例如，不同池的放置组可能会有不同的大小，并具有不同的负载水平，但大多数情况下平衡器会对其进行相同的处理）。然而，compat权重集具有与以前版本的Ceph 向后兼容的巨大优势，这意味着即使Luminous v12.2.z中首次引入权重集，较老的客户端（例如，firefly）仍然可以连接到集群权重集合用于平衡数据。

2、每个池的权重集（A **per-pool** weight set）更加灵活是因为它允许放置到每个数据池来优化。此外，可以针对每个放置位置调整权重，从而使优化器能够针对相对于同级的权重较小的设备来校正数据的倾斜偏差（并且通常只在非常大的集群中出现这种影响，但这会导致平衡问题）。

使用权重集时，与层次结构中每个节点相关联的权重将作为单独的列（标记为 (compat)或者池名称）显示在命令中：

ceph osd crush tree



当compat和per-pool权重集都在使用中时，特定池的数据放置将使用其自己的每个池权重集（如果存在）。如果不存在，则将使用相同的重量组（如果存在）。如果两者都不存在，它将使用正常CRUSH权重。

尽管可以手动设置和操作权重集，但是建议平衡器模块自动启用。

## 13.3 修改CRUSH MAP

### 13.3.1 添加/移动OSD

在正在运行的集群的CRUSH映射中添加或移动OSD：

ceph osd crush set {name} {weight} root={root} [{bucket-type}={bucket-name} ...]

*参数说明：*

**name**

|  |  |
| --- | --- |
| 描述： | OSD的全名。 |
| 类型： | String |
| 需要： | 是 |
| 例： | osd.0 |

**weight**

|  |  |
| --- | --- |
| 描述： | OSD的CRUSH权重，通常以（TB）为单位。 |
| 类型： | Double |
| 需要： | 是 |
| 例： | 2.0 |

**root**

|  |  |
| --- | --- |
| 描述： | OSD所在树的根节点（通常default） |
| 类型： | 键/值对。 |
| 需要： | 是 |
| 例： | root=default |

**bucket-type**

|  |  |
| --- | --- |
| 描述： | 您可以在CRUSH层次结构中指定OSD的位置。 |
| 类型： | 键/值对。 |
| 需要： | 没有 |
| 例： | datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1 |

以下示例将添加osd.0到层次结构中，或从以前的位置移动OSD。

ceph osd crush set osd.0 1.0 root=default datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1

### 13.3.2 调整OSD的CRUSH权重

要调整正在运行的集群的OSD CRUSH权重，请执行以下操作：

ceph osd crush reweight {name} {weight}

*参数说明：*

**name**

|  |  |
| --- | --- |
| 描述： | OSD的全名。 |
| 类型： | String |
| 需要： | 是 |
| 例： | osd.0 |

**weight**

|  |  |
| --- | --- |
| 描述： | OSD的CRUSH权重。 |
| 类型： | Double |
| 需要： | 是 |
| 例： | 2.0 |

### 13.3.3 删除OSD

要从正在运行的集群的CRUSH映射中删除OSD，请执行以下操作：

ceph osd crush remove { name }

*参数说明：*

**name**

|  |  |
| --- | --- |
| 描述： | OSD的全名。 |
| 类型： | String |
| 需要： | 是 |
| 例： | osd.0 |

### 13.3.4 添加桶

要在正在运行的集群的CRUSH映射中添加存储桶，请执行ceph osd crush add-bucket命令：

ceph osd crush add-bucket {bucket-name} {bucket-type}

*参数说明：*

**bucket-name**

|  |  |
| --- | --- |
| 描述： | 桶的全名。 |
| 类型： | String |
| 需要： | 是 |
| 例： | rack12 |

**bucket-type**

|  |  |
| --- | --- |
| 描述： | 桶的类型。类型必须已经存在于层次结构中。 |
| 类型： | String |
| 需要： | 是 |
| 例： | rack |

以下示例将rack12存储区添加到层次结构中：

ceph osd crush add-bucket rack12 rack

### 13.3.5 移动桶

要将存储区移动到CRUSH地图层次结构中的其他位置或位置，请执行以下操作：

ceph osd crush move {bucket-name} {bucket-type}={bucket-name}, [...]

*参数说明：*

**bucket-name**

|  |  |
| --- | --- |
| 描述： | 要移动/重新定位的存储桶的名称。 |
| 类型： | String |
| 需要： | 是 |
| 例： | foo-bar-1 |

**bucket-type**

|  |  |
| --- | --- |
| 描述： | 您可以在CRUSH层次结构中指定存储区的位置。 |
| 类型： | 键/值对。 |
| 需要： | 没有 |
| 例： | datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1 |

### 13.3.6 删除桶

要从CRUSH地图层次结构中删除存储桶，请执行以下操作：

ceph osd crush remove {bucket-name}

**注意：**从CRUSH层次结构中删除它之前，一个存储桶必须是空的。

*参数说明：*

**bucket-name**

|  |  |
| --- | --- |
| 描述： | 要删除的存储桶的名称。 |
| 类型： | String |
| 需要： | 是 |
| 例： | rack12 |

以下示例rack12从层次结构中删除存储区：

ceph osd crush remove rack12

### 13.3.7 创建compat 权重集（★英文文档新增）

要创建一个*compat*权重集：

ceph osd crush weight-set create-compat

*compat*权重集设定的权重可以调整：

ceph osd crush weight-set reweight-compat {name} {weight}

compat权重集可以删除：

ceph osd crush weight-set rm-compat

### 13.3.8 创建per-pool 权重集（★英文文档新增）

为特定的池创建权重集，请执行以下操作：

ceph osd crush weight-set create {pool-name} {mode}

注意：按池计算权重集要求所有服务器和守护进程运行Luminous v12.2.z或更高版本。

*参数说明：*

**pool-name**

|  |  |
| --- | --- |
| 描述： | RADOS池的名称。 |
| 类型： | String |
| 需要： | 是 |
| 例： | rbd |

**mode**

|  |  |
| --- | --- |
| 描述： | flat或positional。flat权重集对每个设备或桶有一个权重。positional权重集在产生的展示位置映射中每个位置有潜在的不同权重。例如，如果某个池的副本数为3，则对于每个设备和存储桶，positional权重集有三个权重。 |
| 类型： | String |
| 需要： | 是 |
| 例： | flat |

调整权重集中项目的权重：

ceph osd crush weight-set reweight {pool-name} {item-name} {weight [...]}

要列出现有的权重集：

ceph osd crush weight-set ls

删除一个权重集：

ceph osd crush weight-set rm {pool-name}

### 13.3.9 为副本池创建规则（★英文文档新增）

对于复制池，创建CRUSH规则时的主要决定是失败域将会是什么。例如，如果选择了一个失败域host，则CRUSH将确保每个数据副本存储在不同的主机上。如果rack 被选中，则每个副本将被存储在不同的机架中。您选择哪个失败域主要取决于您的集群的大小以及您的层次结构的结构。

通常，整个集群层次结构嵌套在名为的根节点下default。如果您已经自定义了层次结构，则可能需要创建一个嵌套在层次结构中某个其他节点上的规则。与该节点关联的类型并不重要（它不一定是root节点）。

也可以创建一个将数据放置限制到特定设备类别的规则。默认情况下，Ceph OSD会根据所使用设备的基本类型自动将其自己分类为hdd或ssd。这些类也可以定制。

要创建复制规则，请执行以下操作：

ceph osd crush rule create-replicated {name} {root} {failure-domain-type} [{class}]

*参数说明：*

**name**

|  |  |
| --- | --- |
| 描述： | 规则的名称。 |
| 类型： | String |
| 需要： | 是 |
| 例： | rbd-rule |

**root**

|  |  |
| --- | --- |
| 描述： | 应该放置数据的节点的名称。 |
| 类型： | String |
| 需要： | 是 |
| 例： | default |

**failure-domain-type**

|  |  |
| --- | --- |
| 描述： | 我们应该分离副本的CRUSH节点的类型。 |
| 类型： | String |
| 需要： | 是 |
| 例： | rack |

**class**

|  |  |
| --- | --- |
| 描述： | 设备类的数据应该放在上面。 |
| 类型： | String |
| 需要： | 没有 |
| 例： | ssd |

### 13.3.10 为纠删码池创建规则（★英文文档新增）

对于纠删码池，需要对复制池进行相同的基本决策：什么是故障域，层次结构中的哪个节点将数据置于（通常default）下，并将放置限制在特定设备类。然而，擦除代码池的创建有点不同，因为它们需要根据所使用的纠删码仔细构建。出于这个原因，您必须将此信息包含在*纠删码配置文件中*。随后将使用该配置文件创建池时显式或自动创建CRUSH规则。

可以列出纠删码配置文件：

ceph osd erasure-code-profile ls



查看现有的配置文件：

ceph osd erasure-code-profile get {profile-name}

通常配置文件不应该被修改；相反，在创建新池或为现有池创建新规则时，应创建并使用新的配置文件。

纠删码配置文件由一组键=值对组成。其中大部分控制着编码池中数据的纠删码的行为。然而，那些以crush-开始的，影响创建的CRUSH规则。

纠删码配置文件参数属性：

* **crush-root**：将数据放置在[default：default] 下的CRUSH节点的名称。
* **crush-failure-domain**：CRUSH类型在[默认：host] 之间分隔纠删码的碎片。
* **crush-device-class**：将数据置于[default：none，表示使用所有设备]的设备类。
* **k**和**m**（对于lrc插件，**l**）：这些决定了纠删代码碎片的数量，从而影响最终的CRUSH规则。

一旦定义了配置文件，您可以创建CRUSH规则：

ceph osd crush rule create-erasure {name} {profile-name}

可以使用以下方法删除池中未使用的规则：

ceph osd crush rule rm {rule-name}

### 13.3.10 删除规则（★英文文档新增）

可以使用以下方法删除池中未使用的规则：

ceph osd crush rule rm {rule-name}

## 13.4 可调参数

### 13.4.1 可调选项非最优时发出警告

从 v0.74 起，如果 CRUSH 可调选项不是最优值（ v0.73 版里的默认值） Ceph 就会发出健康告警，有两种方法可消除这些告警：

1、调整现有集群上的可调选项。注意，这可能会导致一些数据迁移（可能有 10% 之多）。这是推荐的办法，但是在生产集群上要注意此调整对性能带来的影响。此命令可启用较优可调选项：

ceph osd crush tunables optimal

如果切换得不太顺利（如负载太高）且切换才不久，或者有客户端兼容问题（较老的 cephfs 内核驱动或 rbd 客户端、或早于 bobtail 的 librados 客户端），你可以这样切回：

ceph osd crush tunables legacy

2、不对 CRUSH 做任何更改也能消除报警，把下列配置加入 ceph.conf 的 [mon] 段下：

mon warn on legacy crush tunables = false

为使变更生效需重启所有监视器，或者执行下列命令：

ceph tell mon.\\* injectargs --no-mon-warn-on-legacy-crush-tunables

### 13.4.2 一些要点

* **调整这些值将使一些 PG 在存储节点间移位，如果 Ceph 集群已经存储了大量数据，做好移动一部分数据的准备。**
* **一旦更新运行图， ceph-osd 和 ceph-mon 就会开始向新建连接要求功能位，然而，之前已经连接的客户端如果不支持新功能将行为失常。**
* **如果 CRUSH 可调值更改过、然后又改回了默认值， ceph-osd 守护进程将不要求支持此功能，然而， OSD 连接建立进程要能检查和理解旧地图。因此，集群如果用过非默认 CRUSH 值就不应该再运行版本小于 0.48.1 的 ceph-osd ，即使最新版地图已经回滚到了遗留默认值。**

### 13.4.3 调整CRUSH

更改 crush 可调值的最简方法就是改到一个已知配置，它们有：

* legacy：采用 argonaut 及更低版本的行为；
* argonaut：采用 argonaut 版最初的配置；
* bobtail: 采用 bobtail 版的配置；
* firefly: 采用 firefly 版的配置；
* hammer：采用hammer版的配置；
* jewel：采用jewel版的配置；
* optimal: 采用当前最佳配置；
* default: 新建集群可采用当前默认值。这些取决于当前版本的Ceph的值是否是硬编码的、通常是最优值和传统值的混合。这些值通常与optimal先前的LTS版本的配置文件相匹配，或者与我们通常除了更多的用户拥有最新客户端的最新版本相匹配。

您可以使用以下命令在正在运行的集群上选择配置文件：

ceph osd crush tunables { PROFILE }

请注意，这可能会导致一些数据移动。

## 13.5 主亲和性

一 Ceph 客户端读写数据时，总是连接 acting set 里的主 OSD （如 [2, 3, 4] 中， osd.2 是主的）。有时候某个 OSD 与其它的相比并不适合做主 OSD （比如其硬盘慢、或控制器慢），最大化硬件利用率时为防止性能瓶颈（特别是读操作），你可以调整 OSD 的主亲和性，这样 CRUSH 就尽量不把它用作 acting set 里的主 OSD 了。

ceph osd primary-affinity <osd-id> <weight>

主亲和性默认为 1 （就是说此 OSD 可作为主 OSD ）。此值合法范围为 0-1 ，其中 0 意为此 OSD 不能用作主的， 1 意为 OSD 可用作主的；此权重小于 1 时， CRUSH 选择主 OSD 时选中它的可能性低。

# 14. 手动编辑CRUSH映射（★英文文档新增）

注意：手动编辑CRUSH映射被认为是高级管理员操作。绝大多数安装所必需的CRUSH变化可以通过标准的ceph CLI进行，不需要手动编辑CRUSH映射。如果你已经确定了手动编辑的使用情况是必要的，考虑联系Ceph的开发者，使ceph的未来版本可以减少这种不必要的手动操作。

要编辑现有的CRUSH映射：

1、获取CRUSH映射。

2、反编译 CRUSH映射。

3、编辑至少一个设备、存储桶和规则。

4、重新编译 CRUSH映射。

5、设置CRUSH映射。

## 14.1 获取CRUSH映射

要获取集群的CRUSH映射，请执行以下操作：

ceph osd getcrushmap -o {compiled-crushmap-filename}

Ceph将输出（-o）编译的CRUSH映射到您指定的文件名。由于CRUSH映射是以编译形式存在的，因此必须先对其进行反编译，然后才能对其进行编辑。

## 14.2 反编译CRUSH映射

反编译CRUSH映射，请执行以下操作：

crushtool -d {compiled-crushmap-filename} -o {decompiled-crushmap-filename}

## 14.3 组成

CRUSH Map有六个主要部分。

1. **可调参数：**映射顶部的前导码描述 了CRUSH行为的任何*可调参数*，这些*可调参数都*与历史/传统CRUSH行为不同。这些修正了旧的bug，优化或多年来为优化CRUSH性能而做出的其他改变。
2. **设备：**设备是可以存储数据的单个ceph-osd守护进程。
3. **类型**：Bucket types定义了CRUSH层次结构中使用的存储区的类型。存储桶由存储位置（例如：行，机架，机箱，主机等）及其分配权重的层次聚合组成。
4. **存储区：**一旦定义存储区类型，您必须定义层次结构中的每个节点，其类型以及它所包含的设备或其他节点。
5. **规则：**规则定义了数据是如何在层次结构中的设备间分布的有关策略。
6. **choose\_args：** Choose\_args是与层次关联的替代权重，层次已经过调整以优化数据放置。一个choose\_args映射可以用于整个集群，或者可以为每个单独的池创建一个。

## 14.4 CRUSH映射设备

设备是可以存储数据的单个ceph-osd守护进程。通常在这里为您的集群中的每个OSD守护进程定义一个。设备由一个id（一个非负整数）和一个名称来标识，通常为osd.N，其中N表示设备id。

设备也可能有一个与它们相关的设备类（例如， hdd或ssd），使它们可以方便地通过CRUSH规则定位。

# devices

device {num} {osd.name} [class {class}]

例如：

# devices

device 0 osd.0 class ssd

device 1 osd.1 class hdd

device 2 osd.2

device 3 osd.3

在大多数情况下，每个设备映射到一个单独的ceph-osd守护进程。这通常是单个存储设备，一对设备（例如，一个用于数据，另一个用于日志或元数据），或者在某些情况下是小型RAID设备。

## 14.5 CRUSH映射桶类型

CRUSH图中的第二个列表定义了“桶”类型。桶促进节点和叶子的层次结构。节点（或非叶）桶通常代表层次结构中的物理位置。节点聚合其他节点或叶子节点。叶子桶代表ceph-osd守护程序及其相应的存储介质。

**Tips：**在CRUSH的上下文中使用的术语“桶”意味着层次中的节点，即位置或物理硬件。当在RADOS网关API的环境中使用时，它与“桶”这个术语是不同的概念。

要将一个存储桶类型添加到CRUSH映射中，请在存储桶类型列表下创建一个新行。输入type后跟一个唯一的数字ID和存储桶名称。按照惯例，有一个叶子桶，它是type 0；然而，你可以给它设置任何你喜欢的名字（e.g., osd, disk, drive, storage,等等）：

#types

type { num } { bucket - name }

例如：

# types

type 0 osd

type 1 host

type 2 chassis

type 3 rack

type 4 row

type 5 pdu

type 6 pod

type 7 room

type 8 datacenter

type 9 region

type 10 root

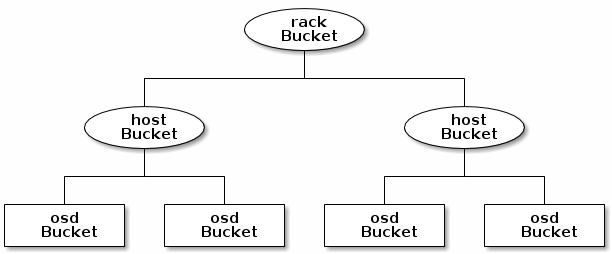
## 14.6 CRUSH地图桶层次

CRUSH算法根据每个设备权重值在存储设备之间分配数据对象，近似均匀的概率分布。CRUSH根据您定义的层次聚簇映射来分发对象及其副本。您的CRUSH映射表示可用的存储设备以及包含它们的逻辑元素。

为了将布局组（placement group）映射到跨越故障域的OSD，CRUSH映射定义了存储桶类型的分层列表（例如，#types在生成的CRUSH映射下）。创建存储桶层次结构的目的是通过其故障域（例如主机，机箱，机架，配电设备，吊舱，行，房间和数据中心）来隔离叶节点。除了代表OSD的叶节点之外，其余的层次结构是任意的，您可以根据自己的需要来定义它。

我们建议根据您公司的硬件命名约定来调整您的CRUSH映射，并使用反映物理硬件的实例名称。当OSD和/或其他硬件出现故障并且管理员需要访问物理硬件时，您的命名实践可以更容易地管理集群并解决问题。

在以下示例中，存储区层次结构具有名为osd的叶子桶和两个分别命名为host、rack的节点存储桶。



注意：编号较高的rack桶类型汇聚了编号较小的host桶类型。

由于叶节点反映CRUSH映射开始时在#devices列表下声明的存储设备，因此不需要将它们声明为存储桶实例。您的层次结构中次低的存储桶类型通常会聚集设备（例如，通常是包含存储介质的计算机，并使用您喜欢描述的任何术语，如“节点”，“计算机”，“服务器”，“主机“，”机器“等）。在高密度环境中，每个机箱看到多个主机/节点的情况越来越普遍。您也应该考虑机箱故障 - 例如，如果节点出现故障，需要拔出机箱，可能会导致无数主机/节点及其OSD出现故障。

当声明一个桶实例时，你必须指定它的类型，给它一个唯一的名字（字符串），给它赋一个唯一的ID，用一个负数表示（可选），指定一个相对于其项目的总容量/容量的权重，指定桶算法（通常为straw）和散列（通常为0，反映散列算法rjenkins1）。一个桶可能有一个或多个项目。这些项目可能由节点桶或叶子组成。物品的重量可以反映物品的相对重量。

您可以使用以下语法声明节点存储桶：

[bucket-type] [bucket-name] {

id [a unique negative numeric ID]

weight [the relative capacity/capability of the item(s)]

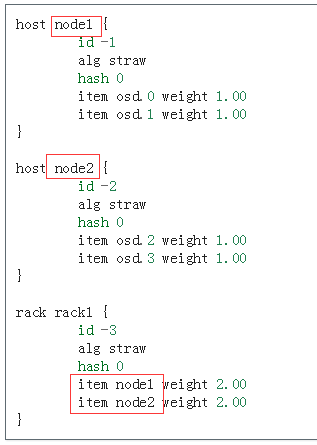
alg [the bucket type: uniform | list | tree | straw ]

hash [the hash type: 0 by default]

item [item-name] weight [weight]

}

例如，使用上面的语法，我们将定义两个主机桶和一个机架桶。OSD被声明为主桶中的项目：



**注意**

在上述示例中，请注意，机架桶不包含任何OSD。相反，它包含较低级别的主机桶，并包括项目条目中它们的权重总和。

**桶类型**

Ceph支持四种桶类型，每种类型代表了性能和重组效率之间的权衡。如果您不确定使用哪种桶类型，建议使用straw桶。有关桶类型的详细讨论，请参阅文档[CRUSH - 复制数据的受控，可扩展，分散布置](https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf)，更具体地说，参见 **第3.4节**。桶类型是：

1. **Uniform**：统一的桶集合设备具有**完全相同**的权重。例如，当企业委托或退役硬件时，他们通常会使用许多具有完全相同物理配置的机器（例如批量购买）。当存储设备具有完全相同的权重时，可以使用uniform存储桶类型，这样CRUSH可以在一段时间内将副本映射到统一存储桶中。使用不统一的权重，您应该使用另一个桶算法。
2. **List**：列表桶将其内容聚集为链接列表。基于RUSH P算法，列表对于**扩展集群来说**是一个自然而直观的选择：要么以一定的概率将对象重新定位到最新的设备，要么像以前一样保留在较旧的设备上。当项目添加到存储桶时，结果是最佳的数据迁移。但是，从列表的中间或尾部删除的项目可能会导致大量不必要的移动，因此列表桶最适合它们**从不（或很少）缩小的情况**。
3. **Tree**：树桶使用二叉搜索树。当一个存储桶包含更多的项目时，它们比列表存储更有效率。基于RUSH R算法，树形存储桶将布局时间缩短到O（log n），使其适用于管理更大型设备或嵌套桶。
4. **Straw**：列表桶和树桶使用分而治之的策略，以某种方式给予某些项目的优先权（例如，在列表的开头），或者根本不需要考虑整个子项目。这提高了副本放置过程的性能，但是当桶的内容由于项目的添加，移除或重新加权而改变时，也可能引入次优重组行为。吸管桶（straw bucket）类型允许所有物品通过与吸管相似的过程相互“竞争”以进行副本放置。
5. **Straw2**：Straw2改善了Straw，能够正确避免任何相邻权重变化时数据之间的移动。例如，重新添加或完全删除项目A的权重时，将只从项目A移入或移出数据。

**哈希**

每个桶使用一个哈希算法。目前，Ceph支持rjenkins1。输入0作为您的哈希设置来选择rjenkins1。

**分配桶的项目的权重**

Ceph将桶权重表示为double，可以进行精细加权。权重是设备容量之间的相对差异。我们建议使用1.00作为1TB存储设备的相对权重。在这种情况下，权重0.5将代表大约500GB，权重3.00将代表大约3TB。更高级别的桶具有由该桶聚合的叶子项目的总权重。

存储区项目权重是一维的，但是您也可以计算您的项目权重以反映存储驱动器的性能。例如，如果您有许多1TB硬盘，其中一些硬盘的数据传输速率相对较低，其他硬盘的数据传输速率相对较高，那么即使它们具有相同的容量，你可以不同地表示它们的权重（例如，第一组驱动器总吞吐量较低，权重为0.80；第二组驱动器总吞吐量较高，为1.20）。

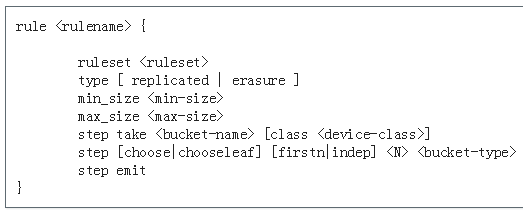
## 14.7 CRUSH映射规则

CRUSH映射支持“CRUSH规则”的概念，决定存储池的数据放置的规则。默认的CRUSH映射对每个池都有一个规则。对于大型集群，您可能会创建许多池，其中每个池可能都有自己的非默认CRUSH规则。

**注意：**在大多数情况下，您将不需要修改默认规则。当您创建新的池时，默认情况下，该规则将被设置为0。

CRUSH规则定义了放置和副本策略或分发策略，允许您指定CRUSH如何放置对象副本。例如，您可以为2路镜像选择一对目标创建规则1，为3路镜像在两个不同的数据中心选择三个目标创建规则2，为六个存储设备上的纠删码创建规则3。有关CRUSH规则的详细讨论，请参阅[CRUSH - 复制数据的受控，可扩展，分散放置](https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf)，更具体地说，参见 **3.2节**。

规则采取以下形式：



*参数说明：*

**ruleset**

|  |  |
| --- | --- |
| 描述： | 用于识别规则的唯一整数。ruleset 这个名字是过去的遗留物，每个池可能有多个CRUSH规则。 |
| 目的： | 规则掩码的一个组成部分。 |
| 类型： | Integer |
| 是否需要： | 是 |
| 默认值 | 0 |

**Type**

|  |  |
| --- | --- |
| 描述： | 描述存储驱动器（复制）或RAID的规则。 |
| 目的： | 规则掩码的一个组成部分。 |
| 类型： | String |
| 是否需要： | 是 |
| 默认值 | replicated |
| 有效值 | 目前只有replicated和erasure |

**min\_size**

|  |  |
| --- | --- |
| 描述： | 如果一个池的副本少于这个数字，CRUSH将 不会选择这个规则。 |
| 目的： | 规则掩码的一个组成部分。 |
| 类型： | Integer |
| 是否需要： | 是 |
| 默认值 | 1 |

**max\_size**

|  |  |
| --- | --- |
| 描述： | 如果一个池的副本比这个数字更多，CRUSH将 不会选择这个规则。 |
| 目的： | 规则掩码的一个组成部分。 |
| 类型： | Integer |
| 是否需要： | 是 |
| 默认值 | 10 |

**step take <bucket-name> [class <device-class>]**

|  |  |
| --- | --- |
| 描述： | 取一个桶名称，并开始迭代树。如果device-class指定，它必须与定义设备时以前使用的类相匹配。所有不属于该类别的设备都被排除在外。 |
| 目的： | 规则的一个组成部分。 |
| 是否需要： | 是 |
| 例子： | step take data |

**step choose firstn {num} type {bucket-type}**

|  |  |
| --- | --- |
| 描述： | 选择给定类型的桶的数量。数字通常是池中副本的数量（例：池大小）。  1、如果{num}==0，选择 pool-num-replicas个桶（全部可用）。  2、如果{num} > 0 && < pool-num-replicas，选择num个桶。  3、如果{num} < 0，选择pool-num-replicas - {num}个桶。 |
| 目的： | 规则的一个组成部分。 |
| 先决条件： | 跟着step take 或step choose |
| 默认值 | step choose firstn 1 type row |

**step chooseleaf firstn {num} type {bucket-type}**

|  |  |
| --- | --- |
| 描述： | 选择一组{bucket-type}的桶并从桶组中的每个桶的子树中选择一个叶节点。集合中桶的数量通常是池中副本的数量（例：池大小）。  1、如果{num}==0，选择 pool-num-replicas个桶（全部可用）。  2、如果{num} > 0 && < pool-num-replicas，选择num个桶。  3、如果{num} < 0，选择pool-num-replicas - {num}个桶。 |
| 目的： | 规则的一个组成部分。用法不需要​​使用两个步骤来选择设备。 |
| 先决条件： | 跟着step take 或step choose |
| 例： | step chooseleaf firstn 0 type row |

**step emit**

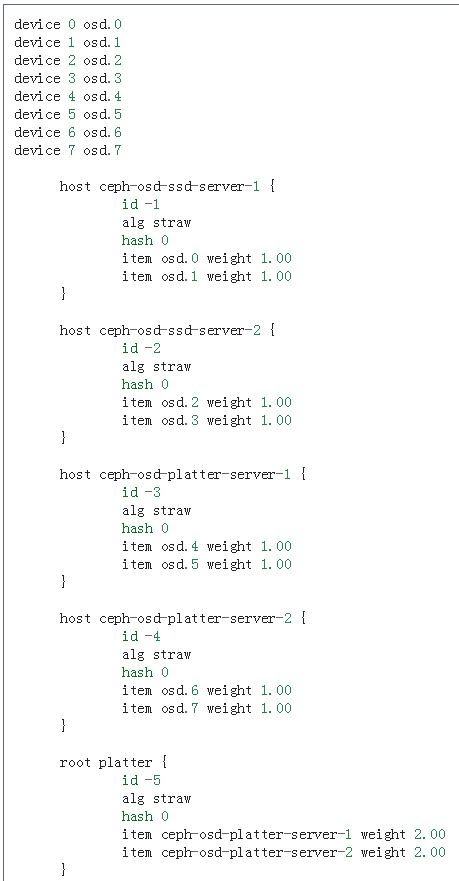
|  |  |
| --- | --- |
| 描述： | 输出当前值并清空堆栈。通常在规则的末尾使用，但也可以用来从相同规则的不同树中选取。 |
| 目的： | 规则的一个组成部分。 |
| 先决条件： | 跟着step choose |
| 例： | step emit |

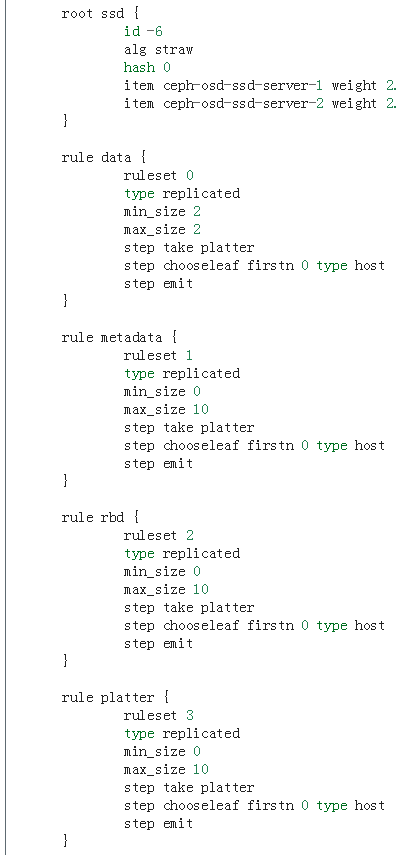
**重要**

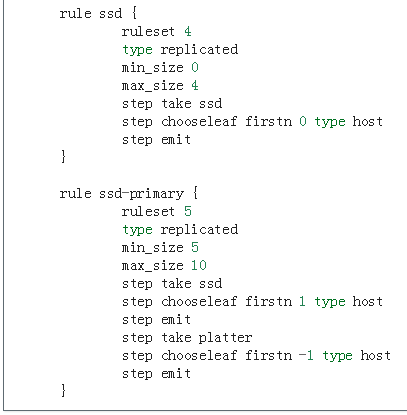
给定的CRUSH规则可以分配给多个池，但单个池不可能有多个CRUSH规则。

# 15. 在不同的OSDS上放置不同的池（★英文文档新增）

假设您希望将大多数池默认为由大型硬盘驱动器支持的OSD，但将一些池映射到由快速固态驱动器（SSD）支持的OSD。在同一个CRUSH映射中可以有多个独立的CRUSH层次结构。使用两个不同的根节点（一个用于硬盘（例如“root platter”）和一个用于SSD（例如“root ssd”））定义两个层次结构，如下所示：







然后您可以通过以下方式设置将池来使用SSD规则：

ceph osd pool set <poolname> crush\_rule ssd

同样，使用该ssd-primary规则将导致池中的每个放置组都以SSD作为主要位置，并将盘片作为副本放置。

## 15.1 调整CRUSH，困难的方式

如果可以确保所有客户端都运行最新的代码，则可以通过提取CRUSH映射、修改值并将其重新注入到集群中来调整可调参数。

1、提取最新的CRUSH图：

ceph osd getcrushmap -o /tmp/crush

2、调整可调参数。这些值似乎为我们测试的大型和小型集群提供了最佳的行为。您需要额外指定--enable-unsafe-tunables参数才能 crushtool使其工作。请谨慎使用此选项。

crushtool -i /tmp/crush --set-choose-local-tries 0 --set-choose-local-fallback-tries 0 --set-choose-total-tries 50 -o /tmp/crush.new

3、重新修改映射：

ceph osd setcrushmap -i /tmp/crush.new

## 15.2 传统值（LEGACY VALUES）

作为参考，CRUSH可调参数的传统值可以使用以下设置：

crushtool -i /tmp/crush --set-choose-local-tries 2 --set-choose-local-fallback-tries 5 --set-choose-total-tries 19 --set-chooseleaf-descend-once 0 --set-chooseleaf-vary-r 0 -o /tmp/crush.legacy

同样，特殊--enable-unsafe-tunables选项是必需的。此外，如上所述，ceph-osd在恢复到传统值之后，要小心运行守护进程的旧版本， 因为特征位没有被完全执行。

# 16. 增加/删除 OSD

## 16.1 增加OSD

在 Ceph 里，一个 OSD 一般是一个 ceph-osd 守护进程，它运行在硬盘之上，如果你有多个硬盘，可以给每个硬盘启动一个 ceph-osd 守护进程。

通常，你应该监控集群容量，看是否达到了容量上限，因为达到了它的 near full 比率后，要增加一或多个 OSD 来扩容。

**Warning**

不要等空间满了再增加 OSD ，空间使用率达到 near full 比率后， OSD 失败可能导致集群空间占满。

### 16.1.1 部署硬件

如果你通过增加主机来增加 OSD ，关于 OSD 服务器硬件的配置请参见[硬件推荐](http://docs.ceph.org.cn/start/hardware-recommendations)。要把一台 OSD 主机加入到集群，首先要安装最新版的 Linux ，而且存储硬盘要做好必要的准备，详情参见[文件系统推荐](http://docs.ceph.org.cn/rados/configuration/filesystem-recommendations)。

把 OSD 主机添加到集群机架上，连接好网络、确保网络通畅。详情见[网络配置参考](http://docs.ceph.org.cn/rados/configuration/network-config-ref)。

### 16.1.2 安装必要软件

在手动部署的集群里，你必须手动安装 Ceph 软件包，详情见[安装 Ceph （手动）](http://docs.ceph.org.cn/install)。你应该配置一个无密码登录 SSH 的用户，且他有 root 权限。

### 16.1.3 增加 OSD （手动）

此过程要设置一个 ceph-osd 守护进程，让它使用一个硬盘，且让集群把数据发布到 OSD 。如果一台主机有多个硬盘，可以重复此过程，把每个硬盘配置为一个 OSD 。

要添加 OSD ，要依次创建数据目录、把硬盘挂载到目录、把 OSD 加入集群、然后把它加入 CRUSH 图。

往 CRUSH 图里添加 OSD 时建议设置权重，硬盘容量每年增长 40% ，所以较新的 OSD 主机拥有更大的空间（即它们可以有更大的权重）。

**Tip**

Ceph 喜欢统一的硬件，与存储池无关。如果你要新增容量不一的驱动器，还需调整它们的权重。但是，为实现最佳性能，CRUSH 的分级结构最好按类型、容量定义。

**1、创建 OSD 。**如果未指定 UUID ， OSD 启动时会自动生成一个。下列命令会输出 OSD 号，后续步骤你会用到。

ceph osd create [{uuid} [{id}]]

如果指定了可选参数 {id} ，那么它将作为 OSD id 。要注意，如果此数字已使用，此命令会出错。

**Warning**

一般来说，我们不建议指定 {id} 。因为 ID 是按照数组分配的，跳过一些依然会浪费内存；尤其是跳过太多、或者集群很大时，会更明显。若未指定 {id} ，将用最小可用数字。

**2、在新 OSD 主机上创建默认目录。**

ssh {new-osd-host}

sudo mkdir /var/lib/ceph/osd/ceph-{osd-number}

**3、如果准备用于 OSD 的是单独的而非系统盘，先把它挂载到刚创建的目录下：**

ssh {new-osd-host}

sudo mkfs -t {fstype} /dev/{drive}

sudo mount -o user\_xattr /dev/{hdd} /var/lib/ceph/osd/ceph-{osd-number}

**4、初始化 OSD 数据目录。**

ssh {new-osd-host}

ceph-osd -i {osd-num} --mkfs --mkkey

运行 ceph-osd 时目录必须是空的。

**5、注册 OSD 认证密钥**， ceph-{osd-num} 路径里的 ceph 值应该是 $cluster-$id ，如果你的集群名字不是 ceph ，那就用改过的名字。

ceph auth add osd.{osd-num} osd 'allow \*' mon 'allow rwx' -i /var/lib/ceph/osd/ceph-{osd-num}/keyring

把 OSD 加入 CRUSH 图，这样它才开始收数据。用 ceph osd crush add 命令把 OSD 加入 CRUSH 分级结构的合适位置。如果你指定了不止一个桶，此命令会把它加入你所指定的桶中最具体的一个，*并且*把此桶挪到你指定的其它桶之内。**重要：**如果你只指定了 root 桶，此命令会把 OSD 直接挂到 root 下面，但是 CRUSH 规则期望它位于主机内。

若用的是 v0.48 版，执行下列命令：

ceph osd crush add {id} {name} {weight} [{bucket-type}={bucket-name} ...]

若用的是 v0.56 及更高版，执行下列命令：

ceph osd crush add {id-or-name} {weight} [{bucket-type}={bucket-name} ...]

你也可以反编译 CRUSH 图、把 OSD 加入设备列表、以桶的形式加入主机（如果它没在 CRUSH 图里）、以条目形式把设备加入主机、分配权重、重编译并应用它。详情参见[增加/移动 OSD](http://docs.ceph.org.cn/rados/operations/crush-map#addosd) 。

**Argonaut 0.48 版最佳实践**

为降低对用户 I/O 性能的影响，加入 CRUSH 图时应该把 OSD 的初始权重设为 0 ，然后每次增大一点、逐步增大 CRUSH 权重。例如每次增加 0.2 ：

ceph osd crush reweight {osd-id} .2

迁移完成前，可以依次把权重重置为 0.4 、 0.6 等等，直到达到期望权重。

为降低 OSD 失败的影响，你可以设置：

mon osd down out interval = 0

它防止挂了的 OSD 自动被标记为 out ，然后逐步降低其权重：

ceph osd reweight {osd-num} .8

还是等着集群完成数据迁移，然后再次调整权重，直到权重为 0 。注意，这会阻止集群在发生故障时自动重复制数据，所以要确保监控的及时性，以便管理员迅速介入。

注意，以上经验在 Bobtail 及后续版本已不再必要。

### 16.1.4 替换一个OSD（★英文文档新增）

当磁盘出现故障时，或者如果一个admnistrator想要重新配置一个新的后端OSD，例如，从FileStore切换到BlueStore，需要替换OSD。与[删除OSD](http://docs.ceph.com/docs/master/rados/operations/add-or-rm-osds/#removing-the-osd)（16.2.4）不同的[是](http://docs.ceph.com/docs/master/rados/operations/add-or-rm-osds/#removing-the-osd)，OSD的ID和CRUSH映射条目在OSD被删除后需要保持不变。

1、首先销毁OSD：

ceph osd destroy {id} --yes-i-really-mean-it

2、如果之前有将磁盘用于其他目的，则为新的OSD清除该磁盘。若磁盘为新的，则不必要进行此步操作：

ceph-volume lvm zap /dev/sdX

3、使用先前销毁的OSD ID准备要更换的磁盘：

ceph-volume lvm prepare --osd-id {id} --data /dev/sdX

4、并激活OSD：

ceph-volume lvm activate {id} {fsid}

或者，不准备和激活，而是可以在一次呼叫中重新创建设备，如：

ceph - volume lvm create - osd - id { id } - data / dev / sdX

### 16.1.5 启动 OSD

把 OSD 加入 Ceph 后， OSD 就在配置里了。然而它还没运行，它现在的状态为 down 且 out 。你必须先启动 OSD 它才能收数据。可以用管理主机上的 service ceph 、或从 OSD 所在主机启动。

在Ubuntu Trusty上使用Upstart。

sudo start ceph-osd id={osd-num}

在其他发行版上使用systemd。

sudo systemctl start ceph-osd@{osd-num}

一旦你启动了 OSD ，其状态就变成了 up 且 in 。

### 16.1.6 观察数据迁移

把新 OSD 加入 CRUSH 图后， Ceph 会重新均衡服务器，一些归置组会迁移到新 OSD 里，你可以用 [ceph](http://docs.ceph.org.cn/rados/operations/monitoring) 命令观察此过程。

ceph -w

你会看到归置组状态从 active+clean 变为 active, some degraded objects （有降级的对象)、且迁移完成后回到 active+clean 状态。（ Ctrl-c 退出）

## 16.2 删除OSD（手动）

要想缩减集群尺寸或替换硬件，可在运行时删除 OSD 。在 Ceph 里，一个 OSD 通常是一台主机上的一个 ceph-osd 守护进程、它运行在一个硬盘之上。如果一台主机上有多个数据盘，你得挨个删除其对应 ceph-osd 。通常，操作前应该检查集群容量，看是否快达到上限了，确保删除 OSD 后不会使集群达到 near full 比率。

**Warning**

删除 OSD 时不要让集群达到 full ratio 值，删除 OSD 可能导致集群达到或超过 full ratio 值。

### 16.2.1 把 OSD 踢出集群

删除 OSD 前，它通常是 up 且 in 的，要先把它踢出集群，以使 Ceph 启动重新均衡、把数据拷贝到其他 OSD 。

ceph osd out {osd-num}

### 16.2.2 观察数据迁移

一旦把 OSD 踢出（ out ）集群， Ceph 就会开始重新均衡集群、把归置组迁出将删除的 OSD 。你可以用 [ceph](http://docs.ceph.org.cn/rados/operations/monitoring) 工具观察此过程。

ceph -w

你会看到归置组状态从 active+clean 变为 active, some degraded objects 、迁移完成后最终回到 active+clean 状态。（ Ctrl-c 中止）

**Note**

有时候，（通常是只有几台主机的“小”集群，比如小型测试集群）拿出（ out ）某个 OSD 可能会使 CRUSH 进入临界状态，这时某些 PG 一直卡在 active+remapped 状态。如果遇到了这种情况，你应该把此 OSD 标记为 in ，用这个命令：

``ceph osd in {osd-num}``

等回到最初的状态后，把它的权重设置为 0 ，而不是标记为 out ，用此命令：

``ceph osd crush reweight osd.{osd-num} 0``

执行后，你可以观察数据迁移过程，应该可以正常结束。把某一 OSD 标记为 out 和权重改为 0 的区别在于，前者，包含此 OSD 的桶、其权重没变；而后一种情况下，桶的权重变了（降低了此 OSD 的权重）。某些情况下， reweight 命令更适合“小”集群。

### 16.2.3 停止 OSD

把 OSD 踢出集群后，它可能仍在运行，就是说其状态为 up 且 out 。删除前要先停止 OSD 进程。

ssh {osd-host}

sudo systemctl stop ceph-osd@{osd-num}

停止 OSD 后，状态变为 down 。

### 16.2.4 删除 OSD（★英文文档改动）

此过程从集群映射中删除OSD，删除其身份验证密钥，从OSD映射中删除OSD，并从ceph.conf文件中删除OSD 。如果您的主机有多个驱动器，则可能需要重复此步骤，为每个驱动器删除一个OSD。

1、让集群首先忘记OSD。这一步从CRUSH映射中删除OSD，删除它的验证密钥。它也从OSD映射中删除。请注意，[清除子命令](http://docs.ceph.com/man/8/ceph#osd)是在Luminous中引入的，对于旧版本，请参阅下面的内容。

ceph osd purge {id} --yes-i-really-mean-it

2、远程到保存集群ceph.conf文件的主副本的主机 。

ssh {admin-host}

cd /etc/ceph

vim ceph.conf

3、从您的ceph.conf文件中删除OSD条目（如果存在）。

[osd.1]

host = {hostname}

4、从保存集群ceph.conf文件的主副本的主机中，将更新的文件复制ceph.conf到/etc/ceph集群中其他主机的目录。

如果您的Ceph集群比Luminous老，而不是使用ceph osd purge，您需要手动执行此步骤：

1. 删除 CRUSH 图的对应 OSD 条目，它就不再接收数据了。你也可以反编译 CRUSH 图、删除 device 列表条目、删除对应的 host 桶条目或删除 host 桶（如果它在 CRUSH 图里，而且你想删除主机），重编译 CRUSH 图并应用它。详情参见[删除 OSD](http://docs.ceph.org.cn/rados/operations/crush-map#removeosd) 。

ceph osd crush remove {name}

1. 删除 OSD 认证密钥：

ceph auth del osd.{osd-num}

ceph-{osd-num} 路径里的 ceph 值是 $cluster-$id ，如果集群名字不是 ceph ，这里要更改。

1. 删除 OSD 。

ceph osd rm {osd-num}

#for example

ceph osd rm 1

# 17. 增加/删除监视器

## 17.1 增加监视器

Ceph 监视器是轻量级进程，它维护着集群运行图的主副本。一个集群可以只有一个监视器，我们推荐生产环境至少 3 个监视器。 Ceph 使用 [Paxos](http://en.wikipedia.org/wiki/Paxos_(computer_science))算法的一个变种对各种图、以及其它对集群来说至关重要的信息达成共识。由于 Paxos 算法天生要求大部分监视器在运行，以形成法定人数（并因此达成共识）。

运行奇数显示器是明智的，但不是强制性的。奇数个显示器比偶数显示器具有更高的故障恢复能力。例如，在2台显示器部署中，为了维持法定人数，不能容忍任何故障; 有3个监视器，可以容忍一个故障; 在4台显示器部署中，可以容忍一个故障; 有5个监视器，可以容忍两个故障。这就是为什么奇数是可取的。总而言之，Ceph需要大多数显示器运行（并且能够相互通信），但是大多数情况下可以使用单个显示器或2个显示器中的2个，3个中的2个，4个中的3个等等来实现。

对于多节点Ceph集群的初始部署，建议部署三台监视器，如果存在多于三个的有效需求，则一次增加两台监视器。

正因为监视器是轻量级的，所以有可能在作为 OSD 的主机上同时运行它；然而，我们推荐运行于单独主机，因为与内核的 fsync 问题会影响性能。

### 17.1.1 部署硬件

如果你增加新监视器时要新增一台主机，关于其最低硬件配置请参见[硬件推荐](http://docs.ceph.org.cn/start/hardware-recommendations)。要增加一个监视器主机，首先要安装最新版的 Linux （如 Ubuntu 12.04 或者 RHEL 7 ）。

把监视器主机安装上架，连通网络。

### 17.1.2 安装必要软件

手动部署的集群， Ceph 软件包必须手动装，详情参见[安装软件包](http://docs.ceph.org.cn/install/install-storage-cluster)。应该配置一个用户，使之可以无密码登录 SSH 、且有 root 权限。

### 17.1.3 增加监视器（手动）

本步骤创建 ceph-mon 数据目录、获取监视器运行图和监视器密钥环、增加一个 ceph-mon 守护进程。如果这导致只有 2 个监视器守护进程，你可以重演此步骤来增加一或多个监视器，直到你拥有足够多 ceph-mon 达到法定人数。

现在该指定监视器的标识号了。传统上，监视器曾用单个字母（ a 、 b 、 c ……）命名，但你可以指定任何形式。在本文档里，要记住 {mon-id} 应该是你所选的标识号，不包含 mon. 前缀，如在 mon.a 中，其 {mon-id} 是 a 。

1. 在新监视器主机上创建默认目录：

ssh {new-mon-host}

sudo mkdir /var/lib/ceph/mon/ceph-{mon-id}

1. 创建临时目录 {tmp} ，用以保存此过程中用到的文件。此目录要不同于前面步骤创建的监视器数据目录，且完成后可删除。

mkdir {tmp}

1. 获取监视器密钥环， {tmp} 是密钥环文件保存路径、 {filename} 是包含密钥的文件名。

ceph auth get mon. -o {tmp}/{key-filename}

1. 获取监视器运行图， {tmp} 是获取到的监视器运行图、 {filename} 是包含监视器运行图的文件名。

ceph mon getmap -o {tmp}/{map-filename}

1. 准备第一步创建的监视器数据目录。必须指定监视器运行图路径，这样才能获得监视器法定人数和它们 fsid 的信息；还要指定监视器密钥环路径。

sudo ceph-mon -i {mon-id} --mkfs --monmap {tmp}/{map-filename} --keyring {tmp}/{key-filename}

1. 启动新监视器，它会自动加入机器。守护进程需知道绑定到哪个地址，通过 --public-addr {ip:port} 或在 ceph.conf 里的相应段设置 mon addr 可以指定。

ceph-mon -i {mon-id} --public-addr {ip:port}

## 17.2 删除监视器

### 17.2.1 删除监视器（手动）

本步骤从集群删除 ceph-mon 守护进程，如果此步骤导致只剩 2 个监视器了，你得增加或删除一个监视器，直到凑足法定人数所必需的 ceph-mon 数。

1. 停止监视器。

service ceph -a stop mon.{mon-id}

1. 从集群删除监视器。

ceph mon remove {mon-id}

1. 删除 ceph.conf 对应条目。

### 17.2.2 从不健康集群删除监视器

本步骤从不健康集群删除 ceph-mon ，例如集群内的监视器不能形成法定人数。

1. 停止所有监视器主机上的所有 ceph-mon 守护进程。

ssh {mon-host}

service ceph stop mon || stop ceph-mon-all

# 要在所有监视器主机上执行

1. 找出一个活着的监视器并登录其所在主机。

ssh {mon-host}

1. 提取 monmap 副本。

ceph-mon -i {mon-id} --extract-monmap {map-path}

# 多数情况下都是：

ceph-mon -i `hostname` --extract-monmap /tmp/monmap

1. 删除不保留或有问题的监视器。例如，如果你有 3 个监视器 mon.a 、 mon.b 和 mon.c ，其中仅保留 mon.a ，按如下步骤：

monmaptool {map-path} --rm {mon-id}

# 例如

monmaptool /tmp/monmap --rm b

monmaptool /tmp/monmap --rm c

1. 把去除过监视器后剩下的运行图注入存活的监视器。比如，用下列命令把一张运行图注入 mon.a 监视器：

ceph-mon -i {mon-id} --inject-monmap {map-path}

# for example,

ceph-mon -i a --inject-monmap /tmp/monmap

1. 只启动保留下来的监视器。
2. 确认这些监视器形成了法定人数（ ceph -s ）。
3. 你也许得把已删除监视器的数据目录 /var/lib/ceph/mon 备份到安全位置，如果您对其余监视器很有信心、或者有足够的冗余，也可以删除。

## 17.3 更改监视器的 IP 地址

**Important**

现有监视器不应该更改其 IP 地址。

监视器是 Ceph 集群的关键组件，它们要维护一个法定人数，这样整个系统才能正常工作。要确立法定人数，监视器得互相发现对方， Ceph 对监视器的发现要求严格。

Ceph 客户端及其它 Ceph 守护进程用 ceph.conf 发现监视器，然而，监视器之间用监视器运行图发现对方，而非 ceph.conf 。例如，你看过的[增加监视器（手动）](http://docs.ceph.org.cn/rados/operations/add-or-rm-mons/#id7)，会发现创建新监视器时得获取当前集群的 monmap ，因为它是 ceph-mon -i {mon-id} \ --mkfs 命令的必要参数。下面几段解释了 Ceph 监视器的一致性要求，和几种改 IP 的安全方法。

### 17.3.1 一致性要求

监视器发现集群内的其它监视器时总是参照 monmap 的本地副本，用 monmap 而非 ceph.conf 可避免因配置错误（例如在 ceph.conf 指定监视器地址或端口时拼写错误）而损坏集群。正因为监视器用 monmaps 相互发现、且共享于客户端和其它 Ceph 守护进程间，所以 monmap 给监视器提供了苛刻的一致性保证。

苛刻的一致性要求也适用于 monmap 的更新，因为任何有关监视器的更新、 monmap 的更改都通过名为 [Paxos](http://en.wikipedia.org/wiki/Paxos_(computer_science)) 的分布式一致性算法运行。为保证法定人数里的所有监视器都持有同版本 monmap ，所有监视器都要赞成 monmap 的每一次更新，像增加、删除监视器。 monmap 的更新是增量的，这样监视器都有最近商定的版本以及一系列之前版本，这样可使一个有较老 monmap 的监视器赶上集群当前的状态。

如果监视器通过 Ceph 配置文件而非 monmap 相互发现，就会引进额外风险，因为 Ceph 配置文件不会自动更新和发布。监视器有可能用了较老的 ceph.conf 而导致不能识别某监视器、掉出法定人数、或者发展为一种 [Paxos](http://en.wikipedia.org/wiki/Paxos_(computer_science)) 不能精确确定当前系统状态的情形。总之，更改现有监视器的 IP 地址必须慎之又慎。

### 17.3.2 更改监视器 IP 地址（正确方法）

仅仅在 ceph.conf 里更改监视器的 IP 不足以让集群内的其它监视器接受更新。要更改一个监视器的 IP 地址，你必须以先以想用的 IP 地址增加一个监视器（见17.1.3 [增加监视器（手动）](http://docs.ceph.org.cn/rados/operations/add-or-rm-mons/#id7)），确保新监视器成功加入法定人数，然后删除用旧 IP 的监视器，最后更新 ceph.conf 以确保客户端和其它守护进程得知新监视器的 IP 地址。

例如，我们假设有 3 个监视器，如下：

[mon.a]

host = host01

addr = 10.0.0.1:6789

[mon.b]

host = host02

addr = 10.0.0.2:6789

[mon.c]

host = host03

addr = 10.0.0.3:6789

要把 host04 上 mon.c 的 IP 改为 10.0.0.4 ，按照17.1.3[增加监视器（手动）](http://docs.ceph.org.cn/rados/operations/add-or-rm-mons/#id7)里的步骤增加一个新监视器 mon.d ，确认它运行正常后再删除 mon.c ，否则会破坏法定人数；最后依照17.1.3[删除监视器（手动）](http://docs.ceph.org.cn/rados/operations/add-or-rm-mons/#id10)删除 mon.c 。 3 个监视器都要更改的话，每次都要重复一次。

### 17.3.3 更改监视器 IP 地址（凌乱方法）

可能有时候监视器不得不挪到不同的网络、数据中心的不同位置、甚至不同的数据中心，这是可能的，但过程有点惊险。

在这种情形下，一种方法是用所有监视器的新 IP 地址生成新 monmap ，并注入到集群内的所有监视器。对大多数用户来说，这并不简单，好在它不常见。再次重申，监视器不应该更改 IP 地址。

以前面的监视器配置为例，假设你想把所有监视器的 IP 从 10.0.0.x 改为 10.1.0.x ，并且两个网络互不相通，步骤如下：

1. 获取监视器运行图，其中 {tmp} 是所获取的运行图路径， {filename} 是监视器运行图的文件名。

ceph mon getmap -o {tmp}/{filename}

1. 下面是一个 monmap 内容示例：

$ monmaptool --print {tmp}/{filename}

monmaptool: monmap file {tmp}/{filename}

epoch 1

fsid 224e376d-c5fe-4504-96bb-ea6332a19e61

last\_changed 2012-12-17 02:46:41.591248

created 2012-12-17 02:46:41.591248

0: 10.0.0.1:6789/0 mon.a

1: 10.0.0.2:6789/0 mon.b

2: 10.0.0.3:6789/0 mon.c

1. 删除现有监视器。

$ monmaptool --rm a --rm b --rm c {tmp}/{filename}

monmaptool: monmap file {tmp}/{filename}

monmaptool: removing a

monmaptool: removing b

monmaptool: removing c

monmaptool: writing epoch 1 to {tmp}/{filename} (0 monitors)

1. 添加新监视器位置。

$ monmaptool --add a 10.1.0.1:6789 --add b 10.1.0.2:6789 --add c 10.1.0.3:6789 {tmp}/{filename}

monmaptool: monmap file {tmp}/{filename}

monmaptool: writing epoch 1 to {tmp}/{filename} (3 monitors)

1. 检查新内容。

$ monmaptool --print {tmp}/{filename}

monmaptool: monmap file {tmp}/{filename}

epoch 1

fsid 224e376d-c5fe-4504-96bb-ea6332a19e61

last\_changed 2012-12-17 02:46:41.591248

created 2012-12-17 02:46:41.591248

0: 10.1.0.1:6789/0 mon.a

1: 10.1.0.2:6789/0 mon.b

2: 10.1.0.3:6789/0 mon.c

从这里开始，假设监视器（及存储）已经被安装到了新位置。下一步把修正的 monmap 散播到新监视器，并且注入每个监视器。

1. 首先，停止所有监视器，注入必须在守护进程停止时进行。
2. 注入 monmap 。

ceph-mon -i {mon-id} --inject-monmap {tmp}/{filename}

1. 重启监视器。

到这里，到新位置的迁移完成，监视器应该照常运行了。

# 18. BLUESTORE迁移（★英文文档新增）

每个OSD可以运行BlueStore或FileStore，并且一个Ceph集群可以包含两者的混合。以前部署FileStore的用户可能需要转换到BlueStore，以便利用改进的性能和稳健性。有几个策略来做这样的转变。

一个单独的OSD不能孤立地转换，然而：BlueStore和FileStore实在太不一样了。“转换”将依赖于集群的正常复制和修复支持，或者将OSD内容从旧的（FileStore）设备复制到新的（BlueStore）设备的工具和策略。

## 18.1 部署带BLUESTORE新OSD

任何新的OSD（例如，当集群扩展时）都可以使用BlueStore进行部署。这是默认行为，所以不需要进行特定的更改。

同样，更换故障驱动器后重新配置的任何OSD都可以使用BlueStore。

## 18.2 转换现有的OSD

### 18.2.1 标记并替换

最简单的方法是依次标记每个设备，等待数据在集群中重新复制，重新配置OSD，然后再次将其标记回来。自动化简单易行。但是，这需要比必须的更多数据迁移，因此不是最佳选择。

1、识别一个FileStore OSD来取代：

ID=<osd-id-number>

DEVICE=<disk-device>

您可以判断给定的OSD是FileStore还是BlueStore：

ceph osd metadata $ID | grep osd\_objectstore

您可以通过以下方式获得当前的filestore vs bluestore的数量：

ceph osd count-metadata osd\_objectstore

2、将filestore OSD标记出来：

ceph osd out $ID

3、等待数据从OSD中移出：

while ! ceph osd safe-to-destroy $ID ; sleep 60 ; done

4、停止OSD：

systemctl kill ceph-osd@$ID

5、记下这个OSD使用哪个设备：

mount | grep /var/lib/ceph/osd/ceph-$ID

6、卸载OSD：

umount /var/lib/ceph/osd/ceph-$ID

7、销毁OSD数据。要*万分小心*，因为这会破坏设备的内容；确保设备上的数据是不需要的（例如，集群是健康的），然后再继续。

ceph-volume lvm zap $DEVICE

8、告诉集群OSD已经被破坏（并且可以用相同的ID重新配置新的OSD）：

ceph osd destroy $ID --yes-i-really-mean-it

9、用相同的OSD ID重新配置一个BlueStore OSD。这要求您根据上面看到的内容来确定要擦除的设备。小心！

ceph-volume create --bluestore --data $DEVICE --osd-id $ID

10、重复。

您可以允许重新填充替换的OSD，以便在下一个OSD消失的同时进行，或者只要您确保集群完全干净（所有数据都具有所有副本），就可以并行执行同一程序任何OSD。如果不这样做，将会减少数据的冗余，并增加（甚至可能导致）数据丢失的风险。

优点：

* 简单。
* 可以在逐个设备的基础上完成。
* 不需要备用设备或主机。

缺点：

* 通过网络将数据复制两次：一次到集群中的其他OSD（保持所需数量的复制品），然后再次回到重新配置的BlueStore OSD。

### 18.2.2 整个主机替换

如果在集群中有一个备用主机，或者有足够的可用空间将整个主机撤出以便将其用作备用，则可以逐个主机地完成转换，每个存储的数据副本只迁移一次。

首先，你需要有没有数据的空主机。有两种方法可以执行此操作：通过从尚未成为集群一部分的新的空主机开始，或通过从集群中的现有主机卸载数据。

#### 18.2.2.1 使用一个新的空主机

理想情况下，主机应该具有与其他主机大致相同的容量（尽管这不重要）。

NEWHOST=<empty-host-name>

将主机添加到CRUSH层次结构，但不要将其附加到根目录：

ceph osd crush add-bucket $NEWHOST host

确保ceph软件包已安装。

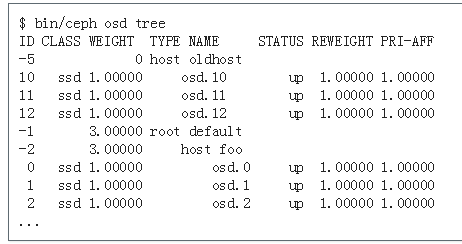
#### 18.2.2.2 使用现有的主机

如果您想要使用已经是集群一部分的现有主机，并且该主机上有足够的可用空间以便可以迁移所有数据，则可以执行以下操作：

OLDHOST=<existing-cluster-host-to-offload>

ceph osd crush unlink $OLDHOST default

其中“default”是CRUSH地图中的直接祖先。（对于未修改配置的较小的集群，这通常是“默认”，但也可能是机架名称）。现在，您应该可以看到OSD树形输出顶部的主机没有父节点：



如果一切正常，请直接跳到下面的“等待数据迁移完成”步骤，然后从那里继续清理旧的OSD。

#### 18.2.2.3 迁移过程

如果您正在使用新的主机，请从第1步开始。对于现有的主机，跳转到下面的步骤＃5。

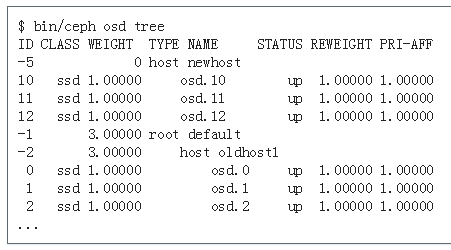
1、为所有设备配置新的BlueStore OSD：

ceph-volume lvm create --bluestore --data /dev/$DEVICE

2、验证OSD加入集群：

ceph osd tree

你应该看到新的主机下面$NEWHOST有所有的OSD，但主机*不*应该嵌套在层次结构中的任何其他节点（如root default）。例如，如果newhost是空的主机，则可能会看到类似如下的内容：



3、确定要转换的第一个目标主机

OLDHOST=<existing-cluster-host-to-convert>

4、将新主机交换到旧主机在集群中的位置：

ceph osd crush swap-bucket $NEWHOST $OLDHOST

此时所有数据$OLDHOST将开始迁移到OSD上$NEWHOST。如果旧主机和新主机的总容量有所不同，那么您可能还会看到一些数据迁移到集群中的其他节点或从其他节点迁移，但只要主机的大小相似，这将是相对较少的数据量。

5、等待数据迁移完成：

while ! ceph osd safe-to-destroy $(ceph osd ls-tree $OLDHOST); do sleep 60 ; done

6、在现在空的$OLDHOST上停止所有旧的OSD ：

ssh $OLDHOST

systemctl kill ceph-osd.target

umount /var/lib/ceph/osd/ceph-\*

7、销毁并清除旧的OSD：

for osd in `ceph osd ls-tree $OLDHOST`; do

ceph osd purge $osd --yes-i-really-mean-it

done

8、擦除旧的OSD设备。这要求您确定要手动擦除哪些设备（请小心！）。对于每个设备：

ceph-volume lvm zap $DEVICE

9、使用现在为空的主机作为新主机，然后重复：

NEWHOST=$OLDHOST

优点：

* 数据仅通过网络复制一次。
* 一次转换整个主机的OSD。
* 可以并行转换多个主机一次。
* 每台主机不需要备用设备。

缺点：

* 需要一个备用主机。
* 整个主机的OSD值将一次性迁移数据。这可能会影响整体集群性能。
* 所有迁移的数据仍然在网络上完成一个完整的跳跃。

### 18.2.3 每个OSD设备拷贝

一个逻辑OSD可以通过使用copy函数来转换ceph-objectstore-tool。这就要求主机有一个免费的设备（或多个设备）来提供一个新的，空的BlueStore OSD。例如，如果集群中的每个主机都有12个OSD，则需要第13个可用设备，以便每个OSD可以在旧设备被回收之前轮流转换，以转换下一个OSD。

注意事项：

* 这个策略要求准备一个空白的BlueStore OSD，而不用分配一个新的OSD ID，这个ceph-volume 工具不支持。更重要的是，dmcrypt的设置与OSD身份密切相关，这意味着这种方法不适用于加密的OSD。
* 该设备必须手动分区。
* 工具没有实施！
* 没有记录！

优点：

* 在转换过程中很少或没有数据通过网络迁移。

缺点：

* 工具没有完全实施。
* 过程没有记录。
* 每个主机必须有一个备用或空的设备。
* 转换期间，OSD处于脱机状态，这意味着新的写入操作只能写入OSD的一个子集。这增加了由于随后的失败而导致的数据丢失的风险。（但是，如果在转换完成之前出现故障，则可以启动原始的FileStore OSD以提供对其原始数据的访问。）

# 19. 命令参考

## 19.1 监视器命令

监视器命令用 ceph 工具发出：

ceph [-m monhost] {command}

命令格式通常是（但不总是）：

ceph {subsystem} {command}

## 19.2 系统命令

### 19.2.1显示集群状态

ceph -s

ceph status

### 19.2.2显示集群状态的运行摘要、及主要事件

ceph -w

### 19.2.3显示监视器法定人数状态，包括哪些监视器参与者、哪个是首领。

ceph quorum\_status

### 19.2.4 查询单个监视器状态，包括是否在法定人数里

ceph [-m monhost] mon\_status

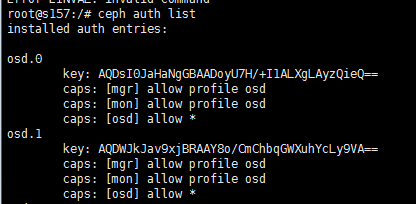
## 19.3 认证子系统

### 19.3.1 要添加一个 OSD 的密钥环

ceph auth add {osd} {--in-file|-i} {path-to-osd-keyring}

### 19.3.2 要列出集群的密钥及其能力

ceph auth list



## 19.4 归置组（pg）子系统

### 19.4.1 显示所有归置组的统计信息

ceph pg dump [--format {format}]

可用输出格式有 plain （默认）和 json 。

### 19.4.2 显示卡在某状态的所有归置组

ceph pg dump\_stuck inactive|unclean|stale|undersized|degraded [--format {format}] [-t|--threshold {seconds}]

--format 可以是 plain （默认）或 json

--threshold 定义了多久算“卡住了”（默认 300 秒）

**Inactive** 归置组不能处理读或写，因为它们在等待数据及时更新的 OSD 回来。

**Unclean** 归置组包含副本数未达期望值的对象，它们应该在恢复中。

**Stale** 归置组处于未知状态——归置组所托付的 OSD 有一阵没向监视器报告了（由 mon osd report timeout 配置）。

### 19.4.3 删除“丢失”对象，或者恢复到其先前状态

可以是前一版本、或如果刚创建就干脆删除。

ceph pg {pgid} mark\_unfound\_lost revert|delete

## 19.5 OSD子系统

### 19.5.1 查询 OSD 子系统状态

ceph osd stat

### 19.5.2 把最新的 OSD 运行图拷贝到一个文件

ceph osd getmap -o file

### 19.5.3 从最新 OSD 运行图拷出 CRUSH 图

ceph osd getcrushmap -o file

前述功能等价于：

ceph osd getmap -o /tmp/osdmap

osdmaptool /tmp/osdmap --export-crush file

### 19.5.4 转储 OSD 运行图

 -f 的可用格式有 plain 和 json ，如未指定 --format 则转储为纯文本。

ceph osd dump [--format {format}]

### 19.5.5 把 OSD 运行图转储为树

每个 OSD 一行、包含权重和状态。

ceph osd tree [--format {format}]

### 19.5.6 找出某对象在哪里或应该在哪里

ceph osd map <pool-name> <object-name>

### 19.5.7 增加或挪动一个新 OSD 条目

要给出 id/name/weight 和位置参数。

ceph osd crush set {id} {weight} [{loc1} [{loc2} ...]]

### 19.5.8 从现有 CRUSH 图删除存在的条目（ OSD ）

ceph osd crush remove {name}

### 19.5.9 从现有 CRUSH 图删除存在的空桶

ceph osd crush remove {bucket-name}

### 19.5.10 把有效的桶从分级结构里的一个位置挪到另一个

ceph osd crush move {id} {loc1} [{loc2} ...]

### 19.5.11 设置 {name} 所指条目的权重为 {weight}

ceph osd crush reweight {name} {weight}

### 19.5.12 把 OSD 标记为丢失

注意：有可能导致永久性数据丢失，慎用！

ceph osd lost {id} [--yes-i-really-mean-it]

### 19.5.13 创建新 OSD 。如果未指定 ID ，有可能的话将自动分配个新 ID 。

ceph osd create [{uuid}]

### 19.5.14 删除指定 OSD 。

ceph osd rm [{id}...]

### 19.5.15 查询 OSD 运行图里的 max\_osd 参数。

ceph osd getmaxosd

### 19.5.16 导入指定 CRUSH 图。

ceph osd setcrushmap -i file

### 19.5.17 设置 OSD 运行图的 max\_osd 参数，扩展存储集群时有必要。

ceph osd setmaxosd

### 19.5.18 把 ID 为 {osd-num} 的 OSD 标记为 down 。

ceph osd down {osd-num}

### 19.5.19 把 OSD {osd-num} 标记为数据分布之外（即不给分配数据）。

ceph osd out {osd-num}

### 19.5.20 把 OSD {osd-num} 标记为数据分布之内（即分配了数据）。

ceph osd in {osd-num}

### 19.5.21 设置或清空 OSD 运行图里的暂停标记。

若设置了，不会有 IO 请求发送到任何 OSD ；用 unpause 清空此标记会导致重发未决的请求。

ceph osd pause

ceph osd unpause

### 19.5.22 把 {osd-num} 的权重设置为 {weight}

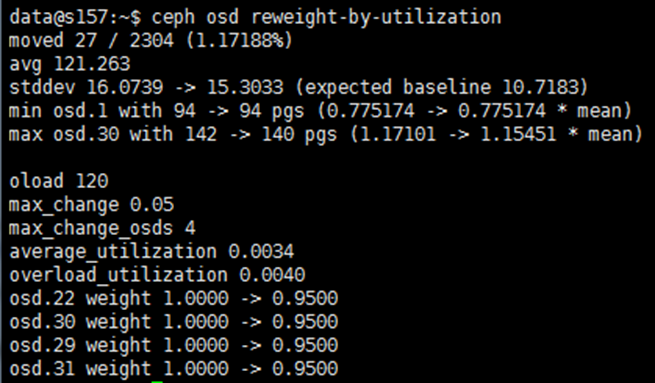
权重相同的两个 OSD 大致会收到相同数量的 I/O 请求、并存储相同数量的数据。 ceph osd reweight 命令可给 OSD 设置一个增益权重，有效值在 0 和 1 之间，它使得 CRUSH 重新归置 (1-weight)一定数量的、本应该放到此处的数据。它不会影响 crush 图里所分配的权重，在 CRUSH 分布算法没能理想地执行时，它可作为一种纠正手段。比如，假设你的某个 OSD 使用率达到了 90% ，但其它的大致都在 50% ，这时你就可以试着下调此权重来补偿它。

ceph osd reweight {osd-num} {weight}

### 19.5.23 重设所有滥用 OSD 的权重

它默认会下调达到平均利用率 120% 的那些OSD ，除非你指定了阀值。

ceph osd reweight-by-utilization [threshold]



### 19.5.24 描述reweight-by-utilization会做什么

ceph osd test-reweight-by-utilization

### 19.5.26 增加、删除黑名单里的地址。

增加地址的时候可以指定有效期，否则有效期为 1 小时。黑名单里的地址不允许连接任何 OSD ，此技术常用于防止滞后的元数据服务器对OSD上的数据做出不好的改变。

这些命令大多只在故障测试时有用，因为黑名单是自动维护的，无需手动干涉。

ceph osd blacklist add ADDRESS[:source\_port] [TIME]

ceph osd blacklist rm ADDRESS[:source\_port]

### 19.5.27 创建/删除存储池快照。

ceph osd pool mksnap {pool-name} {snap-name}

ceph osd pool rmsnap {pool-name} {snap-name}

### 19.5.28 创建/删除/重命名存储池。

ceph osd pool create {pool-name} pg\_num [pgp\_num]

ceph osd pool delete {pool-name} [{pool-name} --yes-i-really-really-mean-it]

ceph osd pool rename {old-name} {new-name}

### 19.5.29 更改存储池设置。

ceph osd pool set {pool-name} {field} {value}

可用的 field 值有：

* size: 设置存储池内数据的副本数；
* pg\_num: 归置组数量；
* pgp\_num: 计算归置组存放的有效数量；
* crush\_ruleset: 用于归置映射的规则号。

### 19.5.30 获取存储池配置值。

ceph osd pool get {pool-name} {field}

可用的 field 值有：

* pg\_num: 归置组数量；
* pgp\_num: 计算归置组存放的有效数量；
* lpg\_num: 本地归置组数量；
* lpgp\_num: 用于存放本地归置组的数量。

### 19.5.31 向 OSD {osd-num} 下达一个洗刷命令，用通配符 \* 把命令下达到所有 OSD 。

ceph osd scrub {osd-num}

### 19.5.32 向 osdN 下达修复命令，用 \* 下达到所有 OSD 。

ceph osd repair N

### 19.5.33 在 osdN 上进行个简单的吞吐量测试

每次写入 BYTES\_PER\_WRITE 、一共写入 TOTAL\_BYTES 。默认以 4MB 增量写入 1GB 。 此压力测试是非破坏性的，不会覆盖已有 OSD 数据，但可能会暂时影响同时访问此 OSD 的客户端性能。

ceph tell osd.N bench [NUMER \_OF\_OBJECTS] [BYTES\_PER\_WRITE]

## 19.6 MDS子系统

### 19.6.1 更改在运行 mds 的参数

ceph tell mds.{mds-id} config set {setting} {value}

例如：

ceph tell mds.0 config set debug\_ms 1

打开了调试消息。

### 19.6.2 显示所有元数据服务器状态。

ceph mds stat

### 19.6.3 活跃 MDS 失败时进行标记，如果有候补此命令会触发故障转移。

ceph mds fail 0

*Todo：ceph mds 子命令缺少文档：set, dump, getmap, stop, setmap*

## 19.7 监视器子系统

### 19.7.1 查看监视器状态

ceph mon stat



末尾的 quorum 列表列出了当前法定人数里的监视器节点。

也可以更直接地获取：

ceph quorum\_status -f json-pretty



如果法定人数未形成，上述命令会一直等待。

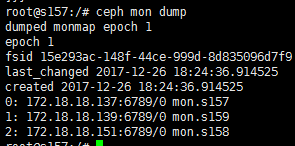
你刚刚连接的监视器的状态（用 -m HOST:PORT 另外指定）：

ceph mon\_status -f json-pretty



监视器状态转储：

ceph mon dump



# 20. 监视器故障排除

## 20.1 使用监视器的管理套接字

通过管理套接字，你可以用 Unix 套接字文件直接与指定守护进程交互。这个文件位于你监视器的 run 目录下，默认配置时它位于 /var/run/ceph/ceph-mon.ID.asok ，但你要是改过就不一定在那里了。如果你在那里没找到它，请看看 ceph.conf 里是否配置了其它路径、或者用下面的命令获取：

ceph-conf --name mon.ID --show-config-value admin\_socket

请牢记，只有在监视器运行时管理套接字才可用。监视器正常关闭时，管理套接字会被删除；如果监视器不运行了、但管理套接字还存在，就说明监视器不是正常关闭的。不管怎样，监视器没在运行，你就不能使用管理套接字， ceph 命令会返回类似 Error111: Connection Refused 的错误消息。

访问管理套接字很简单，就是让 ceph 工具使用 asok 文件。对于 Dumpling 之前的版本，命令是这样的：

ceph --admin-daemon /var/run/ceph/ceph-mon.<id>.asok <command>

对于 Dumpling 及后续版本，你可以用另一个（推荐的）命令：

ceph daemon mon.<id> <command>

ceph 工具的 help 命令会显示管理套接字支持的其它命令。请仔细了解一下 config get 、 config show 、 mon\_status 和 quorum\_status 命令，在排除监视器故障时它们能给你些启发。

## 20.2 理解 MON\_STATUS

当你有一个法定人数，或没有法定人数、但可以通过管理套接字时，可以获取mon\_status。该命令将输出monitor的一系列信息，包括你用quorum\_status能够获得的同样的输出。

以下是一个mon\_status的例子：



在monmap中有3个monitor（a/b/c），quorum由两个monitor组成，c在其中作为peon的角色。哪个monitor不在quorum内？答案是：a。

为什么？

可以看看quorum一栏，有两个monitor在集合中：1和2。这不是Monitor的名字，而是monmap中建立的监控级别（monitor ranks）。这里没有级别为0的monitor，根据monmap可知，mon.a的rank是0。

ranks是如何建立的？

无论何时，当你添加或移除monitor的时候，会遵循一个简单的规则计算：IP:PORT的组合值越小，rank越小。例如127.0.0.1:6789比所有其余的IP:PORT组合，mon.a的等级为0。

## 20.3 常见最监视器的问题

### 20.3.1 有quorum但至少有一个监视器状态是down

发生这种情况时，根据您正在运行的Ceph的版本，您应该看到类似于：

$ ceph health detail

[snip]

mon.a (rank 0) addr 127.0.0.1:6789/0 is down (out of quorum)

**如何解决这个问题？**

首先，确保mon.a正在运行。

其次，确保能够从其他监视器的服务器连接到mon.a的服务器。检查端口。检查所有监视器节点上的iptables，确保不丢弃/拒绝连接。

如果最初的故障排除不能解决您的问题，那么现在是时候深入下去了。

首先，如[使用监视器的管理套接字](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon/#id3)和 [理解MON\_STATUS](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon/#mon-status)中所说，通过管理套接字检查有问题的monitor的MON\_STATUS。

考虑到monitor在quorum之外，其状态应该是 probing，electing 或 synchronizing。如果碰巧是leader或者是peon，那么认为monitor是在quorum中，而剩下的集群肯定不是；或者在我们对monitor进行故障诊断时，monitor进入了quorum，所以请再次检查一下ceph -s，以确认。如果monitor仍未在quorum中，请继续。

**如果状态为probing？**

这意味着monitor仍在寻找其他monitors。每次启动一个monitor时，monitor都将保持此状态一段时间，同时尝试查找monmap中指定的monitor的其余部分。monitor在此状态下花费的时间可能会有所不同。例如，在单monitor集群上时，monitor几乎会瞬间通过探测状态，因为周围没有其他monitor。在多monitor集群上，monitor将保持这种状态，直到他们找到足够的monitor来形成法定人数 - 这意味着如果您有三台monitor中的两台monitor停止工作，剩下的一台monitor将无限期停留在此状态，直到您启动其他monitor之一。

但是，如果您有法定人数，只要能够到达，显示器应该能够快速找到剩余的显示器。如果您的显示器被卡住探测，并且您已经完成了所有通信故障排除，那么显示器尝试将其他显示器连接到错误地址的机会相当可观。mon\_status输出监视器已知的 monmap：检查其他监视器的位置是否属实。如果没有，跳转到 [恢复显示器的损坏的monmap](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon/#recovering-a-monitor-s-broken-monmap) ; 如果他们这样做，那么它可能涉及到严重的时钟脉冲相位差的监测节点之间，你应该参考 [时钟偏移](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon/#id5)首先，如果这不能解决您的问题，那么现在是准备一些日志并与社区联系的时候（请参考[收集所需日志](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon/#id9)了解如何最好地准备日志）。

**如果状态为electing？**

这意味着监视器正在进行选举。这些应该是快速完成，但有时显示器可能会卡住选举。这通常是监视器节点之间时钟偏斜的标志；跳转到 [时钟偏移](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon/#id5)了解更多信息。如果所有时钟都正确同步，那么最好是准备一些日志并联系社区。这不是一个可能会持续下去的状态，除了（*真正的*）老bug之外，除了时钟偏差之外，还有一个明显的原因，那就是为什么会发生这种情况。

**如果状态为synchronizing？**

这意味着监视器正在与集群的其余部分同步以加入法定人数。同步过程与monitor store一样小，因此如果您有大型store，则可能需要一段时间。别担心，应该会尽快完成。

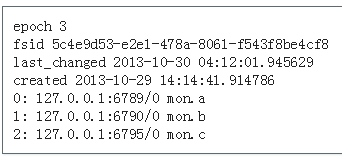
但是，如果您注意到监视器从synchronizing变为 electing，然后返回到synchronizing，那么您确实遇到了一个问题：集群状态正在前进（即生成新映射），因此同步过程不能继续。这在早期的Cuttlefish版本中是一个问题，但是从那以后，同步过程被重构和增强，以避免这种行为。如果这种情况发生在以后的版本让我们知道。并带来一些日志（见[收集所需日志](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon/#id9)）。

**如果状态为leader或peon呢？**

这不应该发生。有一个机会，但是这可能会发生，而且它与时钟脉冲相位差有很大关系-看时钟偏移。如果你没有受到时钟倾斜的困扰，那么请准备好你的日志（见 收集所需日志）并联系我们。

### 20.3.2 恢复监视器损坏的MONMAP

基于monitor的数量，monmap通常看起来是这样的：



但这可能不是你有的。例如，在早期Cuttlefish的版本有一个可能导致你的monmap无效的Bug，完全填满了零。这意味着即使 monmaptool也不能读取它，因为它会发现无法理解纯粹的零。有些时候，你可能会遇到一个监视器，monmap过于严重，因此无法找到剩余的监视器（例如，mon.c是down状态；你添加一个新的监视器mon.d，然后删除mon.a，然后添加一个新的监视器mon.e并删除 mon.b ;最后会从mon.c处得到一个完全不同的 monmap）。

在这种情况下，你有两种可能的解决方案：

**废弃显示器并创建一个新的**

如果您确信您不会丢失该监视器所保存的信息，则只能采取这种方式。你有其他的显示器，而且它们运行的​​很好，这样你的新显示器就能够与其余的显示器同步了。请记住，销毁显示器（如果没有其他内容的副本）可能会导致数据丢失。

**将monmap注入monitor**

通常是最安全的路径。您应该从其余的监视器中获取monmap，并用损坏或丢失的monmap将其注入监视器。

以下是基本步骤：

1、有没有成立的quorum？如果有，从quorum中获取monmap：

$ ceph mon getmap -o /tmp/monmap

2、没有quorum？从另一个监视器直接抓取monmap（这里假定你正在抓取monmap的监视器有ID-FOO并且已经停止）：

$ ceph-mon -i ID -FOO --extract-monmap /tmp/monmap

3、停止你要注入monmap的监视器。

4、注入monmap：

$ ceph-mon -i ID --inject-monmap / tmp / monmap

5、启动显示器

请记住，注入monmaps的功能是一个强大的功能，如果误用，可能会对监视器造成严重破坏，因为它会覆盖监视器保存的最新的monmap。

### 20.3.3 偏移时钟

监视器节点间的时钟偏差可能会严重影响监视器。为了避免这样的问题，你应该在你的监视节点上运行一个时钟同步工具。

**什么是最大容忍时钟歪斜？**

默认情况下，显示器将允许时钟漂移高达0.05 秒。

**我可以增加最大容忍时钟歪斜吗？**

该值是通过配置 mon-clock-drift-allowed选项，虽然你可以，这并不意味着你应该。时钟偏移机制已经到位，因为时钟偏斜的监视器可能不正常工作。作为开发人员和质量保证人员，我们对当前的默认值感到满意，因为在monitor失控之前它会提醒用户。尽管不存在数据丢失的风险，但如果不先对其进行测试，则可能会对监视器的稳定性和整体集群健康性产生不可预见的影响。

**我怎么知道有一个时钟歪斜？**

监视器将以HEALTH\_WARN的形式警告您。ceph health detail应该以下列形式显示：

mon.c addr 10.10.0.1:6789/0 clock skew 0.08235s > max 0.05s (latency 0.0045s)

这意味着mon.c被标记为遭受时钟歪斜。

**如果时钟有偏差，我该怎么办？**

同步你的时钟。运行NTP客户端可能会有帮助。如果您已经在使用它，并且遇到了这类问题，请检查您是否正在使用网络远程的NTP服务器，并考虑在您的网络上托管您自己的NTP服务器。这最后一个选项往往可以减少显示器时钟偏斜的问题。

### 20.3.4 客户端不能连接或挂载

检查防火墙配置。有些系统安装工具把 REJECT 规则加入了 iptables ，它会拒绝除 ssh 以外的所有入栈连接。如果你的监视器主机有这样的 REJECT 规则，别的客户端进来的连接将遇到超时错误而不能挂载。得先找到这条拒绝客户端连入的 iptables 规则，例如，你要找到形似以下的规则：

REJECT all -- anywhere anywhere reject-with icmp-host-prohibited

你也许还要在 Ceph 主机上增加 iptables 规则来放通 Ceph 监视器端口（即默认的 6789 端口）、和 OSD 端口（默认从 6800 到 7300 ）。例如：

iptables -A INPUT -m multiport -p tcp -s {ip-address}/{netmask} --dports 6789,6800:7300 -j ACCEPT

## 20.4 监视存储失败（★英文文档新增）

### 20.4.1 存储损坏（STORE CORRUPTION）的症状

Ceph监视器将[集群映射](http://docs.ceph.com/docs/master/rados/architecture#cluster-map)存储在诸如LevelDB的键/值存储中。如果由于键/值存储损坏而导致监视器失败，则可能会在监视器日志中找到以下错误消息：

Corruption: error in middle of record

或者：

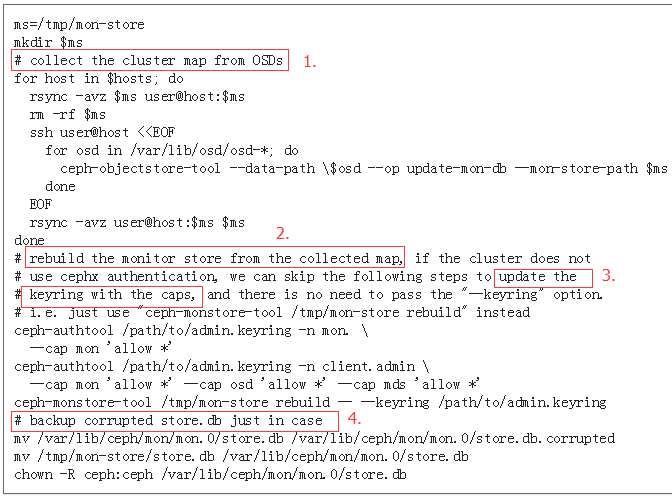
Corruption: 1 missing files; e.g.: /var/lib/ceph/mon/mon.0/store.db/1234567.ldb

### 20.4.2 使用健康的监视器进行恢复

如果有任何遗留问题，我们总是可以用一个新的替换损坏的。开机后，新的监视器将同步健康的监视器，一旦完全同步，它将能够为客户服务。

### 20.4.3 使用OSD进行恢复

但是，如果所有的显示器同时出现故障呢？由于鼓励用户在Ceph集群中至少部署三台监视器，因此同时出现故障的可能性很小。但是，如果数据中心的计划外电源没有正确配置磁盘/ fs设置，则可能会使底层文件系统失效，从而导致所有监视器无法工作。在这种情况下，我们可以使用存储在OSD中的信息来恢复显示器存储。



#### 以上步骤：

1. 从所有OSD主机收集映射；
2. 然后重建存储；
3. 用适当的caps填充密钥环文件中的实体；
4. 使用恢复的副本替换mon.0上损坏的存储。

#### 已知限制

以下信息无法使用上述步骤进行恢复：

* **一些增加的钥匙圈**：使用ceph auth add命令添加的所有OSD钥匙圈都从OSD的副本中恢复。而client.admin钥匙圈是使用ceph-monstore-tool导入的。但是恢复的监视器存储中缺少MDS密钥环和其他密钥环。您可能需要手动重新添加它们。
* **PG设置**：使用ceph pg set\_full\_ratio 和ceph pg set\_nearfull\_ratio配置的full ratio和nearfull ratio设置将丢失。
* **MDS地图**：丢失的MDS映射。

## 20.5 所有尝试都失败了，怎么办？

### 20.5.1 到外面寻求帮助

你可以在IRC的#ceph和＃ceph-devel的OFTC（服务器irc.oftc.net）和ceph-devel@vger.kernel.org和ceph-users@lists.ceph.com找到我们。确保你已经抓住你的日志，并准备好了，如果有人问：互动速度越快，响应时间越短，每个人的时间就越好。

### 20.5.2 收集所需日志

监视日志默认保存在/var/log/ceph/ceph-mon.FOO.log\*中。我们可能想要他们。但是，您的日志可能没有必要的信息。如果在默认位置找不到监视器日志，则可以通过运行来检查应该在哪里：

ceph-conf --name mon.FOO --show-config-value log\_file

日志中的信息量受配置文件强制执行的调试级别的影响。如果你没有强制执行特定的调试级别，那么Ceph将使用默认级别，而你的日志可能不包含重要的信息来追踪你的问题。获取相关信息到日志中的第一步是提高调试级别。在这种情况下，我们会对显示器的信息感兴趣。与其他组件上的情况类似，监视器的不同部分将在不同的子系统上输出其调试信息。

您将不得不提高与您的问题更密切相关的子系统的调试级别。对于不熟悉Ceph故障排除的人来说，这可能不是一件容易的事情。在大多数情况下，在显示器上设置以下选项将足以查明问题的潜在来源：

debug mon = 10

debug ms = 1

如果我们发现这些调试级别还不够，那么我们可能会要求您提高它们甚至定义其他调试子系统以获取信息 - 但至少我们从一些有用的信息开始，而不是大量空的日志很多事情要继续下去。

### 20.5.3 我需要重启监视器来更改调试级别吗？

不，你可以用以下两种方法之一来做：

**有quorum**

将调试选项注入要调试的显示器中：

ceph tell mon.FOO config set debug\_mon 10/10

或一次进入所有显示器：

ceph tell mon.\* config set debug\_mon 10/10

**没有quorum**

使用显示器的管理套接字并直接调整配置选项：

ceph daemon mon.FOO config set debug\_mon 10/10

回到默认值就像使用调试级别1/10重新运行上述命令一样简单。您可以使用管理套接字和以下命令检查您的当前值：

ceph daemon mon.FOO config show

要么：

ceph daemon mon.FOO config get 'OPTION\_NAME'

### 20.5.4 在某个调试级别下重现了问题，然后呢？

理想情况下，您只会将您的日志的相关部分发送给我们。我们意识到找出相应的部分可能不是最简单的任务。因此，如果您提供完整的日志，我们不会拿给您，但应该运用常识。如果你的日志有成千上万行，那么整个过程可能会变得很棘手，特别是如果我们不知道在哪一点上，无论你的问题是什么，都会发生。例如，在复制时，请记住写下当前时间和日期，并根据这些信息提取日志的相关部分。

最后，您应该在IRC的邮件列表上与我们联系，或在[跟踪器](http://tracker.ceph.com/projects/ceph/issues/new)上提交新的问题。

# 21. OSD故障排除

进行 OSD 排障前，先检查一下监视器集群和网络。如果 ceph health 或 ceph -s 返回的是健康状态，这意味着监视器们形成了法定人数。如果你还没监视器法定人数、或者监视器状态错误，要先[解决监视器问题](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-mon)。核实下你的网络，确保它在正常运行，因为网络对 OSD 的运行和性能有显著影响。

## 21.1 收集OSD数据

开始 OSD 排障的第一步最好先收集信息，另外还有[监控 OSD](http://docs.ceph.org.cn/rados/operations/monitoring-osd-pg) 时收集的，如 ceph \ osd tree 。

### 21.1.1 CEPH 日志

如果你没改默认路径，可以在 /var/log/ceph 下找到日志：

ls /var/log/ceph

如果看到的日志还不够详细，可以增大日志级别。请参考[日志记录和调试](http://docs.ceph.org.cn/rados/troubleshooting/log-and-debug)（21），看看如何保证看到大量日志时又不影响集群运行。

### 21.1.2 管理套接字

用管理套接字检索运行时信息。所有 Ceph 套接字列表：

ls /var/run/ceph

然后，执行下例命令显示可用选项，把 {daemon-name} 换成实际的守护进程（如 osd.0 ）：

ceph daemon osd.0 help

另外，你也可以指定一个 {socket-file} （如 /var/run/ceph 下的文件）：

ceph daemon {socket-file} help

和其它手段相比，管理接口允许你：

* 在运行时列出配置
* 列出历史操作
* 列出操作的优先队列状态
* 列出在进行的操作
* 列出性能计数器

### 21.1.3 显示剩余空间

可能是文件系统问题，用 df 显示文件系统剩余空间。

df -h

其它用法见 df --help 。

### 21.1.4 I/O 统计信息

用 iostat 定位 I/O 相关问题。（需要安装iostat）

iostat -x

### 21.1.5 诊断消息

要查看诊断信息，配合 less 、 more 、 grep 或 tail 使用 dmesg ，例如：

dmesg | grep scsi

## 21.2 停止自动重均衡

你得周期性地维护集群的子系统、或解决某个失败域的问题（如一机架）。如果你不想在停机维护 OSD 时让 CRUSH 自动重均衡，提前设置 noout ：

ceph osd set noout

在集群上设置 noout 后，你就可以停机维护失败域内的 OSD 了。

stop ceph-osd id={num}

**Note**

在定位同一故障域内的问题时，停机 OSD 内的归置组状态会变为 degraded 。

维护结束后，重启OSD。

start ceph-osd id={num}

最后，解除 noout 标志。

ceph osd unset noout

## 21.3 OSD 没运行

通常情况下，简单地重启 ceph-osd 进程就可以重回集群并恢复。

### 21.3.1 OSD 起不来

如果你重启了集群，但其中一个 OSD 起不来，依次检查：

* **配置文件：** 如果你新装的 OSD 不能启动，检查下配置文件，确保它合爻性（比如 host 而非 hostname ，等等）；
* **检查路径：** 检查你配置的路径，以及它们自己的数据和日志路径。如果你分离了 OSD 数据和日志、而配置文件和实际挂载点存在出入，启动 OSD 时就可能遇挫。如果你想把日志存储于一个块设备，应该为日志硬盘分区并为各 OSD 分别分配一分区。
* **检查最大线程数：** 如果你的节点有很多 OSD ，也许就会触碰到默认的最大线程数限制（如通常是 32k 个），尤其是在恢复期间。你可以用 sysctl 增大线程数，把最大线程数更改为支持的最大值（即 4194303 ），看看是否有用。例如：

sysctl -w kernel.pid\_max=4194303

如果增大最大线程数解决了这个问题，你可以把此配置 kernel.pid\_max 写入配置文件 /etc/sysctl.conf，使之永久生效，例如：

kernel.pid\_max = 4194303

* **内核版本：** 确认你在用的内核版本以及所用的发布版。 Ceph 默认依赖一些第三方工具，这些工具可能有缺陷或者与特定发布版和/或内核版本冲突（如 Google perftool ）。检查下[操作系统推荐](http://docs.ceph.org.cn/install/os-recommendations)确保你已经解决了内核相关的问题。
* **段错误：** 如果有了段错误，提高日志级别（如果还没提高），再试试。如果重现了，联系 ceph-devel 并提供你的配置文件、显示器输出和日志文件内容。

### 21.3.2 OSD 失败

ceph-osd 挂掉时，监视器可通过活着的 ceph-osd 了解到此情况，且通过 ceph health 命令报告：

ceph health

HEALTH\_WARN 1/3 in osds are down

而且，有 ceph-osd 进程标记为 in 且 down 的时候，你会得到警告，你可以用下面的命令得知哪个 ceph-osd 进程挂了：

ceph health detail

HEALTH\_WARN 1/3 in osds are down

osd.0 is down since epoch 23, last address 192.168.106.220:6800/11080

如果有个硬盘失败或其它错误使 ceph-osd 不能正常运行或重启，一条错误信息将会出现在日志文件 /var/log/ceph/ 里。

如果守护进程因心跳失败、或者底层文件系统无响应而停止，查看 dmesg 获取硬盘或者内核错误。

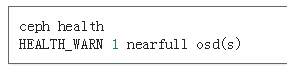
如果是软件错误（失败的插入或其它意外错误），就应该回馈到 [ceph-devel](mailto:ceph-devel%40vger.kernel.org) 邮件列表。

### 21.3.3 硬盘没剩余空间

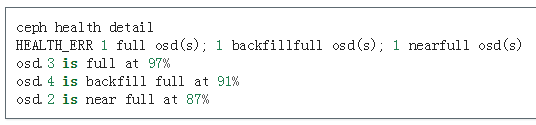
Ceph 不允许你向满的 OSD 写入数据，以免丢失数据。在运行着的集群中，集群空间将满时，你应该能收到警告。 mon osd fullratio 默认为 0.95 、或达到 95% 时它将阻止客户端写入数据。  mon osd backfillfull ratio 默认为0.90，即阻止回填时容量的90%。mon osd nearfull ratio 默认为 0.85 ，也就是说达到容量的 85% 时它会产生健康警告。

满载集群问题一般产生于在小型集群上测试 Ceph如何处理 OSD 失败的问题时。当某一节点利用率较高时，集群会很轻易地达到将满和占满率。如果你在测试小型集群上测试Ceph如何处理OSD失败的问题，应该保留足够的磁盘空间，然后试着临时降低 mon osd full ratio ，mon osd backfillfull ratio和 mon osd nearfull ratio 值。

ceph health 会显示将满的 ceph-osds ：



或者：



处理这种情况的最好方法就是增加新的 ceph-osd ，这允许集群把数据重分布到新 OSD 里。

如果因满载而导致 OSD 不能启动，你可以试着删除那个 OSD 上的一些归置组（placement group）数据目录。

**Important**

如果你准备从填满的 OSD 中删除某个归置组，注意**不要**删除另一个OSD 上的同名归置组，否则**你会丢数据**。**必须**在多个 OSD 上保留至少一份数据副本。

详情见[监视器配置参考](http://docs.ceph.org.cn/rados/configuration/mon-config-ref)。

## 21.4 OSD 龟速或无响应

一个反复出现的问题是龟速或无响应。在深入性能问题前，你应该先确保不是其他故障。例如，确保你的网络运行正常、且 OSD 在运行，还要检查 OSD 是否被恢复流量拖住了。

**Tip**

较新版本的 Ceph 能更好地处理恢复，可防止恢复进程耗尽系统资源而导致 up 且 in 的 OSD 不可用或响应慢。

### 21.4.1 网络问题

Ceph 是一个分布式存储系统，所以它依赖于网络来互联 OSD 们、复制对象、恢复错误、和检查心跳。网络问题会导致 OSD 延时和打摆子，详情参见[打摆子的 OSD](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-osd/#id22) 。

确保 Ceph 进程和 Ceph 依赖的进程连接了、和/或在监听。

netstat -a | grep ceph

netstat -l | grep ceph

sudo netstat -p | grep ceph

检查网络统计信息。

netstat -s

### 21.4.2 驱动器配置

一个存储驱动器应该只用于一个 OSD 。如果有其它进程共享驱动器，顺序读和顺序写吞吐量会成为瓶颈，包括日志记录、操作系统、监视器、其它 OSD 和非 Ceph 进程。

Ceph 在日志记录完成之后才会确认写操作，所以使用 ext4 或 XFS 文件系统时高速的 SSD 对降低响应延时很有吸引力。相反， btrfs 文件系统可以同时读写。

**Note**

给驱动器分区并不能改变总吞吐量或顺序读写限制。把日志分离到单独的分区可能有帮助，但最好是另外一块硬盘的分区。

### 21.4.3 坏扇区和碎片化硬盘

检修下硬盘是否有坏扇区和碎片。这会导致总吞吐量急剧下降。

### 21.4.4 监视器和 OSD 蜗居

监视器是普通的轻量级进程，但它们会频繁调用 fsync() ，这会妨碍其它工作量，特别是监视器和 OSD 共享驱动器时。另外，如果你在 OSD 主机上同时运行监视器，遭遇的性能问题可能和这些相关：

* 运行较老的内核（低于3.0）
* v0.48 版运行在老的 glibc 之上
* 运行的内核不支持 syncfs(2) 系统调用

在这些情况下，同一主机上的多个 OSD 会相互拖垮对方。它们经常导致爆炸式写入。

### 21.4.5 进程蜗居

共存于同一套硬件、并向 Ceph 写入数据的进程（像基于云的解决方案、虚拟机和其他应用程序）会导致 OSD 延时大增。一般来说，我们建议用一主机跑 Ceph 、其它主机跑其它进程，实践证明把 Ceph 和其他应用程序分开可提高性能、并简化故障排除和维护。

### 21.4.6 日志记录级别

如果你为追踪某问题提高过日志级别、但结束后忘了调回去，这个 OSD 将向硬盘写入大量日志。如果你想始终保持高日志级别，可以考虑给默认日志路径挂载个硬盘，即 /var/log/ceph/$cluster-$name.log 。

### 21.4.7 恢复节流

根据你的配置， Ceph 可以降低恢复速度来维持性能，否则它会不顾 OSD 性能而加快恢复速度。检查下 OSD 是否正在恢复。

### 21.4.8 内核版本

检查下你在用的内核版本。较老的内核也许没有移植能提高 Ceph 性能的功能。

### 21.4.9 内核与 SYNCFS 问题

试试在一主机上只运行一个 OSD ，看看能否提升性能。老内核未必支持有 syncfs(2) 系统调用的 glibc 。

### 21.4.10 文件系统问题（★英文文档改动）

目前，我们推荐使用XFS部署集群。

我们建议不要使用btrfs或ext4。btrfs文件系统具有许多有吸引力的功能，但是文件系统中的错误可能会导致性能问题和出色的ENOSPC错误。我们不推荐ext4，因为xattr的大小限制打破了我们对长对象名称（RGW所需）的支持。

有关更多信息，请参阅[文件系统建议](http://docs.ceph.com/docs/master/rados/troubleshooting/configuration/filesystem-recommendations)。

### 21.4.11 内存不足

我们建议为每 OSD 进程规划 1GB 内存。你也许注意到了，通常情况下 OSD 仅会用一小部分（如 100-200MB ）。你也许想用这些空闲内存跑一些其他应用，如虚拟机等等，然而当 OSD 进入恢复状态时，其内存利用率激增，如果没有可用内存，此 OSD 的性能将差的多。

### 21.4.12 old request或slow request

如果某 ceph-osd 守护进程对一请求响应很慢，它会生成日志消息来抱怨请求耗费的时间过长。默认警告阀值是 30 秒，用 osd opcomplaint time 选项来配置。这种情况发生时，集群日志系统会收到这些消息。

很老的版本抱怨 “old requests” ：

osd.0 192.168.106.220:6800/18813 312 : [WRN] old request osd\_op(client.5099.0:790 fatty\_26485\_object789 [write 0~4096] 2.5e54f643) v4 received at 2012-03-06 15:42:56.054801 currently waiting for sub ops

较新版本的 Ceph 抱怨 “slow requests” ：

{date} {osd.num} [WRN] 1 slow requests, 1 included below; oldest blocked for > 30.005692 secs

{date} {osd.num} [WRN] slow request 30.005692 seconds old, received at {date-time}: osd\_op(client.4240.0:8 benchmark\_data\_ceph-1\_39426\_object7 [write 0~4194304] 0.69848840) v4 currently waiting for subops from [610]

可能起因有：

* 坏驱动器（查看 dmesg 输出）；
* 内核文件系统缺陷（查看 dmesg 输出）；
* 集群过载（检查系统负载、 iostat 等等）；
* ceph-osd 守护进程缺陷。

可能的解决方法：

* 从 Ceph 主机去除 VM 云解决方案；
* 升级内核；
* 升级 Ceph ；
* 重启 OSD 。

### 21.4.13 调试慢请求（★英文文档新增）

如果运行“ceph daemon osd.<id> dump\_historic\_ops”或“dump\_ops\_in\_flight”，您将看到一组操作和每个操作经过的事件列表。这些在下面简要描述。

来自Messenger层的事件：

* header\_read：当messenger第一次开始从线上阅读消息
* throttled：当messenger试图获取内存节流空间来将消息读入内存
* all\_read：messenger读完线上的消息
* dispatched：当messenger把消息给OSD
* Initiated：与header\_read相同。两者共存是一个历史的古怪之处。

OSD在准备操作时发生的事件：

* queued\_for\_pg：操作已经被放到队列中供PG处理
* reached\_pg：PG已经开始做这个操作
* waiting for \*：操作正在等待一些其他工作完成，然后才能继续（一个新的OSDMap；其目标为擦除；对于PG来完成对等；所有在消息中指定）
* started：操作已被接受为OSD实际上应该做的事情（不这样做的理由：失败的安全/权限检查；过时的本地状态等），现在实际上正在执行
* waiting for subops from：操作已发送到副本OSD

来自FileStore的事件：

* commit\_queued\_for\_journal\_write：操作已经被赋予FileStore
* write\_thread\_in\_journal\_buffer：操作在日志的缓冲区中并等待被保留（当下一个磁盘写入）
* journaled\_completion\_queued：操作被记录到磁盘，其回调排队等待调用

任务给本地磁盘后，OSD中的事件：

* op\_commit：操作由主要的OSD提交（即写入日志）
* op\_applied：操作已经写入到后备FS（即在内存中应用，但没有刷新到磁盘）在主要的OSD上
* sub\_op\_applied：操作被应用，但是对于副本的“子操作”（subop）
* sub\_op\_committed：操作被提交，但是对于副本的子操作（仅用于EC池）
* sub\_op\_commit\_rec/sub\_op\_apply\_rec from <X>：当听到上述内容时的主要标记，但是对于特定的副本<X>
* commit\_sent：我们向客户端发送了一个回复（或主要的OSD，对于子操作）

这些事件中的很多看起来是多余的，但跨越了内部代码中的重要边界（例如跨越锁向新线程传递数据）。

## 21.5 FLAPPING OSDS

我们建议同时部署公网（前端）和集群网（后端），这样能更好地满足对象复制的容量需求。另一个优点是你可以运营一个不连接互联网的集群，以此避免拒绝估计。 OSD 们互联和检查心跳时会优选集群网（后端），详情见[监视器与 OSD 的交互](http://docs.ceph.org.cn/rados/configuration/mon-osd-interaction)。

然而，如果集群网（后端）失败、或出现了明显的延时，同时公网（前端）却运行良好， OSD 现在不能很好地处理这种情况。这时 OSD 们会向监视器报告邻居 down 了、同时报告自己是 up 的，我们把这种情形称为打摆子（ flapping ）。

如果有东西导致 OSD 打摆子（反复地被标记为 down ，然后又 up ），你可以强制监视器停止：

ceph osd set noup *# prevent OSDs from getting marked up*

ceph osd set nodown *# prevent OSDs from getting marked down*

这些标记记录在 osdmap 数据结构里：

ceph osd dump | grep flags

flags no-up,no-down

下列命令可清除标记：

ceph osd unset noup

ceph osd unset nodown

其它两个标记 noin 和 noout 也是支持的，它们分别可防止正在启动的 OSD 被标记为 in（已分配数据） 、或被误标记为 out （无论 mon osd down out interval 的值是什么）。

**Note**

noup 、 noout 和 nodown 从某种意义上说是临时的，一旦标记清除了，它们被阻塞的动作短时间内就会发生；另一方面，noin 标记阻止 OSD 启动后被标记为in，但其它守护进程都维持原样。

# 22. 归置组（pg）故障排除（▲中文文档未完全翻译）

## 22.1 归置组总不整洁

在创建集群时，如果集群保持活动状态， 活动状态+重新映射状态或活动状态+降级状态，并且永远不会达到 活动状态+清除状态，则可能是配置有问题。

您可能需要查看存储池，归置组和CRUSH配置参考中的设置 并进行适当的调整。

通常，您应该使用多个OSD运行集群，并且池大小大于1个对象副本。

### 22.1.1 一个节点集群

Ceph不再提供在单个节点上运行的文档，因为您不会在单个节点上部署专为分布式计算而设计的系统。另外，在包含Ceph守护进程的单个节点上安装客户端内核模块可能会由于Linux内核本身的问题（除非您为客户端使用虚拟机）导致死锁。尽管有此处所述的限制，但您可以在单节点配置中尝试Ceph。

如果您尝试在单个节点上创建集群，则在创建监视器之前，必须将Ceph配置文件中的osd crush chooseleaf 类型设置的默认值从1（表示 主机或节点）更改为0（表示osd）屏上显示。这告诉Ceph一个OSD可以与同一主机上的另一个OSD同行。如果您尝试设置单节点集群，并且osd crush chooseleaf 类型大于0，则Ceph将根据设置来设定一个OSD上的PGs与另一个节点、机箱、机架甚至数据中心上的OSD上的PGs同行。

**Tips：**

不要将内核客户端直接安装在与Ceph存储集群相同的节点上，因为可能会产生内核冲突。但是，您可以将内核客户端安装在单个节点上的虚拟机（VM）内。

如果您使用单个磁盘创建OSD，则必须先手动为数据创建目录。例如：

mkdir / var / local / osd0 / var / local / osd1

ceph-deploy osd prepare {localhost-name}：/ var / local / osd0 {localhost-name}：/ var / local / osd1

ceph-deploy osd activate {localhost-name}：/ var / local / osd0 {localhost-name}：/ var / local / osd1

### 22.1.2 OSD少于副本数

如果您将两个OSD设置为up和in的状态，但仍没有看到active+clean的归置组，那么你可能有一个 OSD池 默认大小设置为大于2。

有几种方法可以解决这种情况。如果要以具有两个副本的active + degraded状态运行集群，则可以将 osd 池的 default min size设置为2，以便可以将对象以active + degraded状态写入。您也可以将osd 池的 default size 设置为2，以便您只有两个存储的副本（原始副本和一个副本），在这种情况下，集群应达到active+clean状态。

**注意：您可以在运行时进行更改。如果在Ceph配置文件中进行更改，则可能需要重新启动集群。**

### 22.1.3 池大小=1

如果您将osd 池的 默认 大小设置为1，则只会有一个对象的副本。OSD依靠其他OSD来告诉他们应该拥有哪些对象。如果第一个OSD有一个对象的副本，并且没有第二个副本，则没有第二个OSD可以告诉第一个OSD它应该有这个副本。对于映射到第一个OSD的每个放置组（请参阅 ceph pg dump），可以强制第一个OSD通过运行来注意它所需的放置组：

ceph pg force\_create\_pg <pgid>

### 22.1.4 CRUSH图错误

归置组不整洁的问题也可能是因为CRUSH图错误。

## 22.2 卡住的归置组

有失败时归置组会进入“degraded”（降级）或“peering”（连接建立中）状态，这事时有发生，通常这些状态意味着正常的失败恢复正在进行。然而，如果一个归置组长时间处于某个这些状态就意味着有更大的问题，因此监视器在归置组卡 （stuck） 在非最优状态时会警告。我们具体检查：

* inactive （不活跃）——归置组长时间无活跃（即它不能提供读写服务了）；
* unclean （不干净）——归置组长时间不干净（例如它未能从前面的失败完全恢复）；
* stale （不新鲜）——归置组状态没有被 ceph-osd 更新，表明存储这个归置组的所有节点可能都挂了。

你可以摆出卡住的归置组：

ceph pg dump\_stuck stale

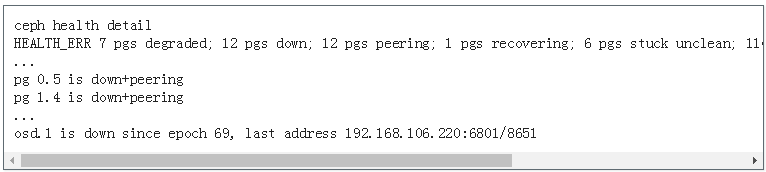
ceph pg dump\_stuck inactive

ceph pg dump\_stuck unclean

卡在 stale 状态的归置组通过修复 ceph-osd 进程通常可以修复；卡在 inactive 状态的归置组通常是互联问题（参见 [*归置组挂了——互联失败*](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-pg/#failures-osd-peering) ）；卡在 unclean 状态的归置组通常是由于某些原因阻止了恢复的完成，像未找到的对象（参见 [*未找到的对象*](http://docs.ceph.org.cn/rados/troubleshooting/troubleshooting-pg/#failures-osd-unfound)）。

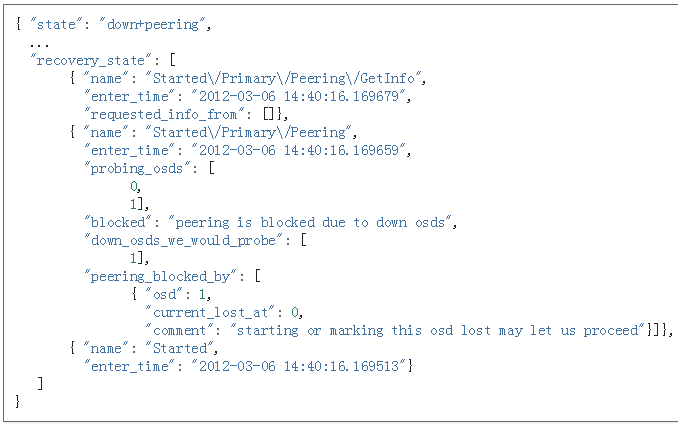
## 22.3 归置组挂了——互联失败

在某些情况下， ceph-osd 连接建立进程会遇到问题，使 PG 不能活跃、可用，例如 ceph health 也许显示：



可以查询到 PG 为何被标记为 down ：

ceph pg 0.5 query



recovery\_state 段告诉我们连接建立因 ceph-osd 进程挂了而被阻塞，本例是 osd.1 挂了，启动这个进程应该就可以恢复。

另外，如果 osd.1 是灾难性的失败（如硬盘损坏），我们可以告诉集群它丢失（ lost ）了，让集群尽力完成副本拷贝。

**Important**

集群不能保证其它数据副本是一致且最新就危险了！

让 Ceph 无论如何都继续：

ceph osd lost 1

恢复将继续。

## 22.4 未找到的对象

某几种失败相组合可能导致 Ceph 抱怨有找不到（ unfound ）的对象：

ceph health detail

HEALTH\_WARN 1 pgs degraded; 78/3778 unfound (2.065%)

pg 2.4 is active+degraded, 78 unfound

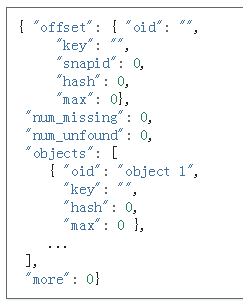
这意味着存储集群知道一些对象（或者存在对象的较新副本）存在，却没有找到它们的副本。下例展示了这种情况是如何发生的，一个 PG 的数据存储在 ceph-osd 1 和 2 上：

* 1 挂了；
* 2 独自处理一些写动作；
* 1 起来了；
* 1 和 2 重新互联， 1 上面丢失的对象加入队列准备恢复；
* 新对象还未拷贝完， 2 挂了。

这时， 1 知道这些对象存在，但是活着的 ceph-osd 都没有副本，这种情况下，读写这些对象的 IO 就会被阻塞，集群只能指望节点早点恢复。这时我们假设用户希望先得到一个 IO 错误。

首先，你应该确认哪些对象找不到了：

ceph pg 2.4 list\_missing [starting offset, in json]



如果在一次查询里列出的对象太多， more 这个字段将为 true ，因此你可以查询更多。（命令行工具可能隐藏了，但这里没有）

其次，你可以找出哪些 OSD 上探测到、或可能包含数据：

ceph pg 2.4 query

本例中，集群知道 osd.1 可能有数据，但它挂了（ down ）。所有可能的状态有：

* 已经探测到了
* 在查询
* OSD 挂了
* 尚未查询

有时候集群要花一些时间来查询可能的位置。

还有一种可能性，对象存在于其它位置却未被列出，例如，集群里的一个 ceph-osd 停止且被剔出，然后完全恢复了；后来的失败、恢复后仍有未找到的对象，它也不会觉得早已死亡的 ceph-osd 上仍可能包含这些对象。（这种情况几乎不太可能发生）。

如果所有位置都查询过了仍有对象丢失，那就得放弃丢失的对象了。这仍可能是罕见的失败组合导致的，集群在写入完成前，未能得知写入是否已执行。以下命令把未找到的（ unfound ）对象标记为丢失（ lost ）。

ceph pg 2.5 mark\_unfound\_lost revert|delete

上述最后一个参数告诉集群应如何处理丢失的对象。

delete 选项将导致完全删除它们。

revert 选项（纠删码存储池不可用）会回滚到前一个版本或者（如果它是新对象的话）删除它。要慎用，它可能迷惑那些期望对象存在的应用程序。

## 22.5 无根归置组

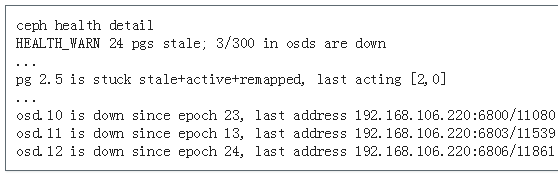
拥有归置组拷贝的 OSD 都可以失败，在这种情况下，那一部分的对象存储不可用，监视器就不会收到那些归置组的状态更新了。为检测这种情况，监视器把任何主 OSD 失败的归置组标记为 stale （不新鲜），例如：

ceph health

HEALTH\_WARN 24 pgs stale; 3/300 in osds are down

你能找出哪些归置组 stale 、和存储这些归置组的最新 OSD ，命令如下：

ceph health detail



如果想使归置组 2.5 重新在线，例如，上面的输出告诉我们它最后由 osd.0 和 osd.2 处理，重启这些 ceph-osd 将恢复之（还有其它的很多 PG ）。

## 22.6 只有几个 OSD 接收数据

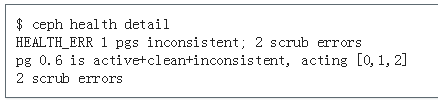
如果你的集群有很多节点，但只有其中几个接收数据，[检查](http://docs.ceph.org.cn/rados/operations/placement-groups#get-the-number-of-placement-groups)下存储池里的归置组数量。因为归置组是映射到多个 OSD 的，这样少量的归置组将不能分布于整个集群。试着创建个新存储池，其归置组数量是 OSD 数量的若干倍。详情见[归置组](http://docs.ceph.org.cn/rados/operations/placement-groups)，存储池的默认归置组数量没多大用，你可以参考[这里](http://docs.ceph.org.cn/rados/configuration/pool-pg-config-ref)更改它。

## 22.7 不能写入数据

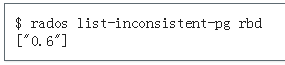
如果你的集群已启动，但一些 OSD 没起来，导致不能写入数据，确认下运行的 OSD 数量满足归置组要求的最低 OSD 数。如果不能满足， Ceph 就不会允许你写入数据，因为 Ceph 不能保证复制能如愿进行。详情参见[存储池、归置组和 CRUSH 配置参考](http://docs.ceph.org.cn/rados/configuration/pool-pg-config-ref)里的 osd pool default min size 。

## 22.8 归置组不一致（★英文文档新增详述）

如果您收到一个active + clean + inconsistent 的状态，这可能是由于清洗过程中的错误。与往常一样，我们可以通过以下方式识别不一致的PG(s)：

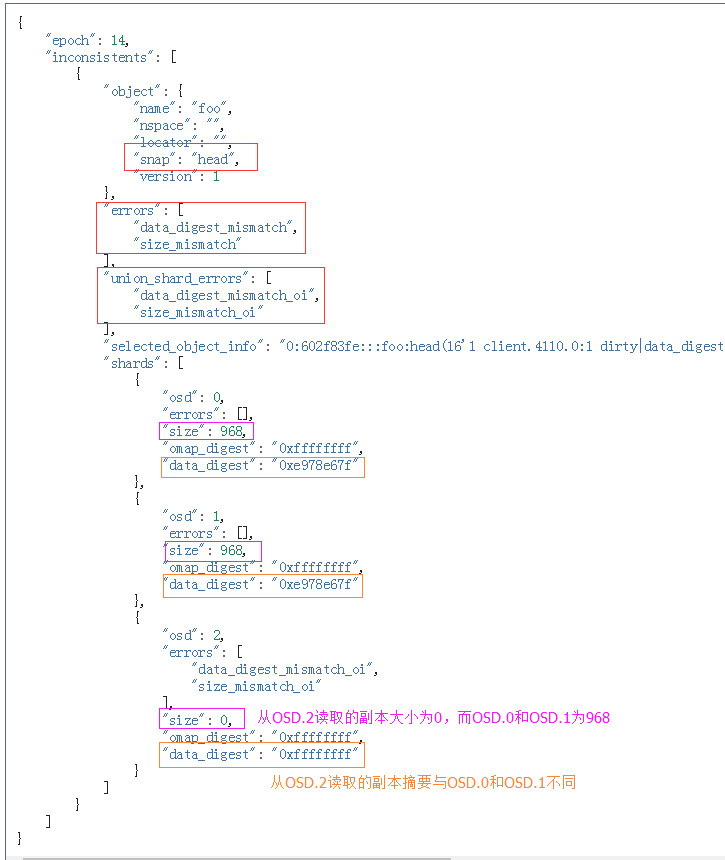


或者：rados list-inconsistent-pg rbd



只有一个一致的状态，但在最坏的情况下，我们可以在多个对象中找到多个不同的不一致性。如果名为foo的对象在PG 0.6中被截断，我们将有：

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



其中，"selected\_object\_info"这一行未显示完：

"selected\_object\_info": "0:602f83fe:::foo:head(16'1 client.4110.0:1 dirty|data\_digest|omap\_digest s 968 uv 1 dd e978e67f od ffffffff alloc\_hint [0 0 0])",

在这种情况下，我们可以从输出中看到：

* 唯一一个不一致的对象叫foo，并且不一致的地方是它的头。
* 不一致分为两类：
  + errors：这些错误表明在没有确定哪个分片不好的情况下分片之间的不一致。在*分片*数组（如果有的话）中检查errors以查明问题。
    - data\_digest\_mismatch：从OSD.2读取的副本的摘要与OSD.0和OSD.1的不同；
    - size\_mismatch：从OSD.2读取的副本大小为0，而OSD.0和OSD.1副本的大小为968。
  + union\_shard\_errors：在 shards数组中所有分片具体errors的集合。这些errors是为给定的分片有问题而设置的，包括像read\_error这一的错误。以oi结尾的errors表示用selected\_object\_info比较出来的结果。查看分片数组以确定哪个分片具有哪个错误。
    - data\_digest\_mismatch\_oi：从OSD.2的分片中读取计算出来的，存储在对象信息（object-info）中的摘要不是 0xffffffff；
    - size\_mismatch\_oi：存储在对象信息中的大小与从OSD.2中读取的大小不同。后者是0。

您可以通过执行以下操作来修复不一致的放置组：

ceph pg repair {placement-group-ID}

这一修复用权威的副本覆盖了坏的副本。在大多数情况下，Ceph能够使用一些预定义的标准从所有可用副本中选择权威副本，但是这并不总是奏效。例如，存储的数据摘要可能会丢失，并且在选择授权副本时，计算的摘要将被忽略。所以，**请谨慎使用上述命令**。

如果read\_error在分片的errors属性中列出，则不一致可能是由于磁盘错误。你可能需要检查你那一OSD使用的磁盘。

如果由于时钟歪斜而定期收到活动的 + 干净 + 不一致状态，则可以考虑在监视器主机上配置[NTP](http://en.wikipedia.org/wiki/Network_Time_Protocol)守护进程以充当对等节点。有关更多详细信息，请参阅[网络时间协议](http://www.ntp.org/)和Ceph [时钟选项](http://docs.ceph.org.cn/rados/configuration/mon-config-ref/#clock)。

## 22.9 纠删编码的归置组不是 ACTIVE+CLEAN

CRUSH 找不到足够多的 OSD 映射到某个 PG 时，它会显示为 2147483647 ，意思是 ITEM\_NONE 或 no OSD found ，例如：

[2,1,6,0,5,8,2147483647,7,4]

### 22.9.1 OSD 不够多

如果 Ceph 集群仅有 8 个 OSD ，但是纠删码存储池需要 9 个，就会显示上面的错误。这时候，你仍然可以另外创建需要较少 OSD 的纠删码存储池：

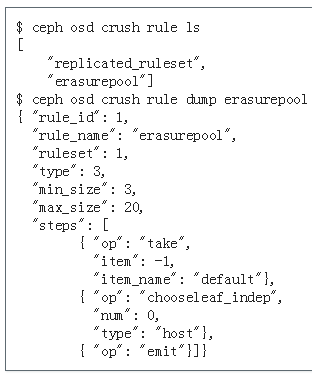
ceph osd erasure-code-profile set myprofile k=5 m=3

ceph osd pool create erasurepool 16 16 erasure myprofile

或者新增一个 OSD ，这个 PG 会自动用上的。

### 22.9.2 CRUSH 条件不能满足

即使集群拥有足够多的 OSD ， CRUSH 规则集的强制要求仍有可能无法满足。假如有 10 个 OSD 分布于两个主机上，且 CRUSH 规则集要求相同归置组不得使用位于同一主机的两个 OSD ，这样映射就会失败，因为只能找到两个 OSD ，你可以从规则集里查看必要条件：



可以这样解决此问题，创建新存储池，其内的 PG 允许多个 OSD 位于同一主机，命令如下：

ceph osd erasure-code-profile set myprofile ruleset-failure-domain=osd

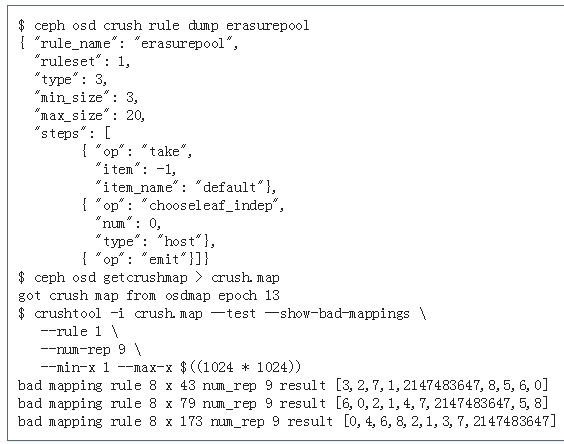
ceph osd pool create erasurepool 16 16 erasure myprofile

### 22.9.3 CRUSH 过早中止

假设集群拥有的 OSD 足以映射到 PG （比如有 9 个 OSD 和一个纠删码存储池的集群，每个 PG 需要 9 个 OSD ）， CRUSH 仍然有可能在找到映射前就中止了。可以这样解决：

* 降低纠删存储池内 PG 的要求，让它使用较少的 OSD （需创建另一个存储池，因为纠删码配置不支持动态修改）。
* 向集群添加更多 OSD （无需修改纠删存储池，它会自动回到清洁状态）。
* 通过手工打造的 CRUSH 规则集，让它多试几次以找到合适的映射。把 set\_choose\_tries 设置得高于默认值即可。

你从集群中提取出 crushmap 之后，应该先用 crushtool 校验一下是否有问题，这样你的试验就无需触及 Ceph 集群，只要在一个本地文件上测试即可：



其中 --num-rep 是纠删码 crush 规则集所需的 OSD 数量， --rule 是 ceph osd crush rule dump 命令结果中 ruleset 字段的值。此测试会尝试映射一百万个值（即 [--min-x,--max-x] 所指定的范围），且必须至少显示一个坏映射；如果它没有任何输出，说明所有映射都成功了，你可以就此打住：问题的根源不在这里。

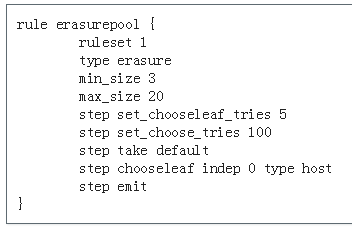
反编译 crush 图后，你可以手动编辑其规则集：

$ crushtool --decompile crush.map > crush.txt

并把下面这行加进规则集：

step set\_choose\_tries 100

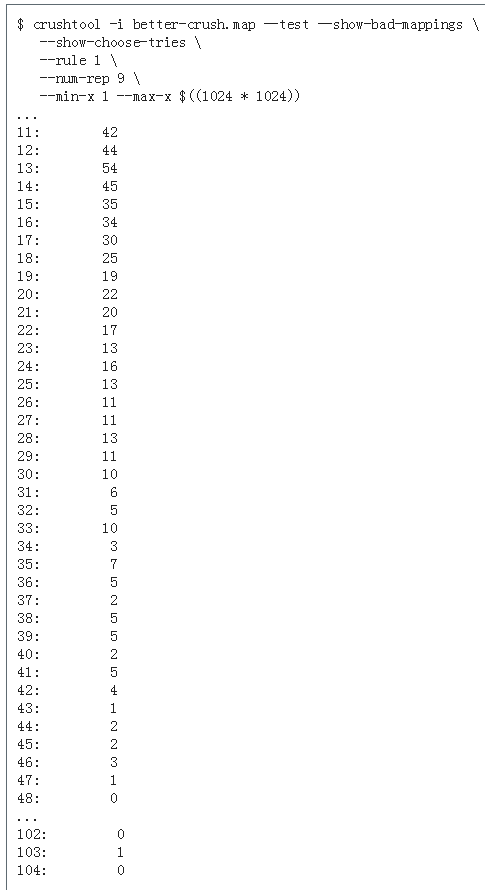
然后 crush.txt 文件内的这部分大致如此：



然后编译、并再次测试：

$ crushtool --compile crush.txt -o better-crush.map

所有映射都成功时，用 crushtool 的 --show-choose-tries 选项能看到成功映射的尝试次数直方图：



有 42 个归置组需 11 次重试、 44 个归置组需 12 次重试，以此类推。这样，重试的最高次数就是防止坏映射的最低值，也就是 set\_choose\_tries 的取值（即上面输出中的 103 ，因为任意归置组成功映射的重试次数都没有超过 103 ）。

# 23. 日志记录和调试（★英文文档命令有改动）

## 23.1 运行时

如果你想查看一进程的运行时配置，必须先登录对应主机，然后执行命令：

ceph daemon {daemon-name} config show | less

例如：

ceph daemon osd.0 config show | less

要在运行时激活 Ceph 的调试输出（即 dout() ），用 ceph tell 命令把参数注入运行时配置：

ceph tell {daemon-type}.{daemon id or \*} config set {name} {value}

用 osd 、 mon 或 mds 替代 {daemon-type} 。你可以用星号（ \* ）把配置应用到同类型的所有守护进程，或者指定具体守护进程的 ID 。例如，要给名为 ods.0 的 ceph-osd 守护进程提高调试级别，用下列命令：

ceph tell osd.0 config set debug\_osd 0/5

ceph tell 命令会贯穿所有监视器。如果你不能绑定监视器，还可以登录你要改的那台主机用 ceph daemon 来更改。例如：

sudo ceph daemon osd.0 config set debug\_osd 0/5

## 23.2 启动时

要在启动时激活调试输出（即 dout() ），你得把选项加入配置文件。各进程共有配置可写在配置文件的 [global] 下，某类进程的配置可写在守护进程段下（如 [mon] 、 [osd] 、 [mds] ）。例如：

[global]

debug ms = 1/5

[mon]

debug mon = 21

debug paxos = 1/5

debug auth = 2

[osd]

debug osd = 1/5

debug filestore = 1/5

debug journal = 1

debug monc = 5/21

[mds]

debug mds = 1

debug mds balancer = 1

debug mds log = 1

debug mds migrator = 1

## 23.3 加快日志更迭

如果你的系统盘比较满，可以修改 /etc/logrotate.d/ceph 内的日志滚动配置以加快滚动。在滚动频率后增加一个尺寸选项（达到此尺寸就滚动）来加快滚动（通过 cronjob ）。例如默认配置大致如此：

rotate 7

weekly

compress

sharedscripts

增加一个 size 选项。

rotate 7

weekly

size 500M

compress

sharedscripts

然后，打开 crontab 编辑器。

crontab -e

最后，增加一条用以检查 /etc/logrorate.d/ceph 文件。

30 \* \* \* \* /usr/sbin/logrotate /etc/logrotate.d/ceph >/dev/null 2>&1

本例中每 30 分钟检查一次 /etc/logrorate.d/ceph 文件。

## 23.4 VALGRIND

你也许还得追踪内存和线程问题，可以在 Valgrind 中运行一个守护进程、一类进程、或整个集群。 Valgrind 是计算密集型程序，应该只用于开发或调试，否则会拖慢系统。其消息记录到 stderr 。

## 23.5 子系统、日志和调试选项

大多数情况下你可以通过子系统打开调试。

### 23.5.1 CEPH 子系统概览（★英文文档内存日志级别有改动）

各子系统都有日志级别用于分别控制其输出日志、和暂存日志，你可以分别为这些子系统设置不同的记录级别。 Ceph 的日志级别从 1 到 20 ， 1 是简洁、 20 是详尽。通常，内存驻留日志不会发送到输出日志，除非：

* 致命信号冒出来了，或者：
* 源码中的 assert 被触发，或者：
* 明确要求发送。详情见[管理套接字文档](http://docs.ceph.com/docs/master/man/8/ceph/#daemon) 。

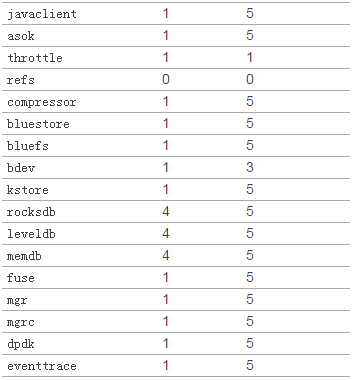
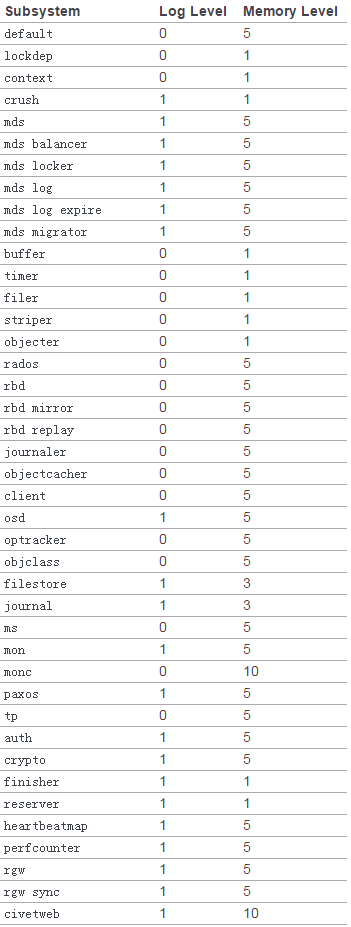
调试选项允许用单个数字同时设置日志级别和内存级别，会设置为一样。比如，如果你指定 debug ms = 5 ， Ceph 会把日志级别和内存级别都设置为 5 。也可以分别设置，第一个选项是日志级别、后一个是内存级别，二者必须用斜线（ / ）分隔。假如你想把 ms 子系统的调试日志级别设为 1 、内存级别设为 5 ，可以写为 debug ms = 1/5 ，如下：

debug {subsystem} = {log-level}/{memory-level}

*#for example*

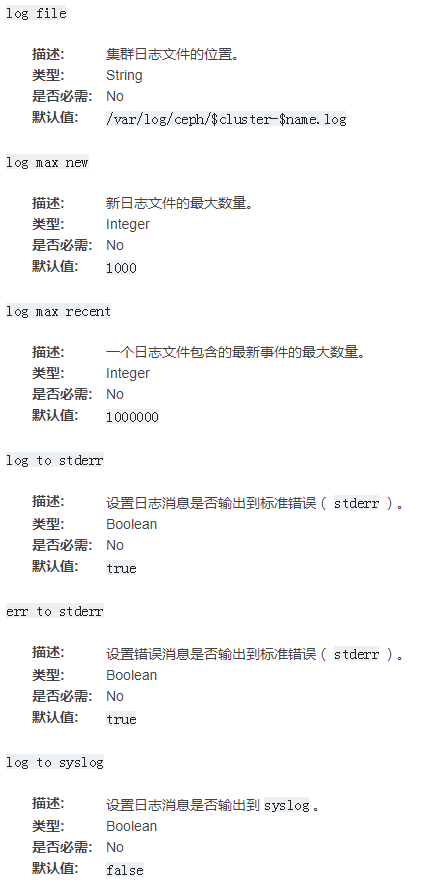
debug mds log = 1/20

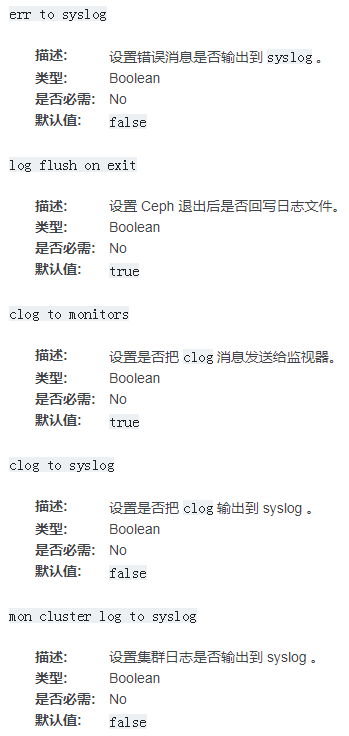
下表列出了 Ceph 子系统及其默认日志和内存级别。一旦你完成调试，应该恢复默认值、或一个适合平常运营的级别。



### 23.5.2 日志记录选项

日志和调试选项不是必需配置，但你可以按需覆盖默认值。 Ceph 支持如下配置：



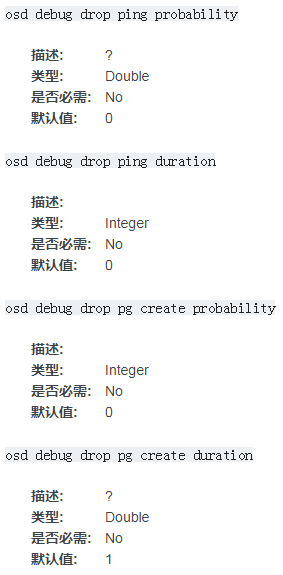


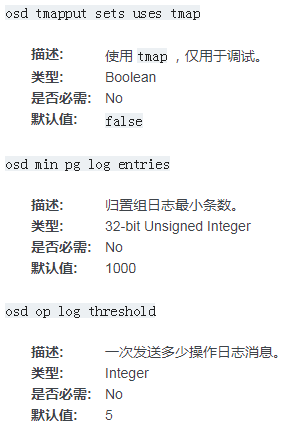
还有一项：

**mon cluster log file**

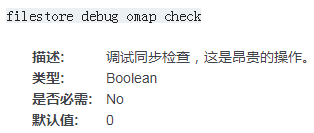
|  |  |
| --- | --- |
| 描述： | 集群的日志文件的位置。Ceph中有两个channel：cluster和audit。 此选项表示从channel到日志文件的映射，该channel的日志条目将发送到该日志文件。 默认条目是未明确指定的通道的后备映射。 因此，以下默认设置将把集群日志发送到$ cluster.log，并将审计日志发送到$ cluster.audit.log，其中$ cluster将被替换为实际集群名称。 |
| 类型： | String |
| 是否必须： | 否 |
| 默认值： | default=/var/log/ceph/$cluster.$channel.log,cluster=/var/log/ceph/$cluster.log |

### 23.5.3 OSD





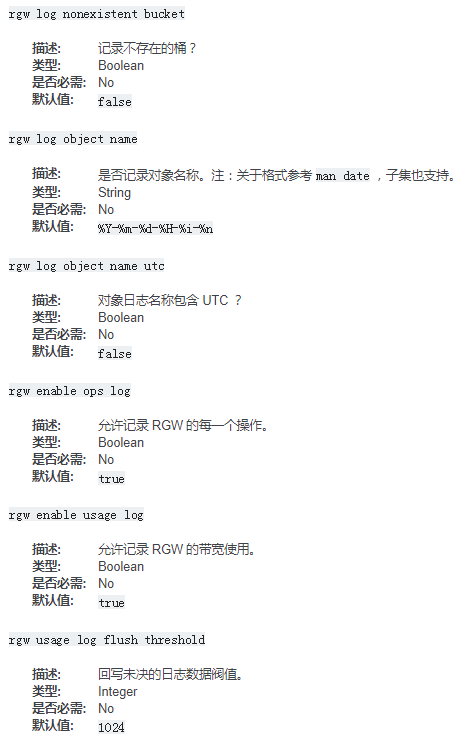
### 23.5.4 FILESTORE

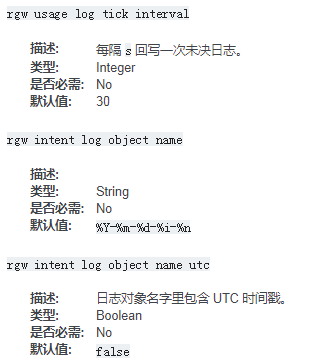


### 23.5.5 MDS



### 23.5.6 RADOS 网关





# 24. CPU剖析

如果你从源码编译时启用了 [oprofile](http://oprofile.sourceforge.net/about/) ，那就可以剖析 Ceph 的 CPU 使用情况，详情见[安装 Oprofile](http://docs.ceph.org.cn/dev/cpu-profiler) 。

## 24.1 初始化 OPROFILE

你首次使用 oprofile 时要初始化，找到对应于当前运行内核的 vmlinux 映像：

ls /boot

sudo opcontrol --init

sudo opcontrol --setup --vmlinux={path-to-image} --separate=library --callgraph=6

## 24.2 启动 OPROFILE

执行下面的命令启动 oprofile ：

opcontrol --start

启动 oprofile 后，你可以运行一些 Ceph 测试。

## 24.3 停止 OPROFILE

执行下面的命令停止 oprofile ：

opcontrol --stop

## 24.4 查看 OPROFILE 运行结果

要查看 cmon 最近的结果，执行下面的命令：

opreport -gal ./cmon | less

要检索 cmon 最近的调用图结果，执行下面的命令：

opreport -cal ./cmon | less

**Important**

回顾结果后，重新剖析前应该先重置，重置 oprofile 动作从会话目录里删除了数据。

## 24.5 重置 OPROFILE

要重置 oprofile ，执行下面的命令：

sudo opcontrol --reset

**Important**

你应该分析后再重置，以免混合不同的剖析结果。

# 25. 内存剖析

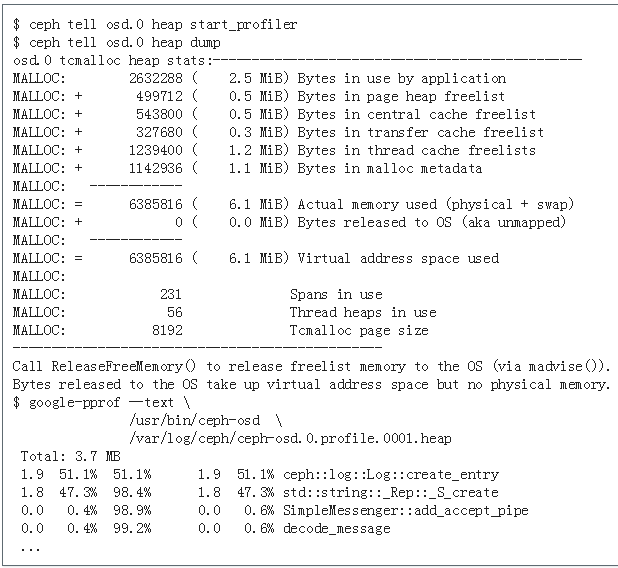
Ceph 监视器、 OSD 、和元数据服务器可利用 tcmalloc 生成堆栈剖析，此功能依赖 google-perftools ：

sudo apt-get install google-perftools

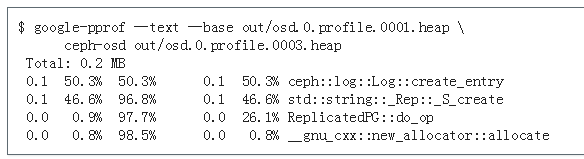
剖析器会把输出保存到 log file 目录（如 /var/log/ceph ），详情见20.[日志记录和调试](http://docs.ceph.org.cn/rados/troubleshooting/log-and-debug)。剖析器日志可用 Google 性能工具来查看，执行如下命令：

google-pprof --text {path-to-daemon} {log-path/filename}

例如：



再次转储堆栈会生成另外一个文件，这样便于和前面的堆栈转储相比较，看看这段时间发生了什么。例如：



安装堆栈剖析器后，启动集群就可以开始使用了。你可以在运行时启用、或禁用堆栈剖析器，或确保它在持续运行。在随后的几个命令行用法中，用 mon 、 osd 或 mds 替换掉 {daemon-type} ，用 OSD 号、监视器或元数据服务器的 ID 替换掉 {daemon-id} 。

## 25.1 启动剖析器

要启动堆栈剖析器，执行如下命令：

ceph tell {daemon-type}.{daemon-id} heap start\_profiler

例如：

ceph tell osd.1 heap start\_profiler

另外，在启动守护进程时若设置了 CEPH\_HEAP\_PROFILER\_INIT=true 环境变量，剖析器也会启动。

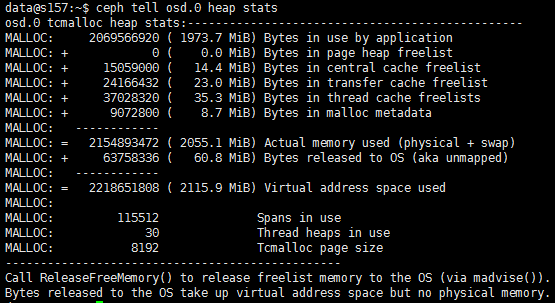
## 25.2 打印统计信息

用下列命令打印统计状态：

ceph tell {daemon-type}.{daemon-id} heap stats

例如：

ceph tell osd.0 heap stats



**Note**

打印状态不要求剖析器在运行，也不会把堆栈分配信息转储到文件。

## 25.3 转储堆栈信息

用下列命令转储堆栈信息：

ceph tell {daemon-type}.{daemon-id} heap dump

例如：

ceph tell mds.a heap dump

**Note**

只能在剖析器运行的时候转储堆栈信息。

## 25.4 释放内存

要释放由 tcmalloc 分配、但不是被 Ceph 守护进程使用的内存，用下列命令：

ceph tell {daemon-type}{daemon-id} heap release

例如：

ceph tell osd.2 heap release

## 25.5 停止剖析器

要停止堆栈剖析器，执行下列命令：

ceph tell {daemon-type}.{daemon-id} heap stop\_profiler

例如：

ceph tell osd.0 heap stop\_profiler